Savitribai Phule Pune University

T. Y. B. B. A. (C. A.) Semester-VI

(CBCS 2019 Pattern)

Advanced Java, Android / Dot Net Framework

CA-606: Lab Book

Student Name: ____

College Name: ____

Roll No.: ____________ **Division:** ___________ **Seat No:** ___________

Academic Year: ______________________________

CERTIFICATE

This is to certify that Mr./Ms.________________________________

Seat Number ___________ of T.Y.B.B.A. (C.A) Sem-VI has successfully completed Laboratory course (Advanced Java, Android/ Dot Net Framework in the year ____________________. He / She has scored _____mark out of 10 (For Lab Book).

Subject Teacher	**H.O.D./Coordinator**
Internal Examiner	**External Examiner**

Introduction

1. **About the work book:**
   This workbook is intended to be used by T.Y.B.B.A. (C.A.) Semester-VI students for Advanced java, Android and Dot Net Framework Practical assignments. This workbook is designed by considering all the practical topics mentioned in syllabus.

2. **The objectives of this workbook are:**
   - Defining the scope of the course.
   - To bring the uniformity in the practical conduction and implementation in all colleges affiliated to SPPU.
   - To have continuous assessment of the course and students.
   - Providing ready reference for the students during practical implementation.
   - Provide more options to students so that they can have good practice before facing the examination.
   - Catering to the demand of slow and fast learners and accordingly providing the practice assignments to them.

3. **How to use this workbook:**
   The workbook is divided into four sections. Section-I is related to Advanced Java assignments, Section-II is related to Android or DotNet Framework assignments, Section III is related with RIT assignments and Section IV is related with Soft skill Assignments .

   Section-I: Advanced java is divided into five assignments.
   Section-II: Android is divided into six assignments.
   OR
   Section-II: Dot Net Framework is divided into five assignments.
   Section III: RIT Assignments
   Section IV: Soft skill Assignments

   Students have to perform practical assignments of selected elective subject from Section-II.

   Each assignment of all sections has three SETs-A, B and C. It is mandatory for students to complete SET A and SET B in lab. It also includes practice set which are expected to be solved by students as home assignments and to be evaluated by subject teachers.

4. **Instructions to the students:**
   Please read the following instructions carefully and follow them during practical.
   - Students are expected to carry this workbook every time they come to the lab for computer practical.
   - Students should prepare for the assignment by reading the relevant material which is mentioned in ready reference and the concepts taught in class.
   - Instructor will specify which problems to solve in the lab during the allotted slot and student should complete them and get verified by the instructor. However, student should spend additional hours in Lab and at home to cover all workbook assignments

if needed.

- Students will be assessed for each exercise on a scale from 0 to 5.

Not Done	0
Incomplete	1
Late Complete	2
Needs improvement	3
Complete	4
Well Done	5

5. Instruction to the Instructors:

Make sure that students should follow above instructions.

- Explain the assignment and related concepts in around ten minutes using whiteboard if required or by demonstrating the software.
- Evaluate each assignment carried out by a student on a scale of 5 as specified above by ticking appropriate box.
- The value should also be entered on assignment completion page of the respective Lab course.

6. Instructions to the Lab administrator:

You have to ensure appropriate hardware and software is made available to each student.

The operating system and software requirements on server side and also client side are as given below:

- Operating System - Windows
- jdk 1.7 onwards. Tomcat 7.0 onwards
- Android
- Dot Net Framework

Assignment Completion Sheet

Section-I: Advanced Java			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	JDBC Programming		
2	Multithreading		
3	Socket Programming		
4	JSP and Servlet		
5	Spring and Hibernate		
Total (Out of 25)			
Total (Out of 5)			

Instructor Signature:

Section-II: Android			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	Introduction to Android		
2	Activity, Layout and Intent		
3	Android User Interface and Event Handling		
4	Android TimePicker, DatePicker, Alert Dailog		
5	Android Adapter and Menu		
6	Android Threads and Services		
7	Android Databases – Sqlite		
8	Location-Based Services and Google Maps		
Total (Out of 40)			
Total (Out of 5)			

'OR'

Section-II: Dot Net Framework			
Sr. No.	**Assignment Name**	**Marks (out of 5)**	**Teacher's Sign**
1	DOT NET FRAMEWORK		
2	VB.Net		
3	C#.Net		
4	ASP.Net		
5	ADO.Net		
Total (Out of 25)			
Total (Out of 5)			

Instructor Signature:

Section – I

Advanced Java

Assignment No. 1: Introduction to JDBC

Introduction:

It is technique used to connect java application either with DSN or Database. For JDBC application java.sql.* package has to import.

Jdbc Process:

For connecting java program with either database or DSN JDBC Process is used. It consists of following steps:

1. **To import JDBC-API.**
   The name of jdbc-api is java.sql. It consists of following classes and interfaces which are require for the JDBC application. The List is as follow:
   Interfaces:
   - Connection
   - Statement
   - PreparedStatement
   - CollableStatement
   - ResultSet
   - ResultSetMetaData
   - DatabaseMetaData

   Classes:
   - DriverManager
   - Class
2. **To load the driver.**
   Loading is a process used to bring the drivers from operating system to JVM. For loading drivers following statement is used:

   Class.forName ("sun.jdbc.odbc.JdbcOdbcDriver");
   The forName() is a static method of class ***Class*** used to load given driver into the JVM, If driver is not available then it throws ClassNotFoundException.

3. **To establish the connection between java program and Database or DSN (Data Source Name).**

   Connection cn=DriverManager.getConnection("jdbc:odbc:DSN", "username", "password");

   The getConnection () is a static method of DriverManager class used to set connection between java program and database or DSN.

4. **To create a Statement.**
   Statement st=cn.createStatement ();

Statement interface is used to execute database queries after making the connection with database. It is used to execute static query. The Statement interface has following methods:

- boolean execute()
- int executeUpdate()
- ResultSet executeQuery()

5. **To generate the result.**

Consider Emp (Eno, EName, Salary) table.
ResultSet rs = st . executeQuery("select * from emp");
After execution of database query, data from Emp table will get store into ResultSet object rs.

Emp Table

ENo	**EName**	**Salary**
101	Advait	70,000
102	Ajay	60,000
103	Vikas	80,000

ResultSet rs

ENo	**EName**	**Salary**
101	Advait	70,000
102	Ajay	60,000
103	Vikas	80,000

For reading data record wise from ResultSet interface next() method is used.
rs.next()

101 Advait 70,000

Index of first column in ResultSet interface is 1. For reading data, column wise from ResultSet interface getXXX() method is used.

The getXXX() method is available in 3 forms:

getInt()	:	It reads numeric value.
getFloat()	:	It reads float value.
getString()	:	It reads all kinds of values.

rs.getString(2); => ***Advait***

6. **To close the Connection.**
cn.close();

DSN (Data Source Name):

It can be created as follow:

1. Open Control Panel + Administrative Tools + Data Sources.
2. Click on Add Button.
3. Select appropriate driver for Access Database. (Microsoft Access Driver (*.mdb).
4. Give name to the DSN and select database for which DSN has to be created.
5. Click on ok button. DSN will be added into the list.

JDBC Components:

1. **DriverManager Class:**

- The DriverManager class acts as an interface between user and drivers.
- It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.
- The DriverManager class maintains a list of Driver classes that have registered themselves by calling the method **DriverManager.registerDriver ().**

Methods used in DriverManager class

Method	**Description**
public static void registerDriver(Driver driver)	It is used to register the given driver with DriverManager.
public static void deregisterDriver(Driver driver)	It is used to deregister the given driver (drop the driver from the list) with DriverManager.
public static Connection getConnection(String url)	It is used to establish the connection with the specified url.
public static Connection getConnection(String url, String userName, String password)	It is used to establish the connection with the specified url, username and password.

Syntax:

Connection cn= DriverManager.getConnection(ConnectionString,userID,Password);

2. Connection Interface:

- A Connection is the session between java application and database.
- The Connection interface is a factory of Statement, PreparedStatement, and DatabaseMetaData i.e. object of Connection can be used to get the object of Statement and DatabaseMetaData.

- ➢ The Connection interface provide many methods for transaction management like commit(), rollback() etc.
- ➢ By default, connection commits the changes after executing queries.

Methods used in Connection Interface:

Method	Description
public Statement createStatement()	Creates a statement object that can be used to execute SQL queries.
public Statement createStatement(int resultSetType,int resultSetConcurrency)	Creates a Statement object that will generate ResultSet objects with the given type and concurrency.
public void setAutoCommit(boolean status)	It is used to set the commit status. By default it is true.
public void commit()	It saves the changes made since the previous commit/rollback permanent.
public void rollback()	Drops all changes made since the previous commit/rollback.
public void close()	closes the connection and Releases a JDBC resources immediately.

Syntax:

Connection cn; //creating a Connection object

3. Statement Interface:

Statement st = cn.createStatement (); **//creating Statement object**
There are three types of Statements

a. Statement
b. PreparedStatement
c. CallableStatement

a. Statement Interface:

- ➢ The Statement interface provides methods to execute queries with the database.
- ➢ The statement interface is a factory of ResultSet i.e. it provides factory method to get the object of ResultSet.

➢ The important methods of Statement interface are as follows:

Method	Description
public ResultSet executeQuery(String sql)	It is used to execute SELECT query. It returns the object of ResultSet.
public int executeUpdate(String sql)	It is used to execute specified query, it may be create, drop, insert, update, delete etc.
public boolean execute(String sql)	It is used to execute queries that may return multiple results.
public int[] executeBatch()	It is used to execute batch of commands.

b. PreparedStatement:

➢ The PreparedStatement interface is a sub interface of Statement.

➢ It is used to execute parameterized precompiled query.

➢ Let's see the example of parameterized query:

String sql="insert into emp values(?,?,?)";

➢ Here, we are passing parameter (?) as a placeholder for the values. Its value will be set by calling the setter methods of PreparedStatement.

➢ The performance of the application will be faster if you use PreparedStatement interface because query is compiled only once.

➢ The prepareStatement () method of Connection interface is used to return the object of PreparedStatement.

➢ **Syntax**

public PreparedStatement prepareStatement(String query)throws SQLException{}
The important methods of PreparedStatement interface are given below:

Method	Description
public void setInt(int paramIndex, int value)	Sets the integer value to the given parameter index.
public void setString(int paramIndex, String value)	Sets the String value to the given parameter index.
public void setFloat(int paramIndex, float value)	Sets the float value to the given parameter index.
public void setDouble(int	Sets the double value to the given parameter

paramIndex, double value)	index.
public int executeUpdate()	Executes the query. It is used for create, drop, insert, update, delete etc.
public ResultSet executeQuery()	Executes the select query. It returns an instance of ResultSet.

c. CallableStatement Interface:

- CallableStatement interface is used to call the stored procedures and functions.
- For Example: Suppose you need the get the age of the employee based on the date of birth, you may create a function that receives date as the input and returns age of the employee as the output.
- The prepareCall() method of Connection interface returns the instance of CallableStatement.
- **Syntax:**
  public CallableStatement prepareCall("{ call procedurename(?,?...?)}");

  - **Syntax:**
    CallableStatement stmt=cn.prepareCall("{call Disp(?,?)}");

    Here it calls the procedure **Disp** that receives 2 arguments.

- **Difference between Statement, PreparedStatement and CallableStatement In Java :**

Statement	**PreparedStatement**	**CallableStatement**
It is used to execute normal SQL queries.	It is used to execute parameterized or dynamic SQL queries.	It is used to call the stored procedures.
It is preferred when a particular SQL query is to be executed only once.	It is preferred when a particular query is to be executed multiple times.	It is preferred when the stored procedures are to be executed.
You cannot pass the parameters to SQL query using this interface.	You can pass the parameters to SQL query at run time using this interface.	You can pass 3 types of parameters using this interface. They are – IN, OUT and IN OUT.
This interface is mainly used for DDL statements like CREATE, ALTER,	It is used for any kind of SQL queries which are to be executed multiple times.	It is used to execute stored procedures and functions.

DROP etc.		
The performance of this interface is very low.	The performance of this interface is better than the Statement interface (when used for multiple execution of same query).	The performance of this interface is high.

4. **ResultSet Interface**

- The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.
- ResultSet object can be moved forward only and it is not updatable.
- But we can make this object to move forward and backward direction by passing either TYPE_SCROLL_INSENSITIVE or TYPE_SCROLL_SENSITIVE in **createStatement(int, int)** method as well as we can make this object as updatable.
- **There are three Types of ResultSet:**

Type	**Description**
ResultSet.TYPE_FORWARD_ONLY (Default)	The cursor can only move forward in the result set.
ResultSet.TYPE_SCROLL_INSENSITIVE	The cursor can scroll forward and backward, and the result set is not sensitive to changes made by others to the database that occur after the result set was created.
ResultSet.TYPE_SCROLL_SENSITIVE.	The cursor can scroll forward and backward, and the result set is sensitive to changes made by others to the database that occur after the result set was created.

For Example

```
Statement st = cn.createStatement (ResultSet.TYPE_SCROLL_INSENSITIVE,
                                   ResultSet.CONCUR_UPDATABLE);
```

- Syntax: *Using Statement Interface*

```
Statement st = cn.createStatement();

ResultSet rs = st.executeQuery("select * from Emp");
```

Using PreparedStatement Interface

```
String sql = "select * from Emp";
PreparedStatement ps = cn.prepareStatement(sql);
ResultSet rs = st.executeQuery();
```

- Methods used in ResultSet interface:

Method	Description
public boolean next()	It is used to move the cursor to the one row next from the current position.
public boolean previous()	It is used to move the cursor to the one row previous from the current position.
public boolean first()	It is used to move the cursor to the first row in result set object.
public boolean last()	It is used to move the cursor to the last row in result set object.
public boolean absolute(int row)	It is used to move the cursor to the specified row number in the ResultSet object.
public boolean relative(int row)	It is used to move the cursor to the relative row number in the ResultSet object, it may be positive or negative.
public int getInt(int columnIndex)	It is used to return the data of specified column index of the current row as int.
public int getInt(String columnName)	It is used to return the data of specified column name of the current row as int.
public String getString(int columnIndex)	It is used to return the data of specified column index of the current row as String.
public String getString(String columnName)	It is used to return the data of specified column name of the current row as String.

5. **Closing the resources:**
   - **Once all the data is retrieved and all the operations are completed. We must close all the resources**

- **For Example:**

```
st.close(); //Statement closed
rs.close(); //ResultSet close
cn.close(); //Connection closed
```

Examples:

1. **Write a Menu Driven Java program for the following:**
    1. Create
    2. Insert
    3. Update
    4. Delete
    5. Display
    6. Exit

Solution:

```
import java.sql.*;
import java.io.*;
class DBDemo
{
  public static void main(String args[]) throws Exception
  {
    String sql,sn;
    int r,p,k,ch;
    BufferedReader br=new BufferedReader (new InputStreamReader(System.in));
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    Connection cn=DriverManager.getConnection("jdbc:odbc:DSN","","");
    Statement st=cn.createStatement();
  do
  {
      System.out.println("1. Create");
      System.out.println("2. Insert");
      System.out.println("3. Update");
      System.out.println("4. Delete");
      System.out.println("5. Display");
      System.out.println("6. Exit");
      System.out.println("Enter Your Choice");
      ch=Integer.parseInt(br.readLine());
      switch(ch)
      {
        case 1:
              sql="create table stud (rno number, sname text, per number)";
              st.execute(sql);
              System.out.println("Table Is Created...!");
        break;
        case 2:
                System.out.println("RNo Sname Per");
                r=Integer.parseInt(br.readLine());
                sn=br.readLine();
                p=Integer.parseInt(br.readLine());
                sql="insert into stud values("+ r + ",'"+ sn +"'," + p +")";
                k=st.executeUpdate(sql);
                if(k>0)
                   System.out.println("Record Is Added...!");
          break;
          case 3:
                System.out.println("RNo And Per");
                r=Integer.parseInt(br.readLine());
                p=Integer.parseInt(br.readLine());
                sql="update stud set per=" + p + " where rno=" + r;
                k=st.executeUpdate(sql);
                if(k>0)
                   System.out.println("Record Is Updated...!");
         break;
         case 4:
                System.out.println("RNo");
                r=Integer.parseInt(br.readLine());
                sql="delete from stud where rno=" + r;
                k=st.executeUpdate(sql);
                if(k>0)
                   System.out.println("Record Is Deleted...!");
         break;
         case 5:
                     System.out.println("Records from Table are...");
                     ResultSet rs=st.executeQuery("select * from stud");
                     while(rs.next())
                     {
                     System.out.println(rs.getString(1) + " " + rs.getString(2) + " " +
                    rs.getString(3));
                     }
         break;
         case 6:
               cn.close();
               System.exit(0);
      }
     }
     while(ch!=6);
   }
  }
```

2. Write a java program for the implementation of DDL Commands.(Use Swing)
Solution:

```
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
import javax.swing.*;
class DDLDemo extends Frame implements ActionListener
```

```
{
  Button b1,b2,b3;
  TextField t1;
  String sql;
  Label l1;
  Connection cn;
  Statement st;
  public DDLDemo()
  {
     setLayout(null);
     l1=new Label("Type Your Query Here");
     t1=new TextField();
     b1=new Button("Create");
     b2=new Button("Alter");
     b3=new Button("Drop");

     l1.setBounds(100,100,100,30);
     t1.setBounds(220,100,300,30);
     b1.setBounds(100,140,100,30);
     b2.setBounds(220,140,100,30);
     b3.setBounds(340,140,100,30);

     add(l1);
     add(t1);
     add(b1);
     add(b2);
     add(b3);

      setSize(500,500);
      setVisible(true);
      setTitle("DDL Commands");

      b1.addActionListener(this);
      b2.addActionListener(this);
      b3.addActionListener(this);
     try
     {
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       cn=DriverManager.getConnection("jdbc:odbc:DSN","","");
       st=cn.createStatement();
     }
     catch(Exception obj)
     {
     }
   }
   public void actionPerformed(ActionEvent ae)
   {
       try
```

```
        {
          Button b=(Button)ae.getSource();
          if(b==b1)
           {
             sql=t1.getText();
             st.execute(sql);
             JOptionPane.showMessageDialog(null," Table Is Created");
           }
         if(b==b2)
          {
            sql=t1.getText();
            st.execute(sql);
            JOptionPane.showMessageDialog(null, "Table Is Altered");
          }
         if(b==b3)
          {
            sql=t1.getText();
            st.execute(sql);
           JOptionPane.showMessageDialog(null, "Table Is Deleted");
          }
        }
        catch(Exception obj)
        {
        }
    }
    public static void main(String args[]) throws Exception
    {
      new DDLDemo();
    }
  }
```

3. Write a java program for the implementation of Scrollable ResultSet.

Solution:

```
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
class ScrImp extends JFrame implements ActionListener
{
  JButton b1,b2,b3,b4;
  JTextField t1,t2,t3;
  JLabel l1,l2,l3;
  String sql,en;
  int e,s,i;
  Connection cn;
  ResultSet rs;
  Statement st;
  public ScrImp()
```

```
{
  setLayout(null);
  l1=new JLabel("ENo");
  l2=new JLabel("EName");
  l3=new JLabel("Salary");
  t1=new JTextField();
  t2=new JTextField();
  t3=new JTextField();
  b1=new JButton("First");
  b2=new JButton("Next");
  b3=new JButton("Prev");
  b4=new JButton("Last");

  l1.setBounds(100,100,100,30);
  t1.setBounds(220,100,100,30);
  l2.setBounds(100,140,100,30);
  t2.setBounds(220,140,100,30);
  l3.setBounds(100,180,100,30);
  t3.setBounds(220,180,100,30);
  b1.setBounds(100,220,100,30);
  b2.setBounds(220,220,100,30);
  b3.setBounds(100,260,100,30);
  b4.setBounds(220,260,100,30);

  add(l1);
  add(t1);
  add(l2);
  add(t2);
  add(l3);
  add(t3);
  add(b1);
  add(b2);
  add(b3);
  add(b4);

  setSize(500,500);
  setVisible(true);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  b1.addActionListener(this);
  b2.addActionListener(this);
  b3.addActionListener(this);
  b4.addActionListener(this);

  try
  {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    cn=DriverManager.getConnection("jdbc:odbc:TYBBACA","","");
```

```
        st=cn.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE,
                                        ResultSet.CONCUR_UPDATABLE);
        rs=st.executeQuery("select * from emp");
        rs.next();
        t1.setText(rs.getString(1));
        t2.setText(rs.getString(2));
        t3.setText(rs.getString(3));
    }
    catch(Exception obj)
    {
    }
}
  public void actionPerformed (ActionEvent ae)
  {
     try
     {
        JButton b=(JButton)ae.getSource();
        if(b==b1)
        {
           rs.first();
           t1.setText(rs.getString(1));
           t2.setText(rs.getString(2));
           t3.setText(rs.getString(3));
        }
        if(b==b2)
        {
           rs.next();
           t1.setText(rs.getString(1));
           t2.setText(rs.getString(2));
           t3.setText(rs.getString(3));
        }
        if(b==b3)
        {
           rs.previous();
           t1.setText(rs.getString(1));
           t2.setText(rs.getString(2));
           t3.setText(rs.getString(3));
        }
        if(b==b4)
        {
           rs.last();
           t1.setText(rs.getString(1));
           t2.setText(rs.getString(2));
           t3.setText(rs.getString(3));
        }
    }
    catch(Exception obj)
    {
   }
 }
 public static void main(String args[])
 {
   new ScrImp();
 }
}
```

4. **Write a java program for the implementation of JTable.**
   Solution:

```
              import java.sql.*;
              import java.awt.*;
              import javax.swing.*;
              class JTab extends JFrame
              {
                String head[]={"ENo", "EName","Salary"};
                String data[][];
                int i;
                public JTab()
                {
                  i=0;
                  setSize(500,500);
                  setVisible(true);
                   try
                   {
               Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
 Connection cn = DriverManager.getConnection ("jdbc:odbc:TYBBACA", "","");
 Statement st=cn.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                         ResultSet.CONCUR_UPDATABLE);
  ResultSet rs=st.executeQuery("select * from emp");
                     rs.last();
                     int r=rs.getRow();
                     data=new String[r][3];
                     rs.first();
                     while(rs.next())
                     {
                       data[i][0]=rs.getString(1);
                       data[i][1]=rs.getString(2);
                       data[i][2]=rs.getString(3);
                       i++;
                     }
              //rs.refreshRow();
                     JTable jt=new JTable(data,head);
                     JScrollPane jp=new JScrollPane(jt);
                     add(jp);
                    }
                    catch(Exception obj)
                    {
```

```
            System.out.println(obj);
        }
      }
      public static void main(String args[])
      {
        new JTab();
      }
    }
```

5. **Write a java program to accept the details of Teacher(TID, TName, Salary) and store it into the database. (PreparedStatement Interface).**

Solution:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.sql.*;
class PDemo extends JFrame implements ActionListener
{
  JLabel l1,l2,l3;
  JTextField t1,t2,t3;
  JButton b1,b2;
  String sql,tn;
  int s,tid,k;
  Connection cn;
  PreparedStatement ps;
  public PDemo()
  {
    setLayout(null);
    l1=new JLabel("TID");
    l2=new JLabel("TName");
    l3=new JLabel("Salary");
    t1=new JTextField();
    t2=new JTextField();
    t3=new JTextField();
    b1=new JButton("Save");
    b2=new JButton("Cancel");

    l1.setBounds(100,100,100,30);
    t1.setBounds(220,100,100,30);
    l2.setBounds(100,140,100,30);
    t2.setBounds(220,140,100,30);
    l3.setBounds(100,180,100,30);
    t3.setBounds(220,180,100,30);
    b1.setBounds(100,220,100,30);
    b2.setBounds(220,220,100,30);
    add(l1);
    add(t1);
    add(l2);
    add(t2);
    add(l3);
    add(t3);
    add(b1);
    add(b2);
    setSize(500,500);
    setVisible(true);
    setTitle("Teacher Info");

    b1.addActionListener(this);
    b2.addActionListener(this);
    try
    {
       Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
       cn=DriverManager.getConnection("jdbc:odbc:DSN","","");
    }
    catch(Exception obj)
    {
    }
  }
  public void actionPerformed(ActionEvent ae)
  {
      JButton b=(JButton)ae.getSource();
      if(b==b1)
      {

      tid=Integer.parseInt(t1.getText());
      tn=t2.getText();
      s=Integer.parseInt(t3.getText());
      try
      {
         sql="insert into emp values(?,?,?)";
         ps=cn.prepareStatement(sql);
         ps.setInt(1,tid);
         ps.setString(2,tn);
         ps.setInt(3,s);

         k=ps.executeUpdate();
         if(k>0)
             JOptionPane.showMessageDialog(null,"Record Is Added....!");
      }
      catch(Exception obj)
      {
         System.out.println(obj);
      }
    }
    if(b==b2)
    {
            try
            {
              cn.close();
              System.exit(0);
            }
            catch(Exception obj)
            {
            }
          }

        }
        public static void main(String args[])
        {
            new PDemo();
        }
      }
```

Set A:

1. Write a java program to count number of records in a table.
2. Write a java program to display all the EName from Emp table. Assume Emp (ENo, EName, Salary) table is already created.
3. Write a java program to create Student table with attributes Rno, Sname, Per.
4. Write a java program to delete salary column from Emp table. Assume Emp table with attributes ENo, EName and salary is already created.
5. Write a java program to delete the details of given Teacher. Assume Teacher table with attributes tno, tname, subject is already created.

Set B:

1. Write a java program to accept the details of Hospital (HId, HName, Address, PH_No) and store it into the database. (Use Swing).
2. Write a java program to display the details of Doctor (DNO, DName, Specialization) on JTable. Assume Doctor table is already created.
3. Write a java program to make the changes in data which is in ResultSet if you make the changes in data in database.
4. Write a java program for the following:
   - ➢ Create a Table
   - ➢ Alter a Table
   - ➢ Drop a Table
5. Write a java program to accept the details of College (CID, CName, Address) and display it on next frame. (Use Swing and PreparedStatement).

Set C:

1. Write a java program for the following:
   a. Create a table.
   b. Insert
   c. Update
   d. Search
   e. Display
2. Write a java program to update the salary of given employee and display updated details in JTable. (Use Swing). Assume Emp (ENo, EName, Sal) table is already created.
3. Write a java program for the implementation of Scrollable ResultSet. Consider Teacher (TID, TName, Subject) table is already created.
4. Write a java program for the following:
   Accept the details of 5 Employees (ENo, EName, Salary), store it into the JTable by clicking on Add Button. If user clicks on Save button then data from JTable must be save into the database.
5. Write a java program to create at least 5 tables in a database. Add a column(field) in a given table. Drop given table from the database.

Assignment Evaluation:

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

Assignment No 2: MultiThreading:

A smallest unit of a program which is in executive mode is called thread. Multithreading allows concurrent execution of two or more parts of a program for maximum utilization of CPU. Each part of such program is called a thread. So, threads are light-weight processes within a process.

There are basically two types of a thread, Main thread and child thread. A thread which contains main method is called main thread and all other threads are called child threads. Execution as well as termination of a program is done from the main thread.

Methods of Thread Class:-

Method	**Description**
String getName()	Retrieves the name of running thread in the current context in String format
void start()	This method will start a new thread of execution by calling run() method of Thread/runnable object.
void run()	This method is the entry point of the thread. Execution of thread starts from this method.
void sleep(int sleeptime)	This method suspend the thread for mentioned time duration in argument (sleeptime in ms)
void yield()	By invoking this method the current thread pause its execution temporarily and allow other threads to execute.
void join()	This method used to queue up a thread in execution. Once called on thread, current thread will wait till calling thread completes its execution
boolean isAlive()	This method will check if thread is alive or dead
void interrupt()	This method interrupts this thread.
void setName(String name)	This method changes the name of this thread to be equal to the argument name
void stop()	This method stops an execution of a thread permanently.

Thread Creation:

Thread can be created as follow:

1. By extending Thread class
2. By implementing Runnable interface.

1. **Thread creation by extending the Thread class:**

```
class MyThread extends Thread
{
    MyThread()
     {
     start();
     }
public void run()
 {
 }
}
 class MainClass
  {
      public static void main(String args[])
       {
            MyThread tobj=new MyThread();
       }
 }
```

2. **Thread creation by implementing Runnable Interface:**

```
class MyThread implements Runnable
{
          Thread T1;
         MyThread()
          {
                    T1 =new Thread(this, "Name Of Thread");
                    T1.start ();
          }
          public void run()
          {
          }
 }
 class MainThread
 {
       public static void main(String args[])
       {
            MyThread tobj=new MyThread ();
       }
}
```

Thread Priority:

1. public static int MIN_PRIORITY

   It holds the minimum priority that can be given to a thread. The value for this is 1.

2. public static int NORM_PRIORITY

   It is the default priority that is given to a thread if it is not defined. The value for this is 5.

3. public static int MAX_PRIORITY

   It is the maximum priority that can be given to a thread. The value for this is 10.

Get and Set methods in Thread priority

1. public final int getPriority()

   In Java, getPriority () method is in java.lang.Thread package. It is used to get the priority of a thread.

2. public final void setPriority(int newPriority)

Synchronization:

Synchronization is a process of handling resource accessibility by multiple thread requests. The main purpose of synchronization is to avoid thread interference. At times when more than one thread try to access a shared resource, we need to ensure that resource will be used by only one thread at a time. The process by which this is achieved is called synchronization. The synchronized keyword creates a block of code referred to as critical section.

```
synchronized (Object)
 {
 }
```

Example: Write a java program for the implementation of synchronization.

```
class Table
{
        synchronized void printTable(int n)
        {
            for(int i=1;i<=10;i++)
            {
                   System.out.println(n*i);
                   try
                    {
                           Thread.sleep(400);
                    }
```

```
                catch(Exception e)
                {
                                System.out.println(e);
                }
            }
        }
    }
    class MyThread1 extends Thread
    {
        Table t;
        MyThread1(Table t)
        {
            this.t=t;
        }
        public void run()
        {
            t.printTable(5);
        }
    }
    class MyThread2 extends Thread
    {
        Table t;
        MyThread2(Table t)
        {
            this.t=t;
        }
        public void run()
        {
            t.printTable(10);
        }
    }
    public class TestSynchronization2
    {
        public static void main(String args[])
        {
            Table obj = new Table();//only one object
            MyThread1 t1=new MyThread1(obj);
            MyThread2 t2=new MyThread2(obj);
            t1.start();
            t2.start();
        }
    }
```

Solved Examples:

1. **Write a java program for bouncing ball.**

Solution:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
/*<applet code="AssBall" width="400" height="400"> </applet> */
public class AssBall extends Applet implements ActionListener, Runnable
{
  Thread t1;
  int y,cnt;
  Button b1;
  public void init()
  {
    cnt=1;
    y=400;
    t1=new Thread(this,"Ass");
    b1=new Button("Start");
    b1.setBounds(100,100,100,30);
    add(b1);
    b1.addActionListener(this);
  }
  public void actionPerformed(ActionEvent ae)
  {
    t1.start();
  }
  public void run()
  {
    try
    {
      while(true)
      {
        t1.sleep(100);
        if(y>0 && cnt==1)
        {
          y=y-5;
          repaint();
        }
        else if(y<=0 && cnt==1)
        {
```

```
                    cnt=0;
                  }
                  else if(y<=400 && cnt==0)
                  {
                      y=y+5;
                      repaint();
                  }
                  else
                     cnt=1;
                }
             }
             catch(Exception obj)
             {
             }
         }
         public void paint(Graphics g)
         {
           Random r=new Random();
           int rr=r.nextInt(255);
           int gg=r.nextInt(255);
           int bb=r.nextInt(255);
           Color c=new Color(rr,gg,bb);
           g.setColor(c);
           g.fillOval(200,y,100,100);
         }
        }
```

2. **Write a java program for drawing flag.**

   Solution:

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.util.*;
/*<applet code="AssFlag" width="400" height="400"> </applet> */
public class AssFlag extends Applet implements Runnable
{
   Thread t1;
   int x,y,x1,y1,x2,y2,x3,y3,x4,y4,x5,y5,x6,y6;
   int x7,y7,x8,y8;
   public void init()
   {

      x7=x8=100;
```

```
  y7=y8=150;
  x5=y5=x6=y6=125;
  x=x2=y2=100;

  x3=x4=150;
  y4=y3=100;
  y=100;
  x1=100;
  y1=100;
  t1=new Thread(this,"Ass");
  t1.start();

}

public void run()
{
  try
  {
    while(true)
    {
      t1.sleep(100);
      if(y1<=300)
      {
        y1=y1+5;
        repaint();
      }
      if(x2<=150)
      {
         x2=x2+5;
         repaint();
      }
      if(x4>=125 && y4<=125)
      {
        x4=x4-5;
        y4=y4+5;
        repaint();
      }
      if(x6<=150 && y6<=150)
      {
          x6=x6+5;
          y6=y6+5;
          repaint();
```

```
            }
            if(x8<=150)
            {
              x8=x8+5;
               repaint();
            }

          }
       }
       catch(Exception obj)
       {
       }
    }
    public void paint(Graphics g)
    {
      g.drawLine(x,y,x1,y1);
      g.drawLine(x,y,x2,y2);
      g.drawLine(x3,y3,x4,y4);
      g.drawLine(x5,y5,x6,y6);
      g.drawLine(x7,y7,x8,y8);
    }
  }
```

3. **Write a java program to blink image.**

   Solution:

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
class Blink extends JFrame implements ActionListener,Runnable
{
   Thread t1;
   JLabel l1;
   JButton b1;
   boolean b;
   public Blink()
   {
      b=true;
      setLayout(null);
      ImageIcon img=new ImageIcon("Koala.jpg");
      l1=new JLabel(img);
      t1=new Thread(this,"DYP");
      b1=new JButton("Blink");
      l1.setBounds(100,100,100,100);
      b1.setBounds(100,210,100,30);
      add(l1);
      add(b1);
      setSize(400,400);
      setVisible(true);
      b1.addActionListener(this);
    }
    public void actionPerformed(ActionEvent ae)
     {
        t1.start();
     }
     public void run()
     {
       try
       {
         while(true)
         {
           t1.sleep(500);
           if(b==true)
           {
             l1.setVisible(true);
             b=false;
           }
           else
           {
              l1.setVisible(false);
              b=true;
           }
         }
        }
       catch(Exception obj)
        {
        }
    }
  public static void main(String args[])
  {
    new Blink();
  }
  }
```

4. **Write a java program to display numbers between 1 to 100 in TextField. Each number should display after 2 seconds.**

Solution:

```
import java.awt.*;
import java.awt.event.*;
class NumText extends Frame implements Runnable, ActionListener
{
  TextField t1;
  Thread t;
   Button b;
  int i;
  public NumText()
  {
    setLayout(null);
    t1=new TextField();
    t=new Thread(this,"Ass7B");
    b=new Button("Start");

    t1.setBounds(100,100,100,30);
    b.setBounds(100,140,100,30);
    add(t1);
    add(b);
    b.addActionListener(this);
    setSize(400,400);
    setVisible(true);
  }
  public void actionPerformed(ActionEvent ae)
  {
     t.start();
  }
  public void run()
  {
    try
    {
       for(i=1;i<=100;i++)
       {
         t.sleep(200);
         t1.setText(Integer.toString(i));
       }
    }
    catch(Exception obj)
    {
    }
 }
```

```
    public static void main(String args[])
    {
        new NumText();
     }
   }
```

5. **Write a program to calculate the sum and average of an array of 1000 integers (generated randomly) using 10 threads. Each thread calculates the sum of 100 integers. Use these values to calculate average. [Use join method].**
   Solution:

```
import java.util.*;
class Mythread implements Runnable
{
                  Thread t;
                  int i,no,sum;
                  int a[]=new int[1000];
                  String TN="";
                  Mythread(String s,int n)
                  {
                          TN=s;
                          Random rs = new Random();
                          t=new Thread(this,s);
                          no=n;
                          int j=0;
                          for(i=1;i<=1000;i++)
                          {
                            a[j]=rs.nextInt(1000);
                            j++;
                          }
                          t.start();
                  }
                public void run()
                {
                  for(i=0;i<100;i++)
                  {
                          sum=sum+a[no];
                          no++;
                  }
                  System.out.println("Name of thread is = "+TN);
                  System.out.println("Sum = "+sum);
                  System.out.println("Avg ="+sum/100);
                }
```

```
        }
public class ThreadDemo
{
        public static void main(String[] arg) throws InterruptedException
        {
            int n=0;
            String TName[]={"One","Two","Third","Fourth","Fifth","Six", "Seven",
                                "Eighth","Nineth", "Tenth" };
            Mythread T[] = new Mythread[10];

                for(int i=0;i<10;i++)
                {
                T[i]=new Mythread(TName[i],n);
                n=n+100;
                T[i].t.join();
                }
        }
}
```

6. Java Program to create multiple threads using the Thread class.

Solution:

```
class MyThread extends Thread
{
String message;
MyThread(String message)
{
        this.message = message;
}
public void run()
{
try
{
        for(int i=1; i<=5; i++)
        {
                System.out.println(message + ”-” + i);
                Thread.sleep(5000); //sleep for 5 seconds
        }
        }
        catch(InterruptedException ie) { }
        }
        }
        public class MultipleThreadDemo
        {
                public static void main( String[] args)
                {
                    MyThread t1 = new MyThread(“One”);
                    MyThread t2 = new MyThread(“Two”);
                    System.out.println(t1);
                    System.out.println(t2); t1.start();
                    t2.start();
                    }
        }
```

7. **Java program to demonstrate Synchronization of a Thread**

   Solution:

```
        class mythread extends Thread
        {
        String msg[]={"Java", "Supports", "Multithreading", "Concept"};
        mythread(String name)
        {
        super(name);
        }
        public void run()
        {
        display(getName());
        System.out.println("Exit from "+getName());
        }
        synchronized void display(String name ) //Synchrinized method
        {
        for(int i=0;i<msg.length;i++)
        {
        System.out.println(name+msg[i]);
        }
        }
        } /* Main class */
        class MySynchro
        {
        public static void main(String args[])
        {
                mythread t1=new mythread("Thread 1: ");
                mythread t2=new mythread("Thread 2: ");
                t1.start();
                t2.start();
                System.out.println("Main thread exited");
        }
        }
```

Set A:

1. Write a multithreading program in java to display all the integers between 1 to 100 randomly after 2 seconds.
2. Write a multithreading program in java to display all the characters between Z to A after 5 seconds.
3. Write a java program to print “Hello Java” message 10 times.

4. Write a program in which thread sleep for 6 sec in the loop in reverse order from 100 to 1 and change the name of thread.
5. Write a java program to display all the vowels from a given String. Each vowel should display after 3 seconds.

Set B:

1. Program to define a thread for printing text on output screen for 'n' number of times. Create 3 threads and run them. Pass the text 'n' parameters to the thread constructor. Example:
   i. First thread prints "Hello" 10 times.
   ii. Second thread prints "Good Morning" 20 times
   iii. Third thread prints "Sir/Mam" 30 times.
2. Write a java program in multithreading to create 3 balls and bounce them vertically.
3. Write a java program in multithreading to draw five lines vertically.
4. Write a java program to create 2 cars and move them randomly from left to right.
5. Write a java program to display the characters from a given string into the TextField. Each character should be displayed after 1 second.

Set C:

1. Write a program to solve producer consumer problem in which a producer produces a value and consumer consume the value before producer generate the next value. (Hint: use thread synchronization)

2. Write a program to calculate the sum and average of an array of 1000 integers (generated randomly) using 10 threads. Each thread calculates the sum of 100 integers. Use these values to calculate average. [Use join method].

3. Write a program that implements a multi-thread application that has three threads. First thread generates random integer every 1 second and if the value is even, second thread computes the square of the number and prints. If the value is odd, the third thread will print the value of cube of the number.
4. Write a java program for drawing simple house.
5. Write a java program for drawing star on the applet container.

Assignment Evaluation:

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

3. Socket Programming

Java Socket programming is used for communication between the applications running on different JRE. Java Socket programming can be connection-oriented or connection-less.

Socket and ServerSocket classes are used for connection-oriented socket programming and DatagramSocket and DatagramPacket classes are used for connection-less socket programming.

The client in socket programming must know two information:

1. IP Address of Server
2. Port number.

Here, we are going to make one-way client and server communication. In this application, client sends a message to the server; server reads the message and prints it. Here, two classes are being used: Socket and ServerSocket. The Socket class is used to communicate client and server. Through this class, we can read and write message. The ServerSocket class is used at server-side. The accept () method of ServerSocket class blocks the console until the client is connected. After the successful connection of client, it returns the instance of Socket at server-side.

Socket class

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Important methods:

Method	**Description**
1) public InputStream getInputStream()	Returns the InputStream attached with this socket.
2) public OutputStream getOutputStream()	Returns the OutputStream attached with this socket.
3) public synchronized void close()	closes this socket

ServerSocket class

The ServerSocket class can be used to create a server socket. This object is used to establish communication with the clients.

Important methods

Method	**Description**
1) public Socket accept()	Returns the socket and establish a connection between server and client.
2) public synchronized void close()	Closes the server socket.

How to develop Socket application:

Creating Server:

To create the server application, we need to create the instance of ServerSocket class. Here, we are using 6666 port number for the communication between the client and server. You may also choose any other port number. The accept () method waits for the client. If clients connects with the given port number, it returns an instance of Socket.

```
ServerSocket ss=new ServerSocket (6666);
Socket s=ss.accept ();//establishes connection and waits for the client.
```

Creating Client:

To create the client application, we need to create the instance of Socket class. Here, we need to pass the IP address or hostname of the Server and a port number. Here, we are using "localhost" because our server is running on same system.

```
Socket s=new Socket("localhost",6666);
```

Let's see a simple of Java socket programming where client sends a text and server receives and prints it.

File: MyServer.java

```
import java.io.*;
import java.net.*;
public class MyServer
{
public static void main(String args[])
{
try
{
        ServerSocket ss=new ServerSocket(6666);
        Socket s=ss.accept();//establishes connection
        DataInputStream dis=new DataInputStream(s.getInputStream());
        String  str=(String)dis.readUTF();
        System.out.println ("message= "+str);
        ss.close();
}
catch(Exception e)
{System.out.println(e);}
}
}
```

File: MyClient.java

```
import java.io.*;
import java.net.*;
public class MyClient
{
        public static void main(String[] args)
        {
                try
                {
                        Socket s=new Socket("localhost",6666);
                        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
                        dout.writeUTF("Hello Server");
                        dout.flush();
                        dout.close();
                        s.close();
                }
                catch(Exception e){System.out.println(e);}
        }
}
```

Socket Application with IP Address of Server:
Client Application:
Establish a Socket Connection
To connect to another machine we need a socket connection. A socket connection means the two machines have information about each other's network location (IP Address) and TCP port. The java.net.Socket class represents a Socket. To open a socket:

Socket socket = new Socket ("127.0.0.1", 5000)

- The first argument – **IP address of Server**. (127.0.0.1 is the IP address of localhost, where code will run on the single stand-alone machine).
- The second argument – **TCP Port**. (Just a number representing which application to run on a server. For example, HTTP runs on port 80. Port number can be from 0 to 65535)

Communication
To communicate over a socket connection, streams are used to both input and output the data.

Closing the connection
The socket connection is closed explicitly once the message to the server is sent.

In the program, the Client keeps reading input from a user and sends it to the server until "Over" is typed.

```
import java.net.*;
import java.io.*;
public class Client
{
        private Socket socket = null;
        private DataInputStream input = null;
        private DataOutputStream out      = null;

        public Client(String address, int port)
        {
                // establish a connection
                try
                {
                        socket = new Socket(address, port);
                        System.out.println("Connected");

                        // takes input from terminal
                        input = new DataInputStream(System.in);

                        // sends output to the socket
                        out = new DataOutputStream (socket.getOutputStream());
                }
                catch(UnknownHostException u)
                {
                        System.out.println(u);
                }
                catch(IOException i)
                {
                        System.out.println(i);
                }

                // string to read message from input
                String line = "";
                // keep reading until "Over" is input
                while (!line.equals("Over"))
                {
                        try
                        {
                                line = input.readLine();
                                out.writeUTF(line);
                        }
                        catch(IOException i)
                {
                        System.out.println(i);
                }
        }

        // close the connection
        try
        {
                input.close();
                out.close();
                socket.close();
        }
        catch(IOException i)
        {
                System.out.println(i);
        }
    }
    public static void main(String args[])
    {
        Client client = new Client("127.0.0.1", 5000);
    }
}
```

Server Programming

Establish a Socket Connection

To write a server application two sockets are needed.

- A ServerSocket which waits for the client requests (when a client makes a new Socket())
- A plain old Socket is use for communication with the client.

Communication

The getOutputStream () method is used to send the output through the socket.

Close the Connection

After finishing, it is important to close the connection by closing the socket as well as input/output streams.

```
// A Java program for a Server
import java.net.*;
import java.io.*;

public class Server
{
```

```
//initialize socket and input stream
private Socket  socket = null;
private ServerSocket server = null;
private DataInputStream in      = null;
// constructor with port
public Server(int port)
{
        // starts server and waits for a connection
        try
        {
                server = new ServerSocket(port);
                System.out.println("Server started");
                System.out.println("Waiting for a client ...");
                socket = server.accept();
                System.out.println("Client accepted");
                // takes input from the client socket
                in = new DataInputStream(new BufferedInputStream
                                        (socket.getInputStream ()));
                String line = "";
                // reads message from client until "Over" is sent
                while (!line.equals("Over"))
                {
                        try
                        {
                                line = in.readUTF();
                                System.out.println(line);
                        }
                        catch(IOException i)
                        {
                                System.out.println(i);
                        }
                }
                System.out.println("Closing connection");
                // close connection
                socket.close();
                in.close();
        }
        catch(IOException i)
        {
                System.out.println(i);
        }
}
```

```
                public static void main(String args[])
                {
                        Server server = new Server(5000);
                }
            }
```

Important Points

- Server application makes a ServerSocket on a specific port which is 5000. This starts our Server listening for client requests coming in for port 5000.
- Then Server makes a new Socket to communicate with the client.
- socket c = server.accept()
- The accept() method blocks(just sits there) until a client connects to the server.
- Then we take input from the socket using getInputStream() method. Our Server keeps receiving messages until the Client sends "Over".
- After we're done we close the connection by closing the socket and the input stream.
- To run the Client and Server application on your machine, compile both of them. Then first run the server application and then run the Client application.
- For knowing IPAddress of a machine type ipconfig command on command prompt.

To run on Terminal or Command Prompt

Open two windows one for Server and another for Client

1. First run the Server application as,

```
 java Server
Server started
Waiting for a client …
```

2. Then run the Client application on another terminal as,

```
 java Client
```

It will show – Connected and the server accepts the client and shows,

```
Client accepted
```

3. Then you can start typing messages in the Client window. Here is a sample input to the Client

```
Hello
I made my first socket connection
Over
```

Which the Server simultaneously receives and shows,

```
Hello
I made my first socket connection
Over
Closing connection
```

Notice that sending “Over” closes the connection between the Client and the Server just like said before.

If you’re using Eclipse or likes of such:

1. Compile both of them on two different terminals or tabs
2. Run the Server program first
3. Then run the Client program
4. Type messages in the Client Window which will be received and shown by the Server Window simultaneously.
5. Type Over to end.

Solved Examples:

1. **Java program to display IPAddress of a Machine.**
   Solution:

```
import java.io.*;
import java.net.*;
class AssAddr
{
  public static void main(String args[]) throws Exception
  {
     System.out.println("IPAddress Is  " + InetAddress.getLocalHost());
  }
}
```

2. **Write a java program to display date of Server machine on client’s machine.**
   Solution:

Client.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Client
{
  public static void main(String args[]) throws Exception
  {
    String str;
    Socket c=new Socket(InetAddress.getLocalHost(),1234);
    DataInputStream din=new DataInputStream(c.getInputStream());
    DataOutputStream dout=new
DataOutputStream(c.getOutputStream());

     dout.writeUTF("Date Plz");
```

```
        str=din.readUTF();
       System.out.println("Date Is " + str);
     }
   }
```

Server.java

```
import java.io.*;
import java.net.*;
import java.util.*;
class Server
{
  public static void main(String args[]) throws Exception
  {
    String str;
    ServerSocket s=new ServerSocket(1234);
    System.out.println("Wait....Server is Starting");
    Socket c=s.accept();
    DataInputStream din=new DataInputStream(c.getInputStream());
    DataOutputStream dout=new DataOutputStream(c.getOutputStream());
    Date d=new Date();
   dout.writeUTF("Date Is  : + d);
  }
}
```

Set A:

1. Write a java program to display IP address of a machine.
2. Write a java program to accept a number in client application, send it to the server, server will check whether it is Armstrong or not and sends result accordingly to the client.
3. Write a java program to send “Hi” message to the Server.
4. Write a java program to accept a user name in client application and greets to the user according to the time on server machine.
5. Write a java program to accept a string in client application and display the vowels from that string on server terminal.

Set B:

1. Write a java program for standalone chatting application.
2. Write a java program to accept a filename in client application and check whether it is available on server machine or not, if it there then display its contents on client’s terminal otherwise display the message “File Not Found”. (Write Client and Server application on different terminals.)
3. Write a java program to accept a number in client application, send it to the server, server will calculates its factors and display them on the client terminal. (Each factor should display after 2 seconds). Write Client and Server application on different terminals.

Set C:

1. Write a java program for chatting application. (Use Swing for GUI). **(**Write Client and Server application on different terminals).
2. Write a java program to send and receive the messages from multiple terminals.
3. Write a java program to share a file on server machine between multiple clients.

Assignment Evaluation:

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

4. Servlet and JSP:

- **Servlet** is a server-side Java program that executes on tomcat web server.
- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.
- Servlet is server side scripting language so it does not have main() method.
- Servlet application compiled by using javac compiler explicitly.
- For developing Servlet application in java following API's must be imported:
    - javax.servlet.*;
    - javax.servlet.http.*;
    - java.io.*
- There are two types Servlet:
- GenericServlet
- HttpServlet
- The GenericServlet and HttpServlet are the classes belongs to packages javax.servlet.* and javax.servlet.http.*.

Life Cycle of Servlet

- The Servlet is a java program executed on tomcat web server. It can be created either by inheriting GenericServlet or HttpServlet class.
- Servlet life cycle contains five steps:

Step 1: Loading of Servlet

- When the web server (e.g. Apache Tomcat) starts up, the servlet container deploys and loads all the Servlet.

Step 2: Creating instance of Servlet

- Once all the Servlet classes loaded, the servlet container creates instances of each servlet class.
- Servlet container creates only one instance per Servlet and all the requests to the servlet are executed on the same servlet instance.

Step 3: Invoke init() Method

- Once all the servlet classes are instantiated, the init() method is invoked for each instantiated servlet.
- This method initializes the servlet. There are certain init parameters that you can specify in the deployment descriptor (web.xml) file.
- For example, if a servlet has value >=0 then its init() method is immediately invoked during web container startup.
- The init() method is called only once during the life cycle of servlet.

Step 4: Invoke service() Method

- Each time the web server receives a request for servlet, it creates a new thread that calls service() method.
- If the servlet is GenericServlet then the request is served by the service() method.
- If the servlet is HttpServlet then service () method receives the request and dispatches it to the correct handler method(POST or GET) based on the type of request.
- For example if it is a **GET Request** the service() method would dispatch the request to the doGet() method by calling the doGet() method with request parameters. Similarly the requests like **POST, HEAD, PUT** etc. are dispatched to the corresponding handlers doPost(), doHead(), doPut() etc. by service() method of servlet.

Step 5: Invoke destroy() Method

- When servlet container shuts down(this usually happens when we stop the web server), it unloads all the Servlet and calls destroy() method for each initialized Servlet.

Methods of Servlet

Method	Description
public void init(ServletConfig config)	Initializes the Servlet. It is the life cycle method of Servlet and invoked by the web container only once.
public void service(ServletRequest request, ServletResponse response)	Provides response for the incoming request. It is invoked at each request by the web container.
public void destroy()	Is invoked only once and indicates that Servlet is being destroyed.
public ServletConfig getServletConfig()	Returns the object of ServletConfig.
public String getServletInfo()	Returns information about Servlet such as writer, copyright, version etc.

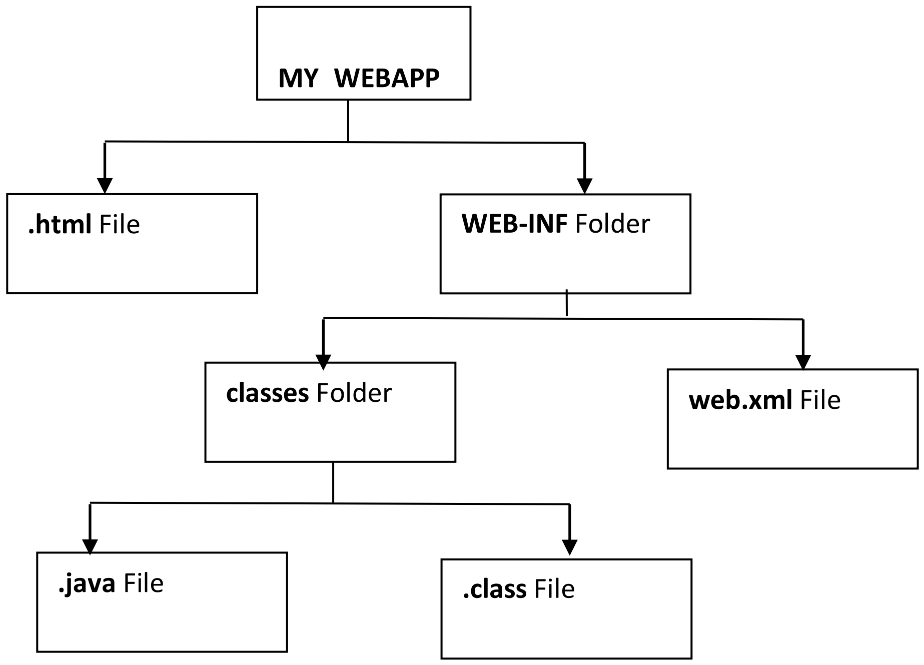
How Servlet Works?

There are three main terminology used in servlet:

- **Web Server:** It can handle HTTP Requests send by clients and responds the request with an HTTP Response.
- **Web Application (webapp):** Webapp is an application developed by you to create a servlet. It consist HTML, java, class, web.xml files.
- **Web Container:** It is also known as Servlet Container and Servlet Engine. It is a part of Web Server that interacts with Servlets. This is the main component of Web Server that manages the life cycle of Servlets.

Developing Servlet

1. **Select C:\Program Files\Apache Software Foundation\ Tomcat 8.0\webapps\ Create your own directory structure as mentioned below:**



Servlet application Directory structure

2. **Write a client side request application and save it into your web application folder with .html extension.**

   For Example: Accepting name from user.

   Save this file as "index.html"

   Note: **In form tag the action parameter should be the Servlet ".class" file name.**

```
<html>
<body>
<form method=post action="Demo">
Enter your Name<input type=text name="nm">
<input type=submit name=submit value=submit>
</form>
</body>
</html>
```

3. **The WEB-INF folder is a web information folder which holds the information about your web application such as the .class file and deployment descriptor i.e. web.xml**

```
<web-app>
 <servlet>
 <servlet-name> My_WEBAPP </servlet-name>
                         <servlet-class>Demo</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name> My_WEBAPP </servlet-name>
                    <url-pattern>/Demo</url-pattern>
 </servlet-mapping>
</web-app>
```

4. **Now write the java program and save it in classes folder. Compile it and a ".class file" will be generated.**
   Note: Classes folder must be represented in a small case letter.

   The ".class file" very important here it is used in form tag of HTML file as well as in web.xml for mapping

   For Example: In the ".html" file we used "Demo" as class file name then we must create a "Demo.java" servlet file.

   Java Servlet Program to accept name from user and display on browser.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class Demo extends HttpServlet
{
     public void doPost(HttpServletRequest request, HttpServletResponse response)
     {
                    try
                    {
                         response.setContentType ("text/html");
                         // set the type of response given to client
                         PrintWriter out = response.getWriter ();
                         // For using println() to display output
                         String name = request.getParameter ("nm"); //accepting
                          input request from client (Note: nm is a variable from .html file)
                         out.print("Hello "+name);
                         out.close();
                    }
                    catch(Exception exp)
                    {
```

```
                System.out.println (exp);
            }
        }
 }
```

Save the above program in C:\Program Files\Apache Software Foundation\Apache Tomcat 8.0.27\webapps\My_WEBAPP\WEB-INF\classes\. Compile and a class file "Demo.class:" is generated.

Methods of ServletRequest

ServletRequest Methods	**Description**
String getParameter(String name)	Obtains the value of a parameter sent to the servlet as part of a get or post request. The name argument represents the parameter name.
Enumeration getParameterNames()	Returns the names of all the parameters sent to the servlet as part of a post request.
String[] getParameterValues(String name)	For a parameter with multiple values, this method Returns an array of strings containing the values for a specified servlet parameter.
String getProtocol()	Returns the name and version of the protocol the request uses in the form *protocol/majorVersion.minorVersion*, for example, HTTP/1.
String getRemoteAddr()	Returns the Internet Protocol (IP) address of the client that sent the request.
String getRemoteHost()	Returns the fully qualified name of the client that sent the request.
int getServerPort()	Returns the port number on which request was received.
String getServerName()	Returns the host name of the server that received the request.
HttpServletRequest Methods	
Cookie[] getCookies()	Returns an array of Cookie objects stored on the client by the server.
HttpSession getSession(boolean create)	Returns an HttpSession object associated with the client's current browsing session. This method can create an HttpSession object (True argument)if one does not already exist for the client.
String getServletPath()	Returns the part of this request's URL that calls the servlet.
String getMethod()	Returns the name of the HTTP method with which this request was made, for example, GET,

	POST, or PUT.
String getQueryString()	Returns the query string that is contained in the request URL after the path.
String getPathInfo()	Returns any extra path information associated with the URL the client sent when it made this request.
String getRemoteUser()	Returns the login of the user making this request, ifm the user has been authenticated, or null if the user has not been authenticated.

ServletResponse Methods	
OutputStream getOutputStream()	Obtains a byte-based output stream for sending binary data to the client.
PrintWriter getWriter()	Obtains a character-based output stream for sending text data (usually HTML formatted text) to the client.
void setContentType(String type)	Specifies the content type of the response to the browser. The content type is also known as MIME (Multipurpose Internet Mail Extension) type of the data. For examples, "text/html", "image/gif"etc.
String setContentLength(int len)	Sets the length of the content body in the response In HTTP servlets, this method sets the HTTP Content-Length header.
HttpServletResponse Methods	
void addCookie(Cookie cookie)	Used to add a Cookie to the header of the response to the client.
void sendError(int ec)	Sends an error response to the client using the specified status.
void sendError(int ec, String msg)	Sends an error response to the client using the specified status code and descriptive message.
void sendRedirect (String url)	Sends a temporary redirect response to the client using the specified redirect location URL.
void setHeader(String name, String value)	Sets a response header with the given name and value.

To handle request and response in servlet java uses following methods:

1. **service(ServletRequest obj,ServletResponse obj):**
   - This method is supported by GenericServlet class. This is a lifecycle method which get automatically override by HttpServlet
2. **public void doPost(HttpServletRequest request, HttpServletResponse response):**

- This method is supported by HttpServlet class.
- This method throws IOException and ServletException.
- The doPost(request, response) method is executed when a web page is activated using a click event(Example: button click, hyperlink).

3. **public void doGet(HttpServletRequest request, HttpServletResponse response):**
   - This method is supported by HttpServlet class.
   - This method throws IOException and ServletException
   - The doGet(request, response) method is executed when a web page is activated or called by specifying address (URL) in web browser's address bar.

Solved Example:

1. Write a servlet application to display marksheet on to the browser.

Solution:

Index.html

```
<html>
<body>
<form method="Get" action="Mark">
<pre>
 Seat No      <input type="text" name="t1">
 Stud Name    <input type="text" name="t2">
 Class        <input type="text" name="t3">
 Total Marks  <input type="text" name="t4">
 <input type="submit" value="Submit">
</pre>
</form>
</body>
</html>
```

web.xml

```
<web-app>
            <servlet>
                <servlet-name>Ass15B</servlet-name>
                <servlet-class>Mark</servlet-class>
 </servlet>
            <servlet-mapping>
                <servlet-name>Ass15B</servlet-name>
 <url-pattern>/Mark</url-pattern>
            </servlet-mapping>
</web-app>
```

Mark.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
```

```
public class Mark extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res) throws
    IOException,ServletException
            {
                res.setContentType("text/html");
                PrintWriter pw=res.getWriter();

                String sno=req.getParameter("t1");
                String sn=req.getParameter("t2");
                String cn=req.getParameter("t3");
                int tm=Integer.parseInt(req.getParameter("t4"));

                int p=tm/6;
                String gd;

                if(p>=70)
                {
                    gd="Dist";
                }
                else if(p>=60 && p<70)
                {
                    gd="First";
                }
                else if(p>=50 && p<60)
                {
                    gd="Second";
                }
                else if(p>=40 && p<50)
                {
                    gd="Pass";
                }
                else
                gd="Fail";

                pw.println("<b>" + "Seat No " + sno);
                pw.println(<b>" + "Stud Name " + sn);
                pw.println("<b>" + "Class Name " + cn);
                pw.println("<b>" + "Percentage " + p);
                pw.println("<b>" + "Grade " + gd);
            }
}
```

INTRODUCTION TO JSP

- JSP is Java Server Page, which is a dynamic web page and used to build dynamic websites.
- To run JSP, we need web server which can be tomcat provided by apache.
- JSP is dynamic file where as Html file is static. HTML cannot get data from database or dynamic data.
- JSP can be interactive and communicate with database and controllable by programmer.
- It is saved by extension of **.jsp**. Each Java server page is compiled into a servlet before it can be used. This is normally done when the first request to the JSP page is made.
- There is no need to compile jsp page it gets compiled implicitly by JSP Engine.
- For writing JSP application its delimiters are used: Starting Delimiter (<%) and ending Delimiter(%>)

JSP Elements

A JSP contains three important types of elements:

1. **JSP Directives:**
   - Directives are message to the JSP container that enable the programmer to specify page setting to include content from other resources & to specify custom tag libraries for use in a JSP.
   - **Syntax:** <%@ name attribute1=”….”, attribute2=”…”…%>

   - **There are three types of Directives:**
     a. **Page Directive:**
        - The page directives specify global settings for the JSP in the JSP container. There can be many page directives, provided that there is only one occurrence of each attribute.
        - **Syntax:** <%@ page attribute="value" %>

For example:

```
<%@ page
[ language="java" ]
[ extends="package.class" ]
[ import="{package.class | package.*}, ..." ] [ session="true|false" ]
[ buffer="none|8kb|sizekb" ] [ autoFlush="true|false" ]
[ isThreadSafe="true|false" ] [ info="text" ]
[ errorPage="relativeURL" ]
[ contentType="mimeType [ ; charset=characterSet ]" | "text/html ; charset=ISO-8859-1"
]
[ isErrorPage="true|false" ]
[ pageEncoding="characterSet | ISO-8859-1" ]
%>
```

Program to display current Date.

```
<html>
<body>
<%@ page import="java.util.Date" %>
Today is: <%= new Date() %>
</body>
</html>
```

b. Include Directive:

- An include directive is used to include the contents of any resource it may be JSP file, HTML file or text file.
- If we want to use any file in JSP file then we can use include directive.
- An include directive includes the original content of the included resource at page translation time (the JSP page is translated only once so it will be better to include static resource).
- **Syntax:**<%@ include file="resourceName" %>
- **Note:** resourseName is any file which we want to use in current JSP file

Program to include welcome.html

```
<html>
<body>
<%@ include file="welcome.html" %>
Today is: <%= java.util.Calendar.getInstance ().getTime() %>
</body>
</html>
```

c. Taglib Directive:

- The JSP taglib directive is used to define a tag library that defines many tags.
- We use the TLD (Tag Library Descriptor) file to define the tags. In the custom tag section we will use this tag so it will be better to learn it in custom tag.
- **Syntax:** <%@ taglib uri="uriofthetaglibrary" prefix="prefixoftaglibrary" %>
- **For Example:** In this example, we are using our tag named currentDate. To use this tag we must specify the taglib directive so the container may get information about the tag.

Program to demonstrate taglib directive.

```
<html>
<body>
<%@ tagliburi="http://www.javatpoint.com/tags" prefix="mytag" %>
<mytag:currentDate/>
</body>
</html>
```

2. Scripting Elements:

- In JSP, java code can be written inside the jsp page using the scriptlet tag.
- The scripting elements provide the ability to insert java code inside the jsp.
- There are three types of scripting elements:

a. Declaration Tag:

- A declaration declares one or more variables or methods that you can use in Java code later in the JSP file.
- **Syntax:** <%! Java declaration statements %>
- **Example:**

```
<%! private int count = 0; %>
<%! int i = 0; %>
```

b. **Expressions Tag:**

- An expression element contains a java expression that is evaluated, converted to a String, and inserted where the expression appears in the JSP file.
- **Syntax:** <%= expression %>
- **Example:** Your name is <%= request.getParameter("name") %>

c. **Scriptlet Tag:**

- A scriptlet contains a set of java statements which is executed.
- A scriptlet can have java variable and method declarations, expressions, use implicit objects and contain any other statement valid in java.
- **Syntax:** <% statements %>

Write a JSP Script to display Hello message to the User.

```
<html>
<body>
<form method=”Get” action=”Name.jsp”>
User Name <input type=”text” name=”username”>
<input type=”submit” value=”Submit”>
<%
String name = request.getParameter("userName");
out.println(“Hello “ + name);%>
```

Actions: Actions encapsulates functionally in predefined tags that programmers can embedded in a JSP.

Implicit objects used in JSP:

- The implicit objects are system defined and can be used directly without declaring and initializing.

Implicit Objects used in JSP

Implicit object	Description
Application	A javax.servlet.ServletContext object that represents the container in which the JSP executes. It allows sharing information between the jsp page's Servlet and any web components within the same object.
	A javax.servlet.ServletConfig object that represents the JSP

Config	configuration options. As with Servlet, configuration options can be specified in a Web application descriptor (web.xml). The method getInitParameter() is used to access the initialization parameters.
Exception	A java.lang.Throwable object that represents an exception that is passed to a JSP error page. This object is available only in a JSP error page.
Out	A javax.servlet.jsp.JspWriter object that writes text as part of the response to a request. This object is used implicitly with JSP expressions and actions that insert string content in a response.
Page	An Object that represents the current JSP instance.
pageContext	A javax.servlet.jsp.PageContext object that provides JSP programmers with access to the implicit objects discussed in this table.
Request	An object that represents the client request and is normally an instance of a class that implements HttpServletRequest. If a protocol other than HTTP is used, this object is an instance of a subclass of javax.servlet.Servlet-Request. It uses the getParameter() method to access the request parameter.
Response	An object that represents the response to the client and is normally an instance of a class that implements HttpServletResponse (package javax.servlet.http). If a protocol other than HTTP is used, this object is an instance of a class that implements javax.servlet.ServletResponse.
Session	A javax.servlet.http.HttpSession object that represents the client session information. This object is available only in pages that participate in a session.

Solved Examples:

1. Write a JSP application to check whether given mail ID is valid or not.

   Solution:

   Index.html

```
<html>
<body>
<form method="Get" action="AssVal.jsp">
Enter EMail ID <input type="text" name="t1"> <br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

AssVal.jsp

```
<%
 String em;
 int i,l,scnt=0,dcnt=0;
 char ch;
 em=request.getParameter("t1");
 l=em.length();
 for(i=0;i<l;i++)
 {
 ch=em.charAt(i);
 if(ch=='.')
 {
 dcnt++;
 }
 if(ch=='@')
 {
 scnt++;
 }
 }
 if(dcnt==0)
 {
 out.println("Invalid....It does not contain Dot(.)");
 }
 else if(scnt>1 || scnt<1)
 {
 out.println("It does not have @ symbol or more than one @
symbol");
 }
 else
 out.println("Valid Email Id " + em);
%>
```

2. Write a JSP Application to accept nick name and Name from a user, If visit count is even then display name otherwise display nickname.

Solution:

Index.html

```
<html>
<body>
<form method="Get" action="NickName.jsp">
<pre>
 Name      <input type="text" name="t1">
 Nick Name <input type="text" name="t2">
 <input type="submit" value="Submit">
</pre>
</form>
</body>
</html>
```

```
NickName.jsp
<%! static int cnt;%>
<%
```

```
 String nm,nk;

 nm=request.getParameter("t1");
 nk=request.getParameter("t2");
 cnt++;
 if(cnt%2==0)
 out.println(nk);
 else
 out.println(nm);
%>
```

Set A

a) Design a servlet that provides information about a HTTP request from a client, such as IP address and browser type. The servlet also provides information about the server on which the servlet is running, such as the operating system type, and the names of currently loaded servlets.

b) Write a Program to make use of following JSP implicit objects:
i. out: To display current Date and Time.
ii. request: To get header information.
iii. response: To Add Cookie
iv. config: get the parameters value defined in <init-param>
v. application: get the parameter value defined in <context-param>
vi. session: Display Current Session ID
vii. pageContext: To set and get the attributes.
viii. page: get the name of Generated Servlet

c) Write a Servlet application to display Hello Java Message on the Browser.

d) Write a JSP script to display all the prime number’s between 1 to n in Red Color.

e) Write a JSP application to accept the details of Emp (Eno, EName, Salary) and display it in Tabular format on the browser.

Set B

a) Design an HTML page which passes customer number to a search servlet. The servlet searches for the customer number in a database (customer table) and returns customer details if found the number otherwise display error message.

b) Design an HTML page containing option buttons (Maths, Physics, Chemistry and Biology) and buttons submit and reset. When the user clicks submit, the server responds by adding a cookie containing the selected subject and sends a message back to the client. Program should not allow duplicate cookies to be written.

c) Write a Servlet program to display the details of PATIENT (PatientNo, PatientName, PatientAddress, Patientage, PatientDiease) in tabular form on browser.

d) Write a JSP application to accept a user name and greets to the user according to the system Time.

e) Write a Servlet program for the implementation of session tracking.

Set C

a) Create a JSP page for an online multiple choice test. The questions are randomly selected from a database and displayed on the screen. The choices are displayed using radio buttons. When the user clicks on next, the next question is displayed. When the user clicks on submit, display the total score on the screen.

b) Write a program to create an Online Book purchase. User must be login and then purchase the book. Each page should have a page total. The last page should display a total book and bill, which consists of a page total of whatever the purchase has been done and print the total. (Use HttpSession)

c). Write a JSP application to accept the details of Teacher (TID, TName, Salary) and display it in tabular format.

Assignment Evaluation:

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

5 Spring and Hibernate

What is Spring?
Spring is an open source framework created to address the complexity of enterprise application development. Spring is a very lightweight framework which provides well-defined infrastructure support for developing Java application.

Why Spring?
Spring is considered to be a secure, low-cost and flexible framework. Spring improves coding efficiency and reduces overall application development time because it is lightweight and efficient at utilizing system resources. Spring removes tedious configuration work so that developers can focus on writing program logic. Spring handles the infrastructure so developers can focus on the application.

Spring Framework
The Spring Framework is one of the most popular Java-based application Frameworks. It is an application framework and Inversion of Control (IoC) container for the Java platform. The Spring Framework is a mature, powerful and highly flexible framework focused on building Web applications in Java. The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

Spring Module

JDBC Module: This module provides the JDBC abstraction layer and helps to avoid tedious JDBC coding.

ORM Module: This module provides integration for object relational mapping APIs such as JPA, Hibernate, JDO, etc.

JMS (Java Messaging Service) Module: This module contains features for producing and consuming messages.

OXM Module: This module provides Object/XML binding.

Transaction Module: This model supports programmatic and declarative transaction management for classes that implement special interfaces and for all the POJOs.

Spring MVC

Model: A model contains the data of the application. A data can be a single object or a collection of objects. The Model encapsulates the application data and in general they will consist of POJO.

View: View is responsible for presenting data to the end user. A view represents the provided information in a particular format. The View is responsible for rendering the model data and in general it generates HTML output that the client's browser can interpret.

Controller: The controller is a logic that is responsible for processing and acting on user requests. The Controller is responsible for processing user requests and building an appropriate model and passes it to the view for rendering.

Annotations of Spring MVC

Annotation	**Description**
@Controller :	It represents the controller class
@RequestMapping:	It can be used for the mapping of incoming requests.
@GetMapping :	It is used to map HTTP Get requests.
@PostMapping :	It is used to map HTTP Post requests.
@RequestParam :	It reads the HTML form data.
@ModelAttribute :	This annotation accesses elements present

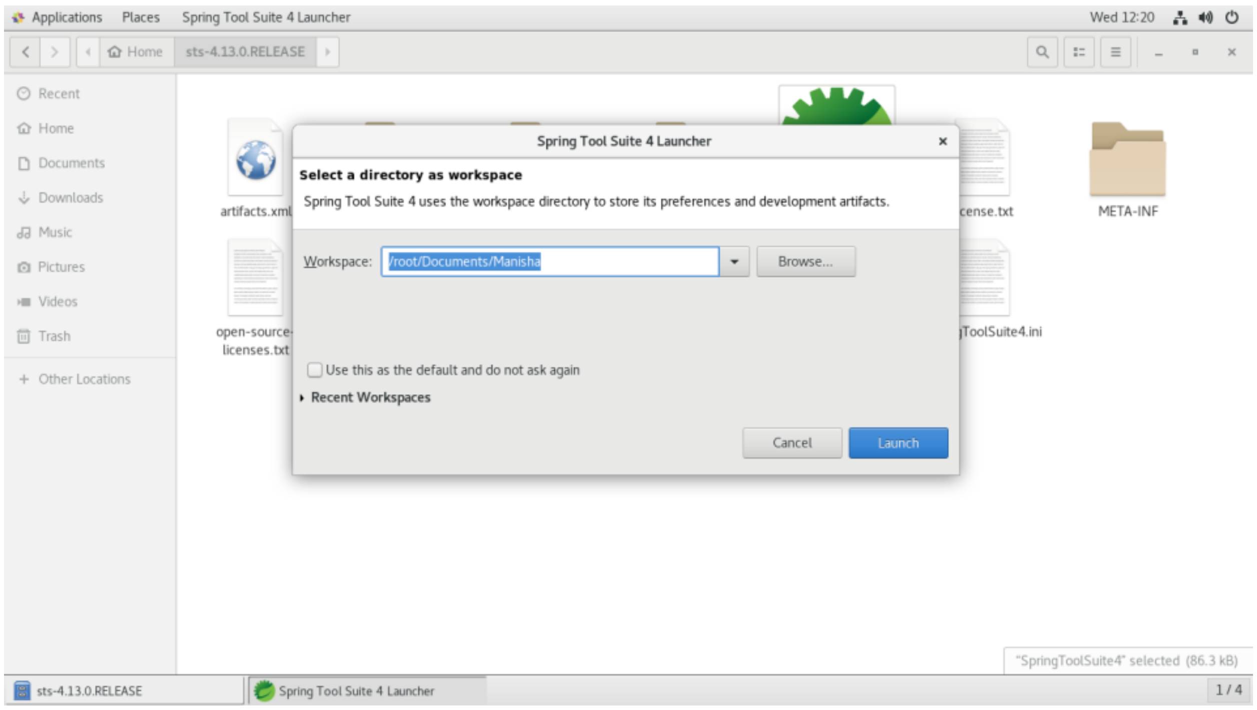
Spring MVC Validation Annotations

Annotation	**Description**
@NotNull :	Checks that the annotated value is not null.
@Min:	Must be a number >=value
@Max:	Must be a number <=value
@Size :	Total no of characters must match the given size.
@Pattern:	Must match a regular expression pattern.
@Future :	Date must be in the future of the given date.
@Past:	Date must be in the past of the given date.

pom.xml

It stands for **Project Object Model.** POM is a fundamental unit of work in Maven. Project Object Model (POM) is a XML file that contains information about the project and configuration details used by Maven to build the project. When executing a task, Maven looks for the POM in the current directory. It reads the POM, gets the required configuration and information, then executes the goal. Configurations specified in the POM are the project dependencies, the plugins or goals that can be executed, the build project.

Download the Spring Tool Suits 4 for that follow the link https://spring.io/tools
Example
Step 1: Create Java Project: The first step is to lunch the workspace and then create a simple spring starter project. Follow the option File → New → Spring Starter Project and finally select Java Project wizard from the wizard list. Lunch workspace window as shown in Fig.



Spring Tool Suite 4 Launcher

Select a directory as workspace

Spring Tool Suite 4 uses the workspace directory to store its preferences and development artifacts.

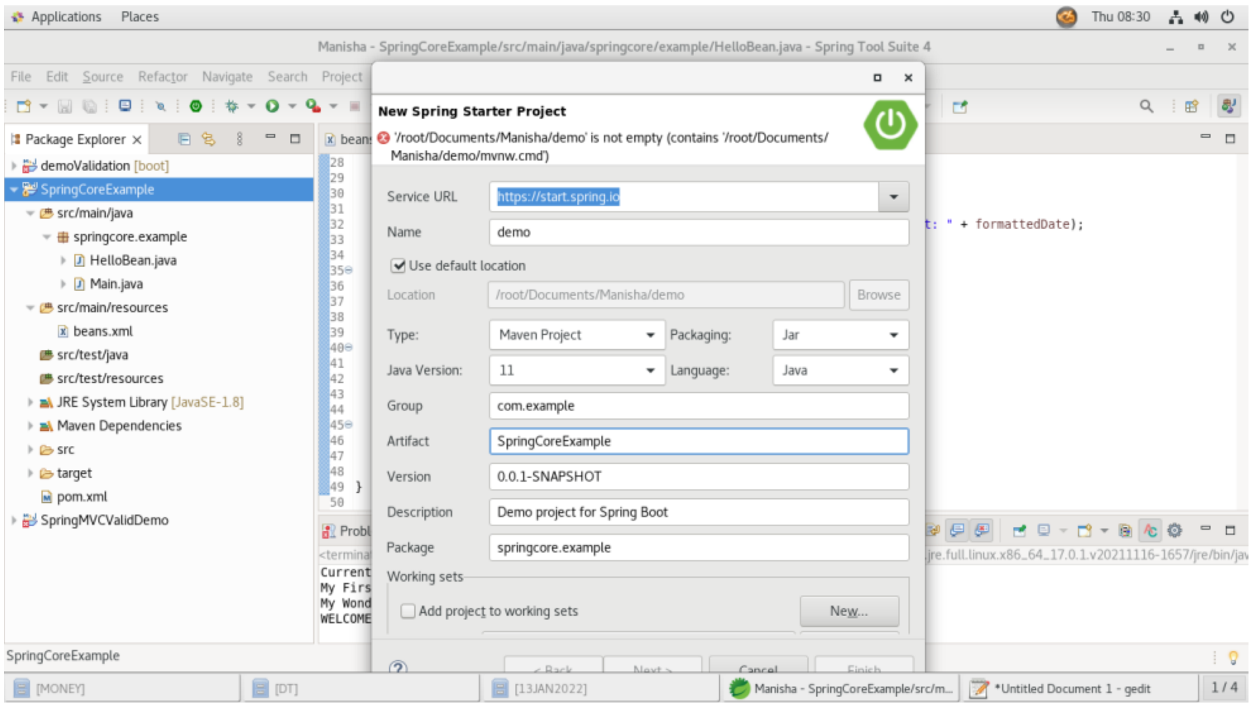
Workspace: /root/Documents/Manisha

Browse...

☐ Use this as the default and do not ask again

Recent Workspaces

Cancel Launch



```
t: " + formattedDate);
```

New Spring Starter Project

'/root/Documents/Manisha/demo' is not empty (contains '/root/Documents/Manisha/demo/mvnw.cmd')

Service URL	https://start.spring.io
Name	demo
☑ Use default location	
Location	/root/Documents/Manisha/demo
Type:	Maven Project
Packaging:	Jar
Java Version:	11
Language:	Java
Group	com.example
Artifact	SpringCoreExample
Version	0.0.1-SNAPSHOT
Description	Demo project for Spring Boot
Package	springcore.example

Working sets

☐ Add project to working sets

New...

< Back Next > Cancel Finish

Now name the project as SpringCore is created successfully.

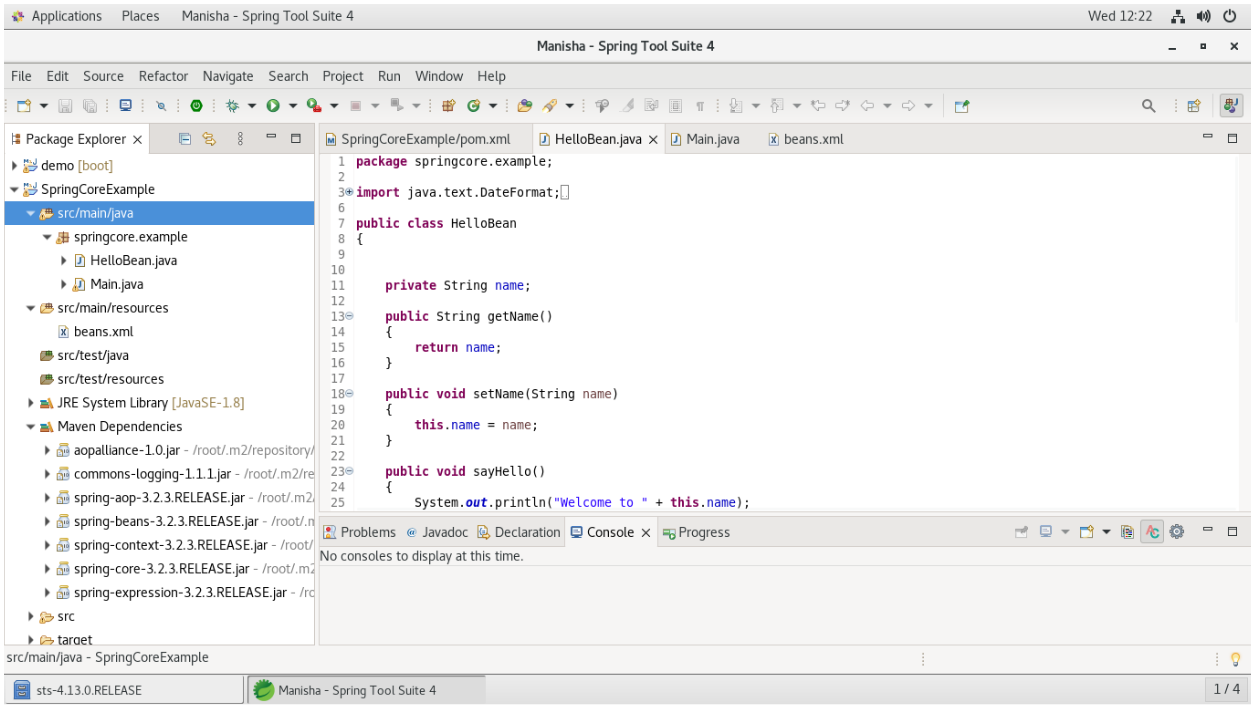
Step 2: Add Required Libraries: As a second step let us add Spring Framework and common logging API libraries in our project. To do this, right-click on the project name springcore and then follow the following option available in the context menu − Build Path - Configure Build Path to display the Java Build Path window.

Now use Add External JARs button available under the Libraries tab to add the following core JARs from Spring Framework and Common Logging installation directories:

- commons-logging-1.1.1
- spring-aop-4.1.6.RELEASE
- spring-aspects-4.1.6.RELEASE
- spring-beans-4.1.6.RELEASE
- spring-context-4.1.6.RELEASE
- spring-context-support-4.1.6.RELEASE
- spring-core-4.1.6.RELEASE
- spring-expression-4.1.6.RELEASE
- spring-instrument-4.1.6.RELEASE
- spring-instrument-tomcat-4.1.6.RELEASE
- spring-jdbc-4.1.6.RELEASE
- spring-jms-4.1.6.RELEASE
- spring-messaging-4.1.6.RELEASE
- spring-orm-4.1.6.RELEASE
- spring-oxm-4.1.6.RELEASE
- spring-test-4.1.6.RELEASE
- spring-tx-4.1.6.RELEASE
- spring-web-4.1.6.RELEASE
- spring-webmvc-4.1.6.RELEASE
- spring-webmvc-portlet-4.1.6.RELEASE
- spring-websocket-4.1.6.RELEASE

Step 3: Create Source Files: Now let us create actual source files under the SpringCore project. First we need to create a package called springcore.

Example: To do this, right click on src in package explorer section and follow the option − New Package. Next we will create HelloBean.java and Main.java files.



```
package springcore.example;

import java.text.DateFormat;

public class HelloBean
{


    private String name;

    public String getName()
    {
        return name;
    }

    public void setName(String name)
    {
        this.name = name;
    }

    public void sayHello()
    {
        System.out.println("Welcome to " + this.name);
```

```
No consoles to display at this time.
```

Here is the content of HelloBean.java file:

```
package springcore.example;
public class HelloBean
{
        private String name;
        public String getName()
        {
                return name;
        }
        public void setName(String name)
        {
                this.name = name;
        }
        public void sayHello()
        {
                System.out.println("Hello" + this.name);
        }
}
```

Content of the second file Main.java

```
//Main.java

        package springcore.example;
        import org.springframework.context.ApplicationContext;
        import org.springframework.context.support.ClassPathXmlApplicationContext;
        public class Main
        {
                private static ApplicationContext context;
                public static void main(String[] args)
                {
                        context = new ClassPathXmlApplicationContext("beans.xml");
                        HelloBean helloBean = (HelloBean) context.getBean("HelloBean");
                        helloBean.sayHello();
                }
        }
```

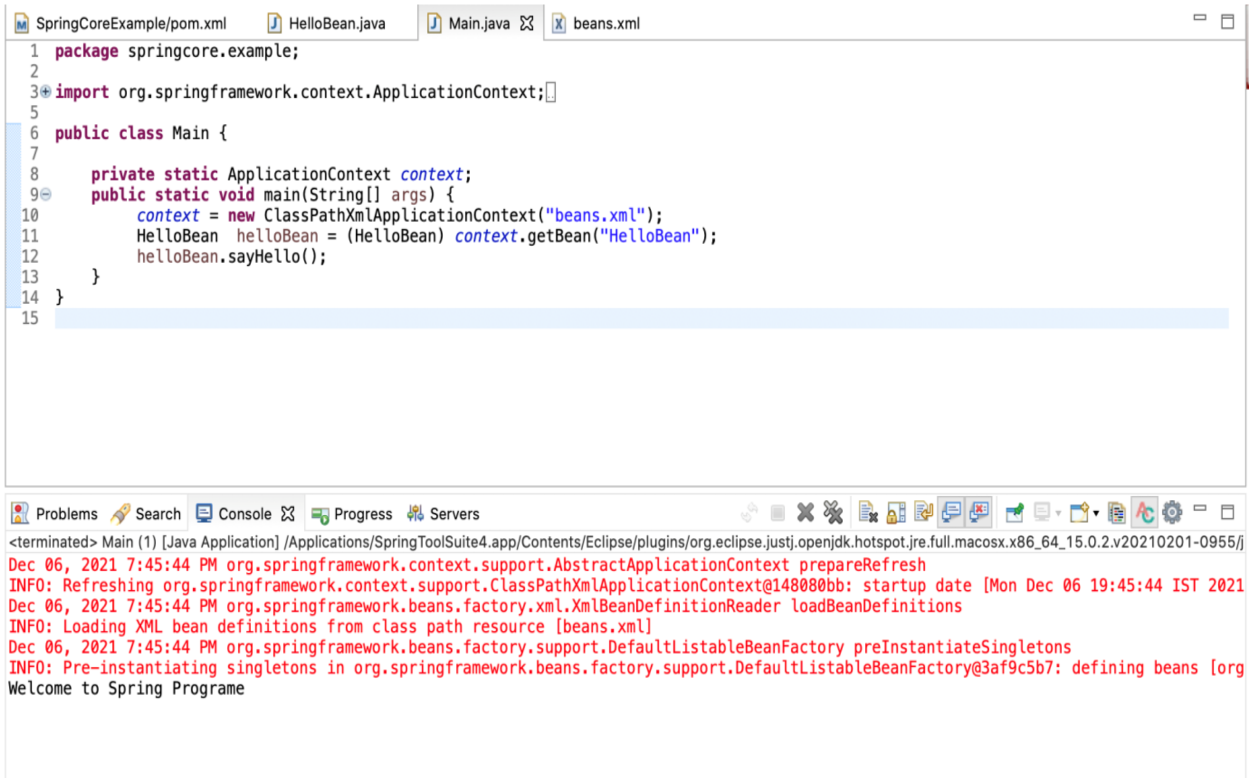
Step 4: Create Bean Configuration File: We need to create a Bean Configuration file which is an XML file and acts as cement that glues the beans, i.e. the classes together. This file needs to be created under the src directory (Src/main/resources) beans.xml.

Usually developers name this file as Beans.xml, but we are independent to choose any name we like.

```
//beans.xml
<beansxmlns="http://www.springframework.org/schema/beans"
        xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
        xmlns:p="http://www.springframework.org/schema/p"
        xmlns:aop="http://www.springframework.org/schema/aop"xmlns:context="http://ww
        w.springframework.org/schema/context"
        xmlns:jee="http://www.springframework.org/schema/jee"xmlns:tx="http://www.sprin
        gframework.org/schema/tx"
        xmlns:task="http://www.springframework.org/schema/task"
        xsi:schemaLocation="http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-3.2.xsd
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.2.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-3.2.xsd
        http://www.springframework.org/schema/jee
        http://www.springframework.org/schema/jee/spring-jee-3.2.xsd
        http://www.springframework.org/schema/tx
        http://www.springframework.org/schema/tx/spring-tx-3.2.xsd
        http://www.springframework.org/schema/task
        http://www.springframework.org/schema/task/spring-task-3.2.xsd">

        <context:component-scanbase-package="springcore.examples"/>
        <bean id="HelloBean"class="springcore.example.HelloBean">
                <propertyname="name"value="Spring Programe"/>
        </bean>
```

```
//pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
        https://maven.apache.org/xsd/maven-4.0.0.xsd">

        <modelVersion>4.0.0</modelVersion>
        <groupId>springcore_example</groupId>
        <artifactId>SpringCoreExample</artifactId>
        <version>0.0.1-SNAPSHOT</version>

<!-- JDK 8 configuration below -->
        <properties>
                <spring.version>3.2.3.RELEASE</spring.version>
                <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
                <maven.compiler.source>1.8</maven.compiler.source>
                <maven.compiler.target>1.8</maven.compiler.target>
        </properties>
<!-- completed -->
<dependencies>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-core</artifactId>
        <version>${spring.version}</version>
</dependency>
<dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>${spring.version}</version>
</dependency>
</dependencies>
</project>
```

Step 5: Running the Program: Once we are done with creating the source and beans configuration files, we are ready for this step, which is compiling and running the program. To do this, keep Main.java file tab active and use either Run option. If everything is fine with the application, this will print the following message in console.



```
package springcore.example;

import org.springframework.context.ApplicationContext;

public class Main {

    private static ApplicationContext context;
    public static void main(String[] args) {
         context = new ClassPathXmlApplicationContext("beans.xml");
         HelloBean  helloBean = (HelloBean) context.getBean("HelloBean");
         helloBean.sayHello();
    }
}

```

```
<terminated> Main (1) [Java Application] /Applications/SpringToolSuite4.app/Contents/Eclipse/plugins/org.eclipse.justj.openjdk.hotspot.jre.full.macosx.x86_64_15.0.2.v20210201-0955/j
Dec 06, 2021 7:45:44 PM org.springframework.context.support.AbstractApplicationContext prepareRefresh
INFO: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@148080bb: startup date [Mon Dec 06 19:45:44 IST 2021
Dec 06, 2021 7:45:44 PM org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
INFO: Loading XML bean definitions from class path resource [beans.xml]
Dec 06, 2021 7:45:44 PM org.springframework.beans.factory.support.DefaultListableBeanFactory preInstantiateSingletons
INFO: Pre-instantiating singletons in org.springframework.beans.factory.support.DefaultListableBeanFactory@3af9c5b7: defining beans [org
Welcome to Spring Programe
```

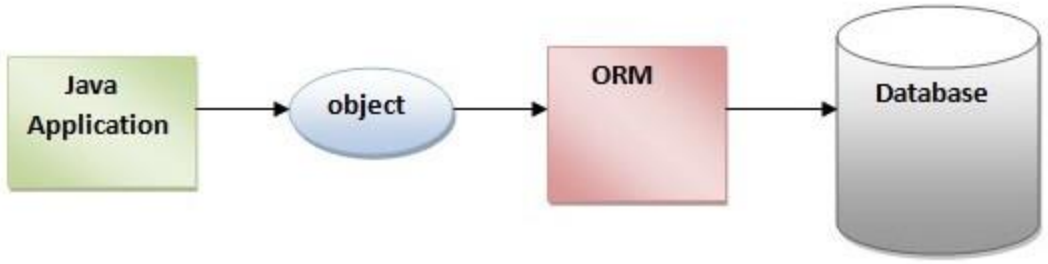
Hibernate:

Hibernate Framework

Hibernate is a Java framework that simplifies the development of Java application to interact with the database. It is an open source, lightweight, ORM (Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence API) for data persistence.

ORM Tool

An ORM tool simplifies the data creation, data manipulation and data access. It is a programming technique that maps the object to the data stored in the database.



The ORM tool internally uses the JDBC API to interact with the database.

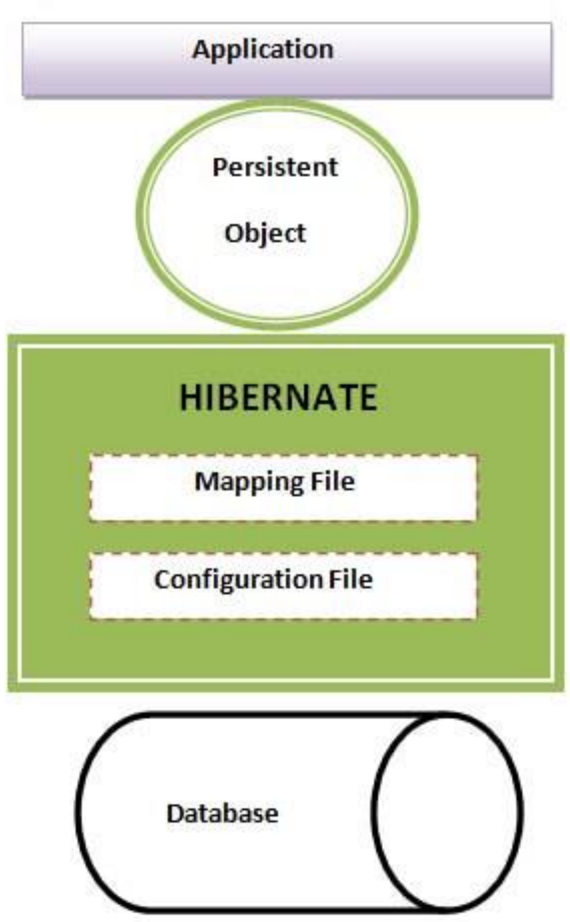
What is JPA?

Java Persistence API (JPA) is a Java specification that provides certain functionality and standard to ORM tools. The **javax.persistence** package contains the JPA classes and interfaces.

The Hibernate architecture includes many objects such as persistent object, session factory, transaction factory, connection factory, session, transaction etc.

The Hibernate architecture is categorized in four layers.

- Java application layer
- Hibernate framework layer
- Backhand api layer
- Database layer



Hibernate framework uses many objects such as session factory, session, transaction etc. alongwith existing Java API such as JDBC (Java Database Connectivity), JTA (Java Transaction API) and JNDI (Java Naming Directory Interface).

Elements of Hibernate Architecture

For creating the first hibernate application, we must know the elements of Hibernate architecture. They are as follows:

SessionFactory

The SessionFactory is a factory of session and client of ConnectionProvider. It holds second level cache (optional) of data. The org.hibernate.SessionFactory interface provides factory method to get the object of Session.

Session

The session object provides an interface between the application and data stored in the database. It is a short-lived object and wraps the JDBC connection. It is factory of Transaction, Query and Criteria. It holds a first-level cache (mandatory) of data. The org.hibernate.Session interface provides methods to insert, update and delete the object. It also provides factory methods for Transaction, Query and Criteria.

Transaction

The transaction object specifies the atomic unit of work. It is optional. The org.hibernate.Transaction interface provides methods for transaction management.

ConnectionProvider

It is a factory of JDBC connections. It abstracts the application from DriverManager or DataSource. It is optional.

TransactionFactory

It is a factory of Transaction. It is optional.

First Hibernate Example without IDE

Here, we are going to create the first hibernate application without IDE. For creating the first hibernate application, we need to follow the following steps:

1. Create the Persistent class
2. Create the mapping file for Persistent class
3. Create the Configuration file
4. Create the class that retrieves or stores the persistent object

5. Load the jar file
6. Run the first hibernate application by using command prompt

1) **Create the Persistent class**

A simple Persistent class should follow some rules:

- **A no-arg constructor:** It is recommended that you have a default constructor at least package visibility so that hibernate can create the instance of the Persistent class by newInstance() method.
- **Provide an identifier property:** It is better to assign an attribute as id. This attribute behaves as a primary key in database.
- **Declare getter and setter methods:** The Hibernate recognizes the method by getter and setter method names by default.
- **Prefer non-final class:** Hibernate uses the concept of proxies, that depends on the persistent class. The application programmer will not be able to use proxies for lazy association fetching.

Let's create the simple Persistent class:

Employee.java

```
package com.javatpoint.mypackage;
public class Employee {
private int id;
private String firstName,lastName;

public int getId() {
   return id;
}
public void setId(int id) {
   this.id = id;
}
public String getFirstName() {
   return firstName;
}
public void setFirstName(String firstName) {
   this.firstName = firstName;
}
public String getLastName() {
   return lastName;
}
public void setLastName(String lastName) {
   this.lastName = lastName;
}
}
```

2) Create the mapping file for Persistent class

The mapping file name conventionally, should be class_name.hbm.xml. There are many elements of the mapping file.

- **hibernate-mapping :** It is the root element in the mapping file that contains all the mapping elements.
- **class :** It is the sub-element of the hibernate-mapping element. It specifies the Persistent class.
- **id :** It is the subelement of class. It specifies the primary key attribute in the class.
- **generator :** It is the sub-element of id. It is used to generate the primary key. There are many generator classes such as assigned, increment, hilo, sequence, native etc. We will learn all the generator classes later.
- **property :** It is the sub-element of class that specifies the property name of the Persistent class.

Let's see the mapping file for the Employee class:

employee.hbm.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 5.3//EN"
 "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">

 <hibernate-mapping>
  <class name="com.javatpoint.mypackage.Employee" table="emp1000">
    <id name="id">
     <generator class="assigned"></generator>
    </id>

    <property name="firstName"></property>
    <property name="lastName"></property>
 </class>
</hibernate-mapping>
```

---

3) Create the Configuration file

The configuration file contains information about the database and mapping file. Conventionally, its name should be hibernate.cfg.xml .

hibernate.cfg.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
        "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
        "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

 <hibernate-configuration>
    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">jtp</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</proper
ty>
    <mapping resource="employee.hbm.xml"/>
    </session-factory>

</hibernate-configuration>
```

4) **Create the class that retrieves or stores the object**

In this class, we are simply storing the employee object to the database.

```
package com.javatpoint.mypackage;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
public class StoreData {
public static void main(String[] args) {
        //Create typesafe ServiceRegistry object
StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hiberna
te.cfg.xml").build();
      Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();
      SessionFactory factory = meta.getSessionFactoryBuilder().build();
      Session session = factory.openSession();
      Transaction t = session.beginTransaction();
      Employee e1=new Employee();
      e1.setId(101);
      e1.setFirstName("Gaurav");
```

```
                e1.setLastName("Chawla");
                session.save(e1);
                t.commit();
                System.out.println("successfully saved");
                factory.close();
                session.close();
               }
        }
```
5) **Load the jar file**
For successfully running the hibernate application, you should have the hibernate5.jar file.
Download the required jar files for hibernate

How to run the first hibernate application without IDE

We may run this hibernate application by IDE (e.g. Eclipse, Myeclipse, Netbeans etc.) or without IDE.

To run the hibernate application without IDE:

- Install the oracle10g for this example.
- Load the jar files for hibernate. (One of the way to load the jar file is copy all the jar files under the JRE/lib/ext folder). It is better to put these jar files inside the public and private JRE both.
- Now, run the StoreData class by **java com.javatpoint.mypackage.StoreData**

Hibernate Example using XML in Eclipse

Here, we are going to create a simple example of hibernate application using eclipse IDE. For creating the first hibernate application in Eclipse IDE, we need to follow the following steps:

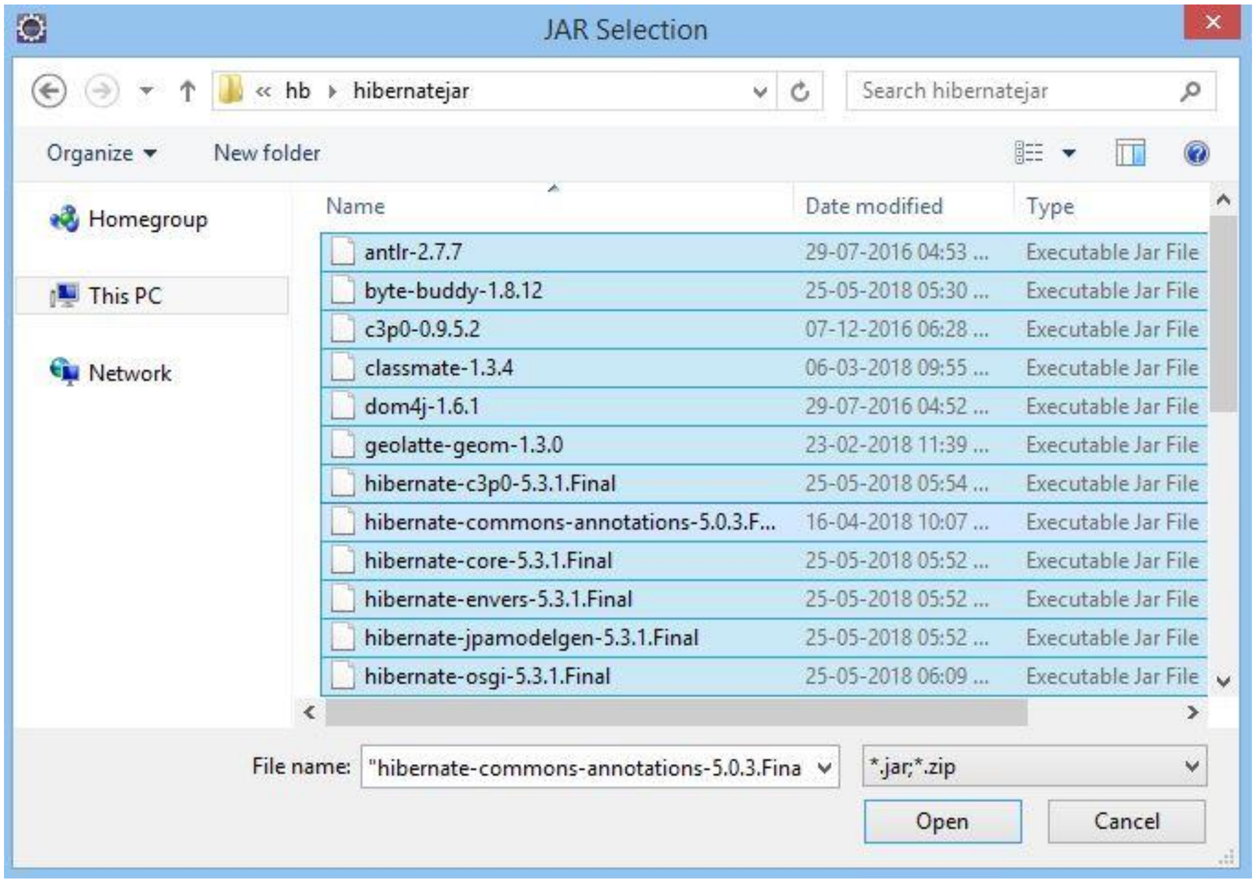
1. Create the java project
2. Add jar files for hibernate
3. Create the Persistent class
4. Create the mapping file for Persistent class
5. Create the Configuration file
6. Create the class that retrieves or stores the persistent object
7. Run the application

1) Create the java project

Create the java project by **File Menu - New - project - java project** . Now specify the project
name e.g. firsthb then **next - finish** .

2) Add jar files for hibernate

To add the jar files **Right click on your project - Build path - Add external archives**. Now select all the jar files as shown in the image given below then click open.



Name	Date modified	Type
antlr-2.7.7	29-07-2016 04:53 ...	Executable Jar File
byte-buddy-1.8.12	25-05-2018 05:30 ...	Executable Jar File
c3p0-0.9.5.2	07-12-2016 06:28 ...	Executable Jar File
classmate-1.3.4	06-03-2018 09:55 ...	Executable Jar File
dom4j-1.6.1	29-07-2016 04:52 ...	Executable Jar File
geolatte-geom-1.3.0	23-02-2018 11:39 ...	Executable Jar File
hibernate-c3p0-5.3.1.Final	25-05-2018 05:54 ...	Executable Jar File
hibernate-commons-annotations-5.0.3.F...	16-04-2018 10:07 ...	Executable Jar File
hibernate-core-5.3.1.Final	25-05-2018 05:52 ...	Executable Jar File
hibernate-envers-5.3.1.Final	25-05-2018 05:52 ...	Executable Jar File
hibernate-jpamodelgen-5.3.1.Final	25-05-2018 05:52 ...	Executable Jar File
hibernate-osgi-5.3.1.Final	25-05-2018 06:09 ...	Executable Jar File

File name: "hibernate-commons-annotations-5.0.3.Fina
.jar;.zip

In this example, we are connecting the application with oracle database. So you must add the ojdbc14.jar file.

download the ojdbc14.jar file

3) Create the Persistent class

Here, we are creating the same persistent class which we have created in the previous topic. To create the persistent class, Right click on **src - New - Class -** specify the class with package name (e.g. com.javatpoint.mypackage) **- finish** .

Employee.java

```
package com.javatpoint.mypackage;
public class Employee {
private int id;
private String firstName,lastName;
public int getId()
{
   return id;
}
public void setId(int id) {
   this.id = id;
}
public String getFirstName()
{
   return firstName;
}
public void setFirstName(String firstName)
{
   this.firstName = firstName;
}
public String getLastName()
{
   return lastName;
}
public void setLastName(String lastName)
 {
   this.lastName = lastName;
}
}
```

4) Create the mapping file for Persistent class

Here, we are creating the same mapping file as created in the previous topic. To create the mapping file, Right click on **src** - **new** - **file** - specify the file name (e.g. employee.hbm.xml) **- ok**. It must be outside the package.

employee.hbm.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-mapping PUBLIC
 "-//Hibernate/Hibernate Mapping DTD 5.3//EN"
 "http://hibernate.sourceforge.net/hibernate-mapping-5.3.dtd">
 <hibernate-mapping>
  <class name="com.javatpoint.mypackage.Employee" table="emp1000">
```

```
    <id name="id">
     <generator class="assigned"></generator>
    </id>
    <property name="firstName"></property>
    <property name="lastName"></property>
  </class>
 </hibernate-mapping>
```

5**) Create the Configuration file**

The configuration file contains all the information's for the database such as connection_url, driver_class, username, password etc. The hbm2ddl.auto property is used to create the table in the database automatically. We will have in-depth learning about Dialect class in next topics. To create the configuration file, right click on src - new - file. Now specify the configuration file name e.g. hibernate.cfg.xml.

hibernate.cfg.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<!DOCTYPE hibernate-configuration PUBLIC
          "-//Hibernate/Hibernate Configuration DTD 5.3//EN"
          "http://hibernate.sourceforge.net/hibernate-configuration-5.3.dtd">

<hibernate-configuration>

    <session-factory>
        <property name="hbm2ddl.auto">update</property>
        <property name="dialect">org.hibernate.dialect.Oracle9Dialect</property>
        <property name="connection.url">jdbc:oracle:thin:@localhost:1521:xe</property>
        <property name="connection.username">system</property>
        <property name="connection.password">oracle</property>
        <property name="connection.driver_class">oracle.jdbc.driver.OracleDriver</proper
ty>
    <mapping resource="employee.hbm.xml"/>
    </session-factory>
  </hibernate-configuration>
```

6) Create the class that retrieves or stores the persistent object

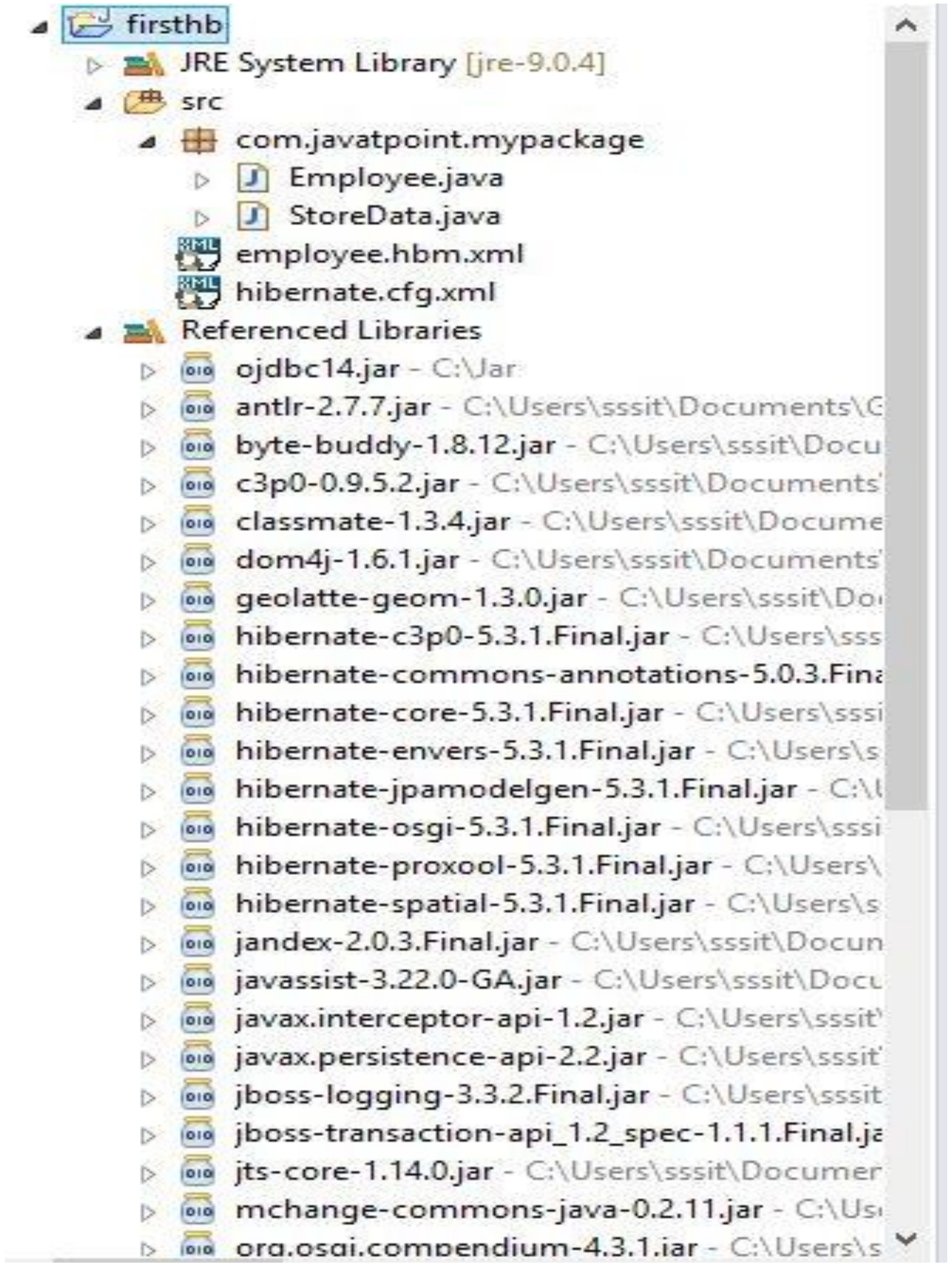
In this class, we are simply storing the employee object to the database.

```
package com.javatpoint.mypackage;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.hibernate.boot.Metadata;
import org.hibernate.boot.MetadataSources;
import org.hibernate.boot.registry.StandardServiceRegistry;
import org.hibernate.boot.registry.StandardServiceRegistryBuilder;
public class StoreData {
public static void main( String[] args )
{
StandardServiceRegistry ssr = new StandardServiceRegistryBuilder().configure("hiberna
te.cfg.xml").build();
Metadata meta = new MetadataSources(ssr).getMetadataBuilder().build();
SessionFactory factory = meta.getSessionFactoryBuilder().build();
Session session = factory.openSession();
Transaction t = session.beginTransaction();
Employee e1=new Employee();
e1.setId(1);
e1.setFirstName("Gaurav");
e1.setLastName("Chawla");
session.save(e1);
t.commit();
System.out.println("successfully saved");
factory.close();
session.close();
 }
}
```

7) Run the application

Before running the application, determine that directory structure is like this.

To run the hibernate application, right click on the StoreData class - Run As - Java Application.



```
firsthb
  JRE System Library [jre-9.0.4]
  src
    com.javatpoint.mypackage
      Employee.java
      StoreData.java
    employee.hbm.xml
    hibernate.cfg.xml
  Referenced Libraries
    ojdbc14.jar - C:\Jar
    antlr-2.7.7.jar - C:\Users\sssit\Documents\G
    byte-buddy-1.8.12.jar - C:\Users\sssit\Docu
    c3p0-0.9.5.2.jar - C:\Users\sssit\Documents
    classmate-1.3.4.jar - C:\Users\sssit\Docume
    dom4j-1.6.1.jar - C:\Users\sssit\Documents
    geolatte-geom-1.3.0.jar - C:\Users\sssit\Do
    hibernate-c3p0-5.3.1.Final.jar - C:\Users\sss
    hibernate-commons-annotations-5.0.3.Fina
    hibernate-core-5.3.1.Final.jar - C:\Users\sssi
    hibernate-envers-5.3.1.Final.jar - C:\Users\s
    hibernate-jpamodelgen-5.3.1.Final.jar - C:\U
    hibernate-osgi-5.3.1.Final.jar - C:\Users\sssi
    hibernate-proxool-5.3.1.Final.jar - C:\Users\
    hibernate-spatial-5.3.1.Final.jar - C:\Users\s
    jandex-2.0.3.Final.jar - C:\Users\sssit\Docun
    javassist-3.22.0-GA.jar - C:\Users\sssit\Docu
    javax.interceptor-api-1.2.jar - C:\Users\sssit
    javax.persistence-api-2.2.jar - C:\Users\sssit
    jboss-logging-3.3.2.Final.jar - C:\Users\sssit
    jboss-transaction-api_1.2_spec-1.1.1.Final.ja
    jts-core-1.14.0.jar - C:\Users\sssit\Documer
    mchange-commons-java-0.2.11.jar - C:\Us
    org.osgi.compendium-4.3.1.jar - C:\Users\s
```

To run the hibernate application, right click on the StoreData class - Run As - Java Application.

Set A

a) Create a Spring core example to display the message “DYPIAN”.

b) Write a program to display the Current Date using spring.

c). Write a Hibernate application to display “Hello” message.

Set B

a) Design a Spring application for accepting student information like Student_id, Student_Name and Student_Age.

b) Design an Employee login form application using spring form MVC validation.

Assignment Evaluation:

0: Not Done [ ]	1: Incomplete [ ]	2: Late Complete [ ]
3: Needs Improvement [ ]	4: Complete [ ]	5: WellDone [ ]

Signature of Instructor

Section II: Android

Assignment 1: Introduction to Android

Objectives

- Study Android Studio installation.
- Create basic android application.

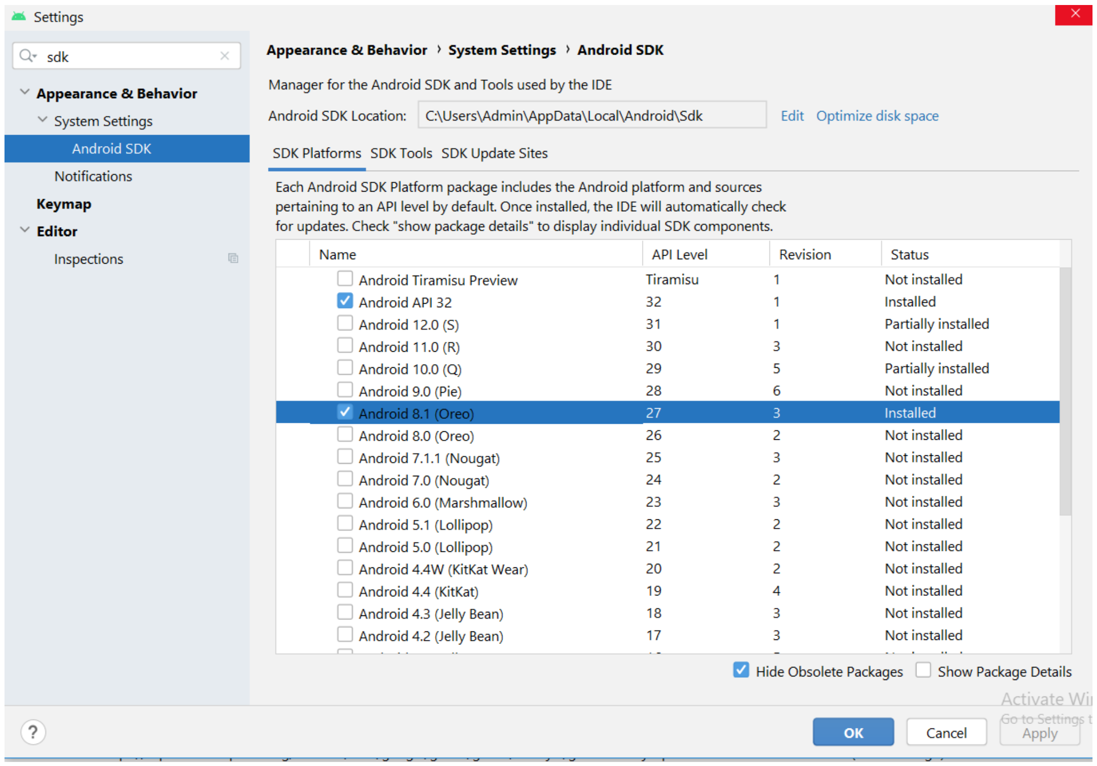
Reading

You should read the following topics before starting this exercise:

1. Installing Android Studio
2. Create "Hello World" application
3. Explore the project structure
4. The Gradle build system
5. Create a virtual device (emulator)
6. Run your app on an emulator
7. Android Architecture

1. Installing Android Studio

1) Install Java JDK: Get the latest version. The JDK may be downloaded here https://www.oracle.com/java/technologies/downloads/#java8
2) Install Android Studio bundle. Use the latest stable version. Android Studio may be downloaded here: https://developer.android.com/studio
3) Follow the prompts to complete the installation. I used the default settings.
4) Allow Android Studio access to the network.
5) Select your desired UI theme.
6) Android Studio will download additional components. This will take several minutes.
7) Select ―Configure/SDK Manager.



Appearance & Behavior › System Settings › Android SDK

Manager for the Android SDK and Tools used by the IDE

Android SDK Location: C:\Users\Admin\AppData\Local\Android\Sdk Edit Optimize disk space

SDK Platforms SDK Tools SDK Update Sites

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, the IDE will automatically check for updates. Check "show package details" to display individual SDK components.

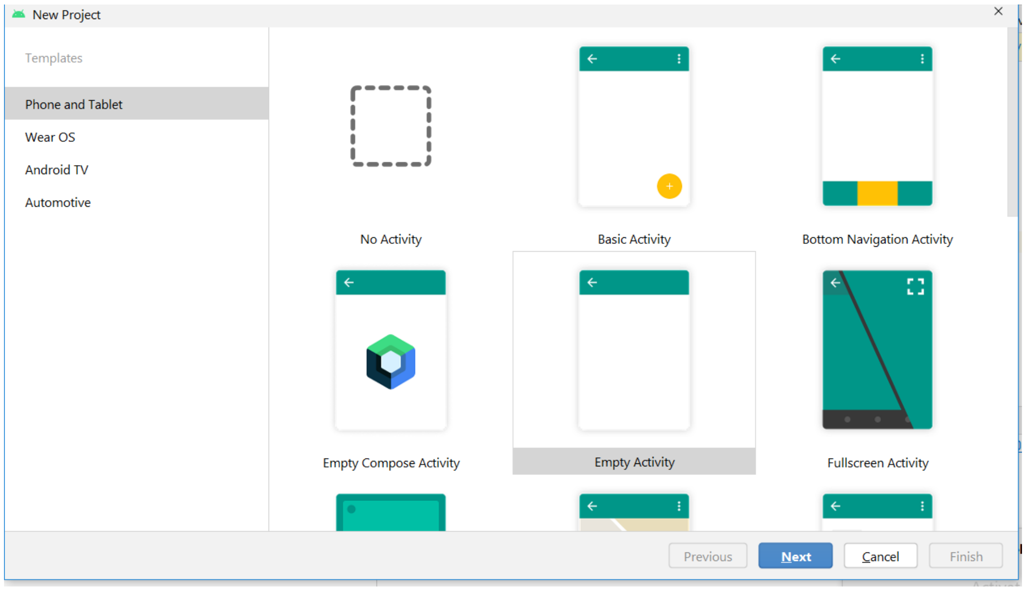
	Name	API Level	Revision	Status
☐	Android Tiramisu Preview	Tiramisu	1	Not installed
✓	Android API 32	32	1	Installed
☐	Android 12.0 (S)	31	1	Partially installed
☐	Android 11.0 (R)	30	3	Not installed
☐	Android 10.0 (Q)	29	5	Partially installed
☐	Android 9.0 (Pie)	28	6	Not installed
✓	Android 8.1 (Oreo)	27	3	Installed
☐	Android 8.0 (Oreo)	26	2	Not installed
☐	Android 7.1.1 (Nougat)	25	3	Not installed
☐	Android 7.0 (Nougat)	24	2	Not installed
☐	Android 6.0 (Marshmallow)	23	3	Not installed
☐	Android 5.1 (Lollipop)	22	2	Not installed
☐	Android 5.0 (Lollipop)	21	2	Not installed
☐	Android 4.4W (KitKat Wear)	20	2	Not installed
☐	Android 4.4 (KitKat)	19	4	Not installed
☐	Android 4.3 (Jelly Bean)	18	3	Not installed
☐	Android 4.2 (Jelly Bean)	17	3	Not installed

✓ Hide Obsolete Packages ☐ Show Package Details

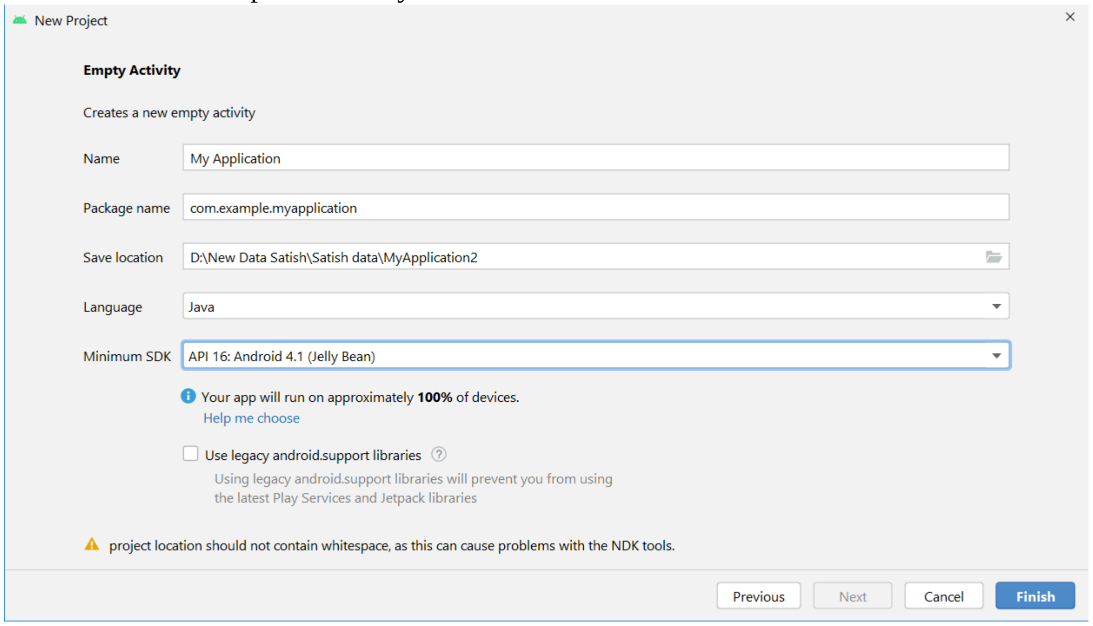
8) Deselect All. Scroll down and select ―Android 8.1 (API 27) Install the packages.

2. Create "Hello World" application

1. In the **Welcome to Android Studio** window, click **Create New Project**. (If you have a project already opened, select **File** > **New** > **New Project**.)
2. In the Select a Project Template window, select **Empty Activity** and click **Next.**



3. In the Configure your project window, complete the following:
   a. Enter "MyFirstApp" in the Name field.
   b. Enter "com.example.myfirstapp" in the Package name field.
   c. If you'd like to place the project in a different folder, change its Save location.
   d. Select either Java or Kotlin from the Language drop-down menu.
   e. Select the lowest version of Android you want your app to support in the Minimum SDK field.
   f. Leave the other options as they are.



New Project

Empty Activity

Creates a new empty activity

Name: My Application

Package name: com.example.myapplication

Save location: D:\New Data Satish\Satish data\MyApplication2

Language: Java

Minimum SDK: API 16: Android 4.1 (Jelly Bean)

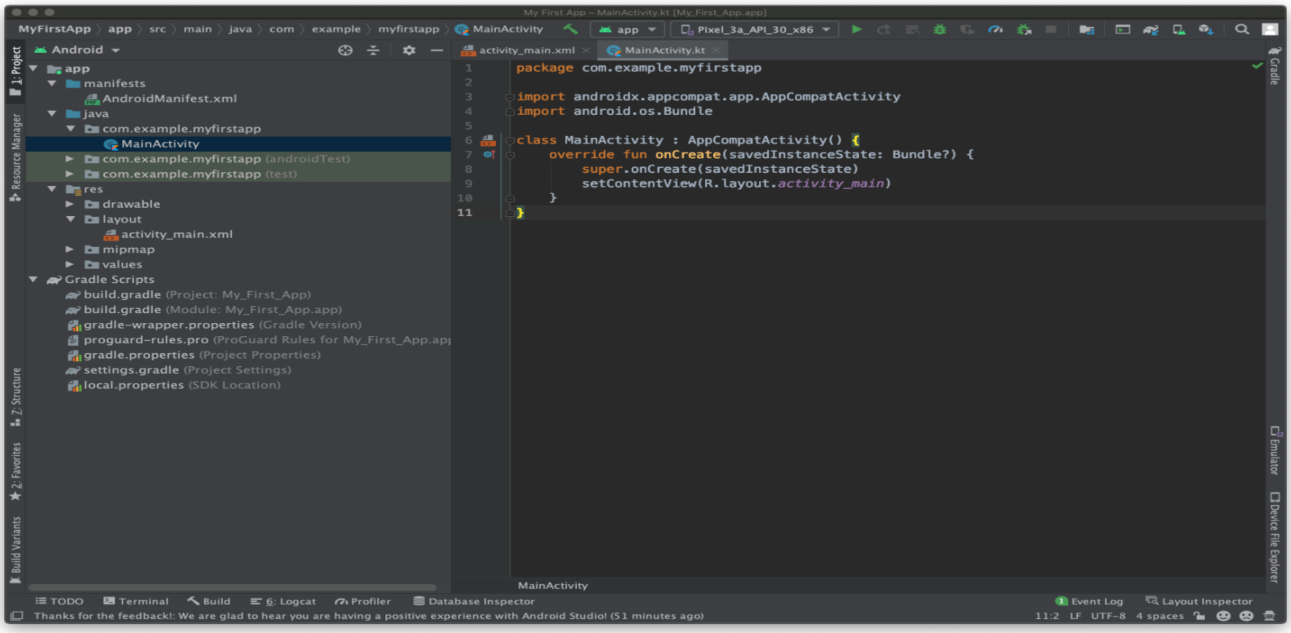
Your app will run on approximately **100%** of devices.
Help me choose

Use legacy android.support libraries
Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

project location should not contain whitespace, as this can cause problems with the NDK tools.

4. Click Finish.

5. After some processing time, the Android Studio main window appears.



```
package com.example.myfirstapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }
}
```

3. Explore the project structure

In the Project > Android view of your previous task, there are three top-level folders below your app folder: manifests, java, and res.

1. Expand the manifests folder.

This folder contains **AndroidManifest.xml**. This file describes all of the components of your Android app and is read by the Android run-time system when your program is executed.

2. **Expand the java folder.**

All your Java language files are organized in this folder. The java folder contains three subfolders:

- **com.example.hello.helloworld** (or the domain name you have specified): All the files for a package are in a folder named after the package. For your Hello World application, there is one package and it only contains MainActivity.java (the file extension may be omitted in the Project view).
- **com.example.hello.helloworld(androidTest)**: This folder is for your instrumented tests, and starts out with a skeleton test file.
- **com.example.hello.helloworld(test)**: This folder is for your unit tests and starts out with an automatically created skeleton unit test file.

3. **Expand the res folder**.

This folder contains all the resources for your app, including images, layout files, strings, icons, and styling. It includes these subfolders:

- **drawable**: Store all your app's images in this folder.
- **layout**: Every activity has at least one layout file that describes the UI in XML. For Hello World, this folder contains **activity_main.xml**.
- **mipmap**: Store your launcher icons in this folder. There is a sub-folder for each supported screen density. Android uses the screen density, that is, the number of pixels per inch to determine the required image resolution. Android groups all actual screen densities into generalized densities, such as medium (mdpi), high (hdpi), or extra-extra-extra-high (xxxhdpi). The ic_launcher.png folder contains the default launcher icons for all the densities supported by your app.
- **values**: Instead of hard coding values like strings, dimensions, and colors in your XML and Java files, it is best practice to define them in their respective values file. This makes it easier to change and be

consistent across your app. Expand the values subfolder within the res folder. It includes these subfolders:

- **colors.xml**: Shows the default colors for your chosen theme, and you can add your own colors or change them based on your app's requirements.
- **dimens.xml**: Store the sizes of views and objects for different resolutions.
- **strings.xml**: Create resources for all your strings. This makes it easy to translate them to other languages.
- **styles.xml**: All the styles for your app and theme go here. Styles help give your app a consistent look for all UI elements.

4. The Gradle build system:

Android Studio uses Gradle as its build system. As you progress through these practicals, you will learn more about gradle and what you need to build and run your apps.

1. Expand the Gradle Scripts folder. This folder contains all the files needed by the build system.
2. Look for the build.gradle(Module:app) file. When you are adding app-specific dependencies, such as using additional libraries, they go into this file

5. Create a virtual device (emulator)

In this task, you will use the Android Virtual Device (AVD) manager to create a virtual device or emulator that simulates the configuration for a particular type of Android device. Using the AVD Manager, you define the hardware characteristics of a device and its API level, and save it as a virtual device configuration. When you start the Android emulator, it reads a specified configuration and creates an emulated device that behaves exactly like a physical version of that device, but it resides on your computer. With virtual devices, you can test your apps on different devices (tablets, phones) with different API levels to make sure it looks good and works for most users. You do not need to depend on having a physical device available for app development. In order to run an emulator on your computer, you have to create a configuration that describes the virtual device.

- In Android Studio, select Tools > AVD Manager, or click the AVD Manager icon in the toolbar.
- Click the **+Create Virtual Device….** (If you have created a virtual device before, the window shows all of your existing devices and the button is at the bottom.) The Select Hardware screen appears showing a list of preconfigured hardware devices. For each device, the table shows its diagonal display size (Size), screen resolution in pixels (Resolution), and pixel density (Density).
- Choose which version of the Android system to run on the virtual device. You can select the latest system image. There are many more versions available than shown in the recommended tab.
- If a Download link is visible next to a system image version, it is not installed yet, and you need to download it. If necessary, click the link to start the download, and click Finish when it's done.
- On System Image screen, choose a system image and click Next.
- Verify your configuration, and click Finish. (If the Your Android Devices AVD Manager window stays open, you can go ahead and close it.)

6. Run your app on an emulator

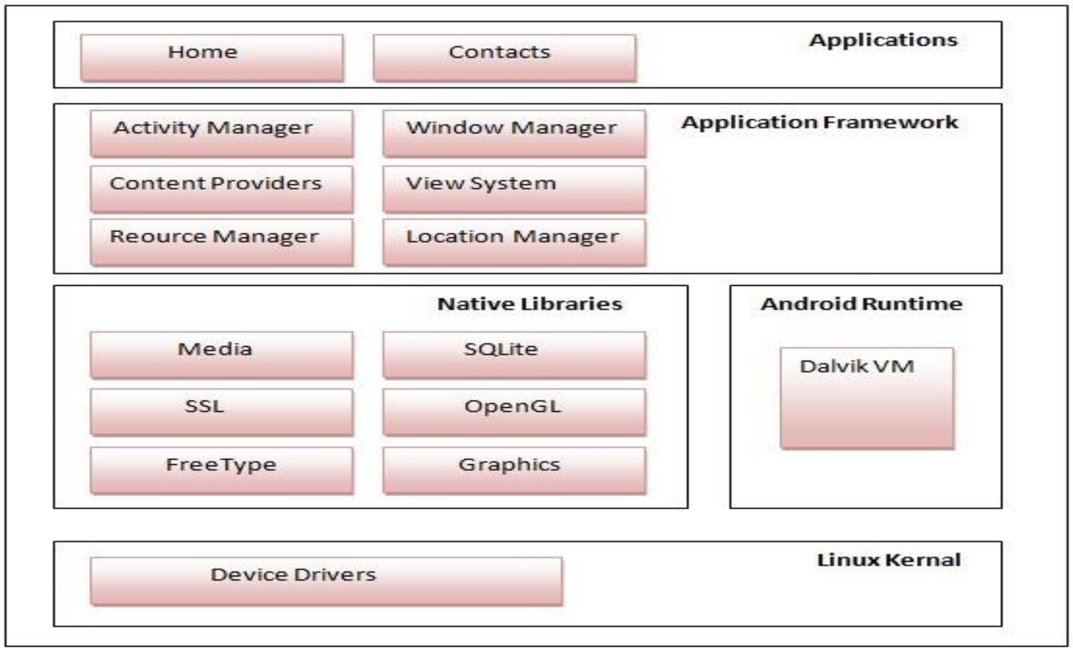
1) In Android Studio, select Run > Run app or click the Run icon in the toolbar.
2) In the Select Deployment Target window, under Available Emulators, select Pixel API 27 and click OK
3) You should see the Hello World app as shown in the following screenshot.

7. Android Architecture

Android architecture or Android software stack is categorized into five parts:

1. Linux kernel

2. Native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications



1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. **Linux kernel** is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of Linux kernel, there are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc. The WebKit library is responsible for browser support. SQLite is for database. FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes **Android API's** such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using Linux kernel.

Assignment 2 : Layout, Activity and Intent

Objectives

- To study how to use Activities, Layouts and Intents in the application.

- To study different layout in an Android.
- To study how to link Activities and interaction between Intent.

Reading

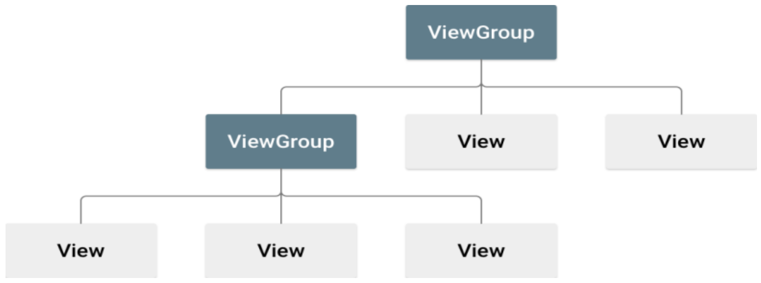
You should read the following topics before starting this exercise:

- Android - UI Layouts
- Activity, Activity Lifecycle
- Linking Activities using Intent.

Ready Reference

1) Android - UI Layouts

A layout defines the structure for a user interface in our app, such as in an activity. All elements in the layout are built using a hierarchy of View and ViewGroup objects. A View usually draws something the user can see and interact with. Whereas a View Group is an invisible container that defines the layout structure for View and other ViewGroup objects.



The View objects are usually called "widgets" and can be one of many subclasses, such as Button or TextView. The ViewGroup objects are usually called "layouts" can be one of many types that provide a different layout structure, such as LinearLayout or ConstraintLayout . There are number of Layouts provided by Android which we will use in almost all the Android applications to provide different view, look and feel and they are...

LinearLayout: Android LinearLayout is a view group that aligns all children in a single direction either vertically or horizontally.	<?xml version="1.0" encoding="utf-8"?> <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent" android:orientation="vertical"> <TextView android:id="@+id/text" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="This is a TextView"/> <Button android:id="@+id/button" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="This is a Button"/> <EditText android:id="@+id/edtPainText" android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="This is a EditText"/> <!-- More GUI components go here --> </LinearLayout>
RelativeLayout:	<?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent" android:layout_height="match_parent"

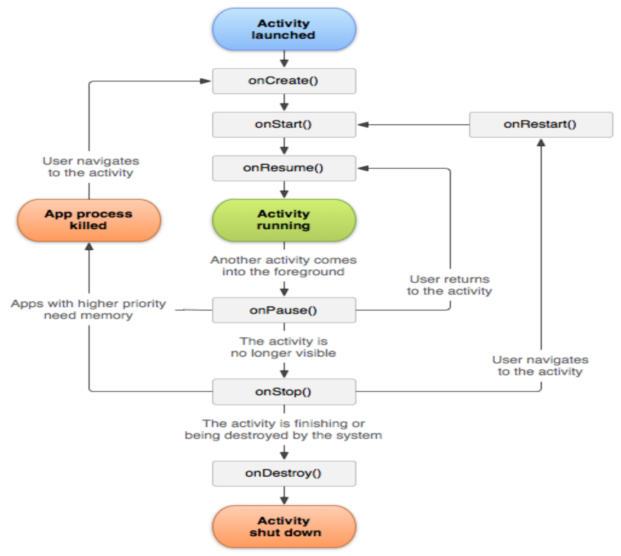
Android RelativeLayout enables you to specify how child views are positioned relative to each other. The position of each view can be specified as relative to sibling elements or relative to the parent. Using RelativeLayout, you can align two elements by right border, or make one below another, centered in the screen, centered left, and so on. By default, all child views are drawn at the top-left of the layout.	android:paddingLeft="16dp" android:paddingRight="16dp" > <EditText android:id="@+id/name" android:layout_width="match_parent" android:layout_height="wrap_content" android:hint="@string/reminder" /> <Spinner android:id="@+id/dates" android:layout_width="0dp" android:layout_height="wrap_content" **android:layout_below="@id/name"** **android:layout_alignParentLeft="true"** **android:layout_toLeftOf="@+id/times"** /> <Spinner android:id="@id/times" android:layout_width="96dp" android:layout_height="wrap_content" **android:layout_below="@id/name"** **android:layout_alignParentRight="true"** /> <Button android:layout_width="96dp" android:layout_height="wrap_content" **android:layout_below="@id/times"** **android:layout_alignParentRight="true"** android:text="@string/done" /> </RelativeLayout>
TableLayout : Android TableLayout going to be arranged groups of views into rows and columns. You will use the <TableRow> element to build a row in the table. Each row has zero or more cells; each cell can hold one View object, like ImageView, TextView or any other view. For building a row in a table you will use the <TableRow> element. Table row objects are the child views of a table layout. Total width of a table is defined by its parent container. Column can be both stretchable and shrinkable. If shrinkable then the width of column can be shrunk to fit the table into its parent object and if stretchable then it can expand in width to fit any extra space available.	<TableLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent"> <TableRow android:layout_width="fill_parent" android:layout_height="fill_parent"> <TextView android:text="First Name" android:layout_width="wrap_content" android:layout_height="wrap_content" android:layout_column="1"/> <EditText android:width="200px" android:layout_width="wrap_content" android:layout_height="wrap_content"/> </TableRow> <TableRow android:layout_width="fill_parent" android:layout_height="fill_parent"> <Button android:layout_width="wrap_content" android:layout_height="wrap_content" android:text="Submit" android:id="@+id/button" android:layout_column="2"/> </TableRow> </TableLayout>
AbsoluteLayout: In Android, an Absolute Layout is a layout used to design the custom layouts. An Absolute Layout	<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="fill_parent" android:layout_height="fill_parent"> <Button android:layout_width="100dp"

lets you specify exact locations (x/y coordinates) of its children. Absolute layouts are less flexible and harder to maintain than other types of layouts without absolute positioning	android:layout_height="wrap_content" android:text="OK" android:layout_x="50px" android:layout_y="361px"/> <Button android:layout_width="100dp" android:layout_height="wrap_content" android:text="Cancel" **android:layout_x="225px" android:layout_y="361px"/>** </AbsoluteLayout>
FrameLayout : Frame Layout is designed to block out an area on the screen to display a single item. Generally, FrameLayout should be used to hold a single child view, because it can be difficult to organize child views in a way that's scalable to different screen sizes without the children overlapping each other. You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the android:layout_gravity attribute.	**ConstraintLayout :** Android Constraint Layout is a ViewGroup (i.e. a view that holds other views) which allows you to create large and complex layouts with a flat view hierarchy, and also allows you to position and size widgets in a very flexible way. It was created to help reduce the nesting of views and also improve the performance of layout files.
ListView : List of scrollable items can be displayed in Android using ListView. It helps you to displaying the data in the form of a scrollable list. Users can then select any list item by clicking on it. ListView is default scrollable so you do not need to use scroll View or anything else with ListView. A very common example of ListView is phone contact book	**GridView :** In android GridView is a view group that display items in two or more dimensional scrolling grid (rows and columns), the grid items are not necessarily predetermined but they are automatically inserted to the layout using a ListAdapter. Users can then select any grid item by clicking on it. GridView is default scrollable so you don't need to use ScrollView or anything else with GridView. An example of GridView is your default Gallery
ScrollView : In Android, a ScrollView is a view group that is used to make vertically scrollable views. A scroll view contains a single direct child only. In order to place multiple views in the scroll view, one needs to make a view group(like LinearLayout) as a direct child and then you can define many views inside it. A ScrollView supports Vertical scrolling only, so in order to create a horizontally scrollable view, HorizontalScrollView is used.	<?xml version="1.0" encoding="utf-8"?> <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android" android:layout_width="match_parent" android:layout_height="match_parent" tools:context=".MainActivity"> <ScrollView android:layout_width="match_parent" android:layout_height="match_parent" tools:layout_editor_absoluteX="0dp" tools:layout_editor_absoluteY="-127dp"> <TextView android:id="@+id/scrolltext" android:layout_width="match_parent" android:layout_height="wrap_content" android:text="@string/scrolltext" android:textColor="@color/green"/> </ScrollView> </androidx.constraintlayout.widget.ConstraintLayout>

2) Activity, Activity Lifecycle

Activity : An activity represents a single screen with a user interface just like window or frame of Java. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View).

Activity Lifecycle : Activities in the system are managed as an activity stack. When a new activity is started, it is placed on the top of the stack and becomes the running activity - the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits.

Activity Lifecycle Public Methods , Description & Structure	
1) onCreate ( ) : called when activity is first created. This is where you should do all of your normal static set up: create views, bind data to lists, ect.	 Activity launched onCreate() onStart() onRestart() User navigates to the activity onResume() App process killed Activity running Another activity comes into the foreground User returns to the activity Apps with higher priority need memory onPause() The activity is no longer visible User navigates to the activity onStop() The activity is finishing or being destroyed by the system onDestroy() Activity shut down
2) onStart ( ) : called when activity is becoming visible to the user.	
3) onResume ( ): called when activity will start interacting with the user.	
4) onPause ( ) : called when activity is not visible to the user.	
5) onStop ( ) : called when activity is no longer visible to the user.	
6) onRestart( ) : called after your activity is stopped, prior to start.	
7) onDestroy ( ) : called before the activity is destroyed	

3) Linking Activities using Intent.

Intent is the objects, which is used in android for passing the information among Activities in an Application and from one app to another also.

For example: Intent facilitates you to redirect your activity to another activity on occurrence of any event. By calling, startActivity() you can perform this task.

Intent intent = new Intent (getApplicationContext ( ), SecondActivity.class);
startActivity (intent);

In the above example, foreground activity is getting redirected to another activity i.e. SecondActivity.java.

getApplicationContext() returns the context for your foreground activity.

Intent are of two types: **Explicit Intent and Implicit Intent**

Explicit Intent: Explicit Intents are used to connect the application internally.In Explicit we use the name of component which will be affected by Intent. For Example: If we know class name then we can navigate the app from One Activity to another activity using Intent. Explicit Intent work internally within an application to perform navigation and data transfer.

Intent intent = new Intent(getApplicationContext(), SecondActivity.class);
startActivity(intent);

Here SecondActivity is the JAVA class name where the activity will now be navigated.

Implicit Intent: In Implicit Intents we do need to specify the name of the component. We just specify the Action which has to be performed and further this action is handled by the component of another application. The basic example of implicit Intent is to open any web page

Intent intentObj = new Intent(Intent.ACTION_VIEW);
intentObj.setData(Uri.parse("https://www.google.com"));
startActivity(intentObj);

Unlike Explicit Intent you do not use any class name to pass through Intent(). In this example you have just specified an action. Now when we will run this code then Android will automatically start your web browser and it will open google home page.

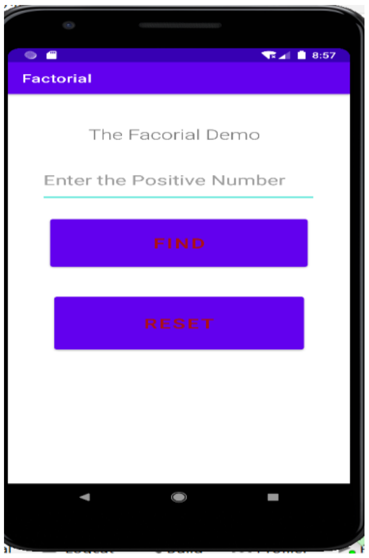
Example :

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="404dp"
android:layout_height="36dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="52dp"
        android:fontFamily="sans-serif-light"
android:gravity="center"
        android:text="The Facorial Demo"       android:textSize="24sp"
        app:layout_constraintBottom_toTopOf="@+id/edNum"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <EditText
        android:id="@+id/edNum"
        android:layout_width="318dp"       android:layout_height="76dp"
        android:layout_marginTop="24dp"       android:ems="10"
        android:hint="Enter the Positive Number"
        android:inputType="textPersonName"
        android:textColor="#C50F0F"       android:textSize="24sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.494"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />
    <Button
        android:id="@+id/btFind"
        android:layout_width="296dp"       android:layout_height="95dp"
        android:layout_marginBottom="348dp"       android:text="Find"
        android:textColor="#AD1111"       android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/edNum" />
    <Button
        android:id="@+id/btReset"
        android:layout_width="287dp"
android:layout_height="104dp"
        android:text="Reset"
        android:textColor="#AD1111"       android:textSize="24sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btFind"
        app:layout_constraintVertical_bias="0.151" />
    <TextView
        android:id="@+id/txtDisplay"
        android:layout_width="269dp"       android:layout_height="78dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btReset"
        app:layout_constraintVertical_bias="0.245" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.java

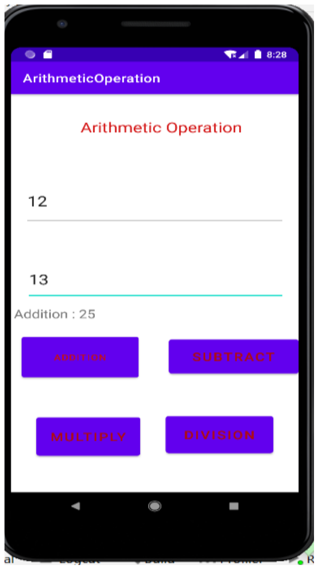
```
package com.example.factorial;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
public class MainActivity extends AppCompatActivity {
    EditText edNum;
    Button btFind,btReset;
    TextView txtDisplay;
   @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        init();
        btFind.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                int num = Integer.parseInt(edNum.getText().toString());
                int f = 1;
                for(int i = 1;i<=num;i++)
                    f = f * i;
                 txtDisplay.setText("Facorial : "+f);
            }
        });
    }
    public void init()
    {
        edNum = findViewById(R.id.edNum);
        btFind = findViewById(R.id.btFind);
        btReset = findViewById(R.id.btReset);
       txtDisplay = findViewById(R.id.txtDisplay);
    }
}
```



Lab Assignments:

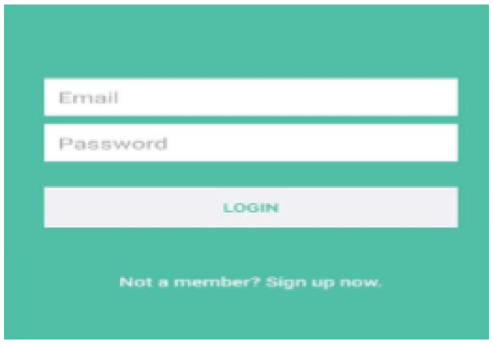
Set A

1. Create a Simple Application Which Shows Life Cycle of Activity.
2. Create a Simple Application Which Send ―Hello‖ message from one activity to another with help of Button (Use Intent).
3. Create a Simple Application, which read a positive number from the user and display its factorial value in another activity.
4. Create a Simple Application, that performs Arithmetic Operations. (Use constraint layout)

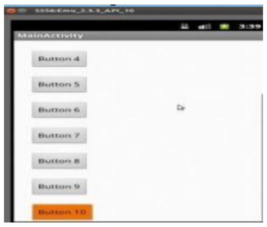


Set B

1. Create an Android App, Which reads the Students Details (Name, Surname, Class, Gender, Hobbies, Marks) and Display the all information in another activity in table format on click of Submit button.
2. Create an Android App with Login Screen. On successful login, gives message go to next Activity (Without Using Database & use Table Layout).

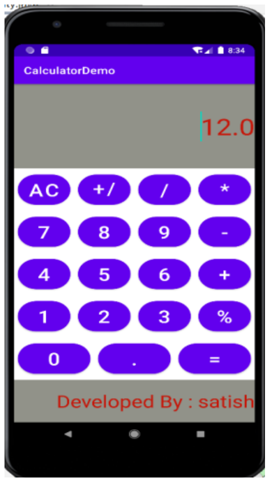


3. Create following Vertical Scroll View Creation in Android.

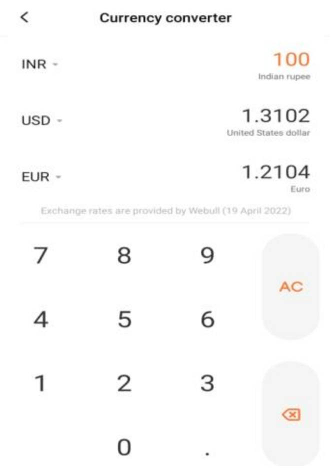


Set C

1. Create a Simple calculator. (Use Linear Layout)



2) Create an Android Application to convert Indian Rupee(IND) to USD & EUR.



Signature of the instructor: ------------------------- Date:------------------------

Assignment Evaluation

0: Not Done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs Improvement		5: Well Done	

Assignment 3 : Android User Interface and Event Handling

Objectives

- Study how to create user interface in Android.
- Study how to perform event handling

Reading

You should read the following topics before starting this exercise:

- Android - UI Layouts
- Activity, Activity Lifecycle

Control Description	**Xml code**	**Java Code**

Android - TextView Control:
 	
In Android, TextView displays text to the user and optionally allows them to edit it programmatically.
	<TextView
	android:id="@+id/textView"
	android:layout_width="wrap_content"
	android:layout_height="wrap_content"
	android:text="Dr. D Y Patil College"
	android:textSize="20sp"
	android:gravity="center_horizontal"
	android:textColor="#f00"
	android:textStyle="bold|italic" />
		TextView textView = findViewById(R.id.textView);
		textView.setText(" Satish Mulgi "); //set text for text view
		textView.setTextColor(Color.***RED***); //set red color for text view
		textView.setTextSize(20); //set 20sp size of text
		//set background color
		textView.setBackgroundColor(Color.***BLACK);***

Android - EditText Control :
In Android, EditText is a standard entry widget in android apps. It is an overlay over TextView that configures itself to be editable. **We often use EditText in our applications in order to provide an input or text field**
	<EditText
	android:id="@+id/simpleEditText"
	android:layout_width="fill_parent"
	android:layout_height="wrap_content"
	android:layout_centerInParent="true"
	android:hint="Enter Your Name Here"
	android:textColorHint="#0f0" />
		EditText editText = findViewById(R.id.simpleEditText);
		editText.setHint("Enter Your Name Here");//display the hint
		//set the green hint color
		simpleEditText.setHintTextColor(Color.*green*(0));
		String editTextValue = editText.getText().toString();
		editText.setText(" Satish Mulgi "); //set text for text view
		editText.setTextSize(20); //set 20sp size of text

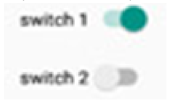
Android - Button Control :
In Android, **Button** represents a push button. A Push buttons can be clicked, or pressed by the user to perform an action.
	<Button
	android:id="@+id/simpleButton"
	android:layout_width="wrap_content"
	android:layout_height="wrap_content"
	android:gravity="right|center_vertical"
	android:textColor="#f00"
	android:textSize="25sp"
	android:textStyle="bold|italic"
	android:background="#147D03"
	android:text="ClickMe"/>
		simpleButton = findViewById(R.id.simpleButton);
		//get id of button
		simpleButton.setOnClickListener(new View.OnClickListener() {
		    @Override
		    public void onClick(View view) {
		      Toast.makeText(getApplicationContext(),
		        "Simple Button ", Toast.LENGTH_LONG).show();
		//display the text of button
		    }
		  });

Android - ImageView Control Android ImageButton Control
In Android, ImageView class is used to display an image file in application.
In Android, ImageButton is used to display a normal button with a custom image in a button
	<ImageView
	android:id="@+id/img"
	android:layout_width="fill_parent"
	android:layout_height="wrap_content"
	android:src="@drawable/lion" />
	<ImageButton
	android:id="@+id/img1"
	android:layout_width="wrap_content"
	android:layout_height="wrap_content"
	android:src="@drawable/home"
	android:background="#000"/>
		ImageView img = findViewById(R.id.img);
		//get the id of second image view
		  img.setOnClickListener(new View.OnClickListener() {
		    @Override
		    public void onClick(View view) {
		  Toast.makeText(getApplicationContext(),
		      "Lion", Toast.LENGTH_LONG).show();
		//display the text on image click event
		    }
		  });

Android - CheckBox Control :
In Android, CheckBox is a type of two state button either unchecked or checked in Android. Or you can say it is a type of on/off switch that can be toggled by the users.
	<CheckBox android:id="@+id/cb"
	android:layout_width="wrap_content"
	android:layout_height="wrap_content"
	android:text="Cricket"
	android:textColor="#44f"
	android:textSize="20sp"/>
		CheckBox cb = findViewById(R.id.*cb*);
		// set the current state of a check box
		cb.setChecked(true);
		//check current state of a check box (true or false)
		Boolean checkBoxState = cb.isChecked();

Android - Switch (On/Off) :

In Android, Switch is a two-state toggle, switch widget that can select between two options. It is used to display checked and unchecked state of a button providing slider control to user. It is basically an off/on button which indicate the current state of Switch. It is commonly used in selecting on/off in Sound, Bluetooth, WiFi etc.



```
<Switch
   android:id="@+id/simpleSwitch"
  android:layout_width="wrap_content"
 android:layout_height="wrap_content"
   android:checked="true"
   android:text="switch"
android:layout_centerHorizontal="true"
   android:textOn="On"
   android:textOff="Off"
   android:textColor="#f00"
   android:padding="20dp"
   android:gravity="center"
   android:textSize="25sp"
   android:background="#000"/>
```

```
 simpleSwitch = findViewById(R.id.simpleSwitch);
      submit = (Button) findViewById(R.id.submitButton);
      submit.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
              String ss;
              if (simpleSwitch.isChecked())
                  ss = simpleSwitch.getTextOn().toString();
              else
                  ss = simpleSwitch.getTextOff().toString();

Toast.makeText(getApplicationContext(), "Switch :" + ss + "\n", Toast.LENGTH_LONG).show();
```

ToggleButton (On/Off) :

In Android, ToggleButton is used to display checked and unchecked state of a button. ToggleButton basically an off/on button with a light indicator, which indicate the current state of toggle button. The most simple example of ToggleButton is doing on/off in sound, Bluetooth, wifi, hotspot etc.

```
<ToggleButton
 android:id="@+id/tb"
   android:layout_width="wrap_content"
  android:layout_height="wrap_content"
   android:checked="true"
   android:textOff="Off State"
   android:textOn="On State"
   android:textSize="25sp"
 android:layout_centerHorizontal="true"
   android:textColor="#f00"
   android:padding="40dp"/>
```

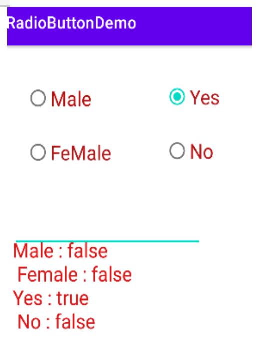
```
 tb = findViewById(R.id.tb);
 submit = (Button) findViewById(R.id.submitButton);
 submit.setOnClickListener(new View.OnClickListener() {
          @Override
          public void onClick(View view) {
              String status = "ToggleButton : " + tb.getText();
        Toast.makeText(getApplicationContext(),
status, Toast.LENGTH_SHORT).show();
// display the current state of toggle button's
          }  });
```

RadioButton & RadioGroup

In Android, RadioButton are mainly used together in a RadioGroup.
In RadioGroup checking the one radio button out of several radio button added in it will automatically unchecked all the others.
It means at one time we can checked only one radio button from a group of radio buttons which belong to same radio group.
RadioButon is a two state button that can be checked or unchecked. If a radio button is unchecked then a user can check it by simply clicking on it. Once a RadiaButton is checked by user it can't be unchecked by simply pressing on the same button.
It will automatically unchecked when you press any other RadioButton within same RadioGroup.

```
<RadioGroup
   android:layout_width="wrap_content" android:layout_height="wrap_content">
<RadioButton
  android:id="@+id/simpleRadioButton"
   android:layout_width="wrap_content"
  android:layout_height="wrap_content"
   android:checked="true"
   android:textSize="25sp"
   android:textStyle="bold|italic"
   android:padding="20dp"
  android:layout_centerHorizontal="true"
   android:text="Male"
   android:textColor="#f00"
   android:background="#000"/>
 <RadioButton
  android:id="@+id/female" android:layout_width="wrap_content" android:layout_height="wrap_content"/>
 </RadioGroup>
```

```
public void showMethod(View view){
male = findViewById(R.id.male);
female = findViewById(R.id.female);
   str = "Male : "+male.isChecked()+"\n Female : "+female.isChecked();
   str = str+"\nYes : "+yes.isChecked()+"\n No : "+no.isChecked();
   textView.setText(str.toString());
   Toast toast =
Toast.makeText(getApplicationContext(),str,Toast.LENGTH_LONG);
   toast.setMargin(200,100);
   toast.show();                }
```



```
Male : false
 Female : false
Yes : true
 No : false
```

Android Toast : Android Toast can be used to display information for the short period of time. Toast class is used to show notification for a particular interval of time. After sometime it disappears. It doesn't block the user interaction. Constants of toast :

public static final int LENGTH_LONG : displays view for the long duration of time.

public static final int LENGTH_SHORT : displays view for the short duration of time.

public static Toast makeText(Context context, CharSequence text, int duration) : makes the toast containing text and duration.

public void show() : displays toast.

public void setMargin (float horizontalMargin, float verticalMargin): changes the horizontal and vertical margin difference.

Example :

```
Toast toast=Toast.makeText(getApplicationContext(),"Hello Friends”, Toast.LENGTH_SHORT);
toast.setMargin(50,50);
toast.show();
```

RatingBar : RatingBar is used to get the rating from the app user. A user can simply touch, drag or click on the stars to set the rating value. The value of rating always returns a floating point number which may be 1.0, 2.5, 4.5 etc.

1) The **getRating()** method of android RatingBar class returns the rating number.

```
RatingBar simpleRatingBar = (RatingBar) findViewById(R.id.simpleRatingBar);  // initiate a rating bar
Float ratingNumber = simpleRatingBar.getRating();          // get rating number from a rating bar
```

2) **numStars**: numStars attribute is used to set the number of stars (or rating items) to be displayed in a rating bar. By default a rating bar shows five stars but we can change it using numStars attribute. numStars must have a integer number like 1,2 etc.

```
<RatingBar
        android:id="@+id/simpleRatingBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:numStars="7" />
```

Java code : simpleRatingBar.setNumStars(7); // set total number of stars

3) **getNumStars():** is used to get the number of stars of a RatingBar.
   int numberOfStars = simpleRatingBar.getNumStars(); // get total number of stars of rating bar

4) **rating:** Rating attribute set the default rating of a rating bar. It must be a floating point number.

```
<RatingBar
        android:id="@+id/simpleRatingBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:rating="3.5" />
```

Java code : simpleRatingBar.setRating((float) 3.5); // set default rating

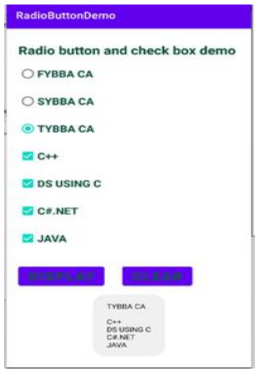
Lab Assignments:

Set A

1. Create an Android Application that will change color of the College Name(Use TextView) on click of Push Button and change the font size, font style of text view using xml.

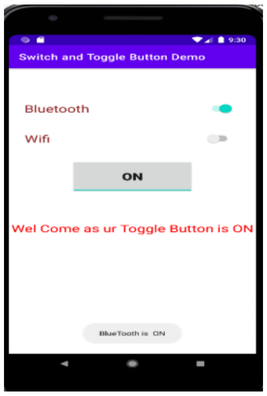
2. Create an Android Application to accept two numbers(Use PainText) and create two buttons (power and Average). Display the result on the next activity on Button click

3. Design Following Screens Using RadioButtons & CheckBoxes.Display the selected text using Toast.



TYBBA CA
C++
DS USING C
C#.NET
JAVA

4. Create an Android Application that Demonstrate Switch and Toggle Button.

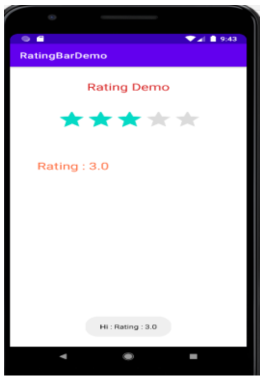


Wel Come as ur Toggle Button is ON

BlueTooth is ON

Set B

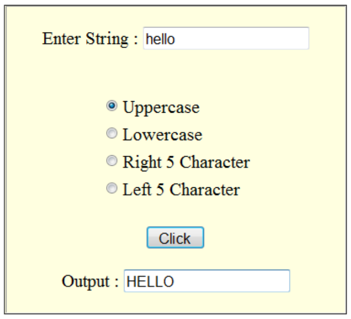
1. Create an Android Application that Demonstrate RatingBar and Display the number of stars selected on Toast and TextView



Rating : 3.0

Hi : Rating : 3.0

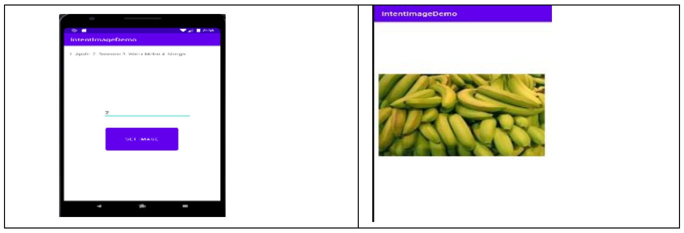
2. Create an Android Application to perform following string operation according to user selection of radio button.


Enter String : hello

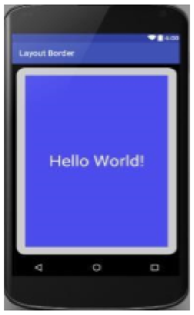
Uppercase [✓]
Lowercase
Right 5 Character
Left 5 Character

Output : HELLO

3. Write an Android Application to Change the Image Displayed on the Screen

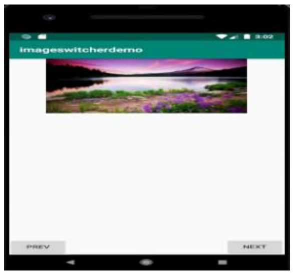


4. Design following-add a border to an Android Layout .

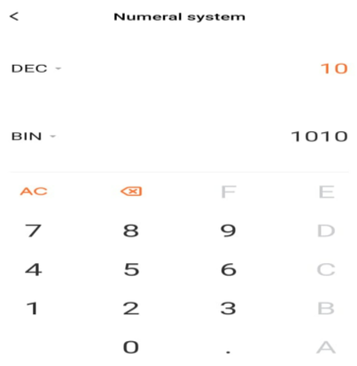


Set C

1. Construct image switcher using setFactory().



2. Create an Android Application to convert Decimal number to Binary equivalent.



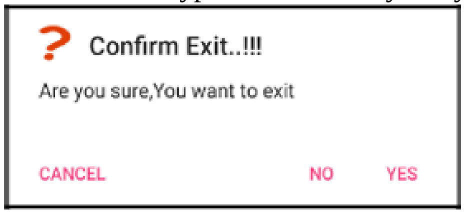
Signature of the instructor: ------------------------ Date:-----------------------

Assignment Evaluation

0: Not Done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs Improvement		5: Well Done	

Assignment 4 : Android TimePicker, DatePicker, Alert Dailog

Android - Alert Dialog: Alert Dialog in an android UI prompts a small window to make decision on mobile screen. Sometimes before making a decision, it is required to give an alert to the user without moving to next activity. For example you have seen this type of alert when you try to exit the App and App ask you to confirm exiting.


Confirm Exit..!!!
Are you sure,You want to exit

AlertDialog.Builder is used to create an interface for Alert Dialog in Android for setting like alert title, message, image, button, button onclick functionality etc.

AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);

1) **setTitle(CharSequence title)** – This component is used to set the title of the alert dialog. It is optional component.

alertDialogBuilder.setTitle("Confirm Exit..!!!"); // Setting Alert Dialog Title

2) **setIcon(Drawable icon)** – This component add icon before the title. You will need to save image in drawable icon.

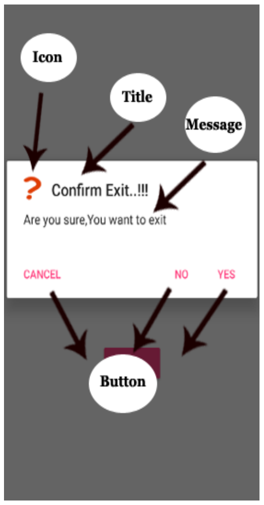
alertDialogBuilder.setIcon(R.drawable.question); // Icon Of Alert Dialog

3) **setMessage(CharSequence message)** – This component displays the required message in the alert dialog.

alertDialogBuilder.setMessage("Are you sure,You want to exit");// Setting Alert Dialog Message

4) **setCancelable(boolean cancelable)** – This component has boolean value i.e true/false. If set to false it allows to cancel the dialog box by clicking on area outside the dialog else it allows.

alertDialogBuilder.setCancelable(false);



5) **setPositiveButton(CharSequence text, DialogInterface.OnClickListener listener)** – This component add positive button and further with this user confirm he wants the alert dialog question to happen.

```
alertDialogBuilder.setPositiveButton("Yes",
        new DialogInterface.OnClickListener() {
  @Override
         public void onClick(DialogInterface arg0, int arg1) {
    finish ();
} });
```

6) **setNegativeButton(CharSequence text, DialogInterface.OnClickListener listener)** – This component add negative button and further with this user confirm he doesn't want the alert dialog question to happen.

```
                        alertDialogBuilder.setNegativeButton("No",
                            new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
Toast.makeText(MainActivity.this,"You clicked over No",Toast.LENGTH_SHORT).show();
                        } });
```

7) **setNeutralButton(CharSequence text, DialogInterface.OnClickListener listener)** – This component simply add a new button and on this button developer can set any other onclick functionality like cancel button on alert dialog.

```
                        alertDialogBuilder.setNeutralButton("Cancel",
                            new DialogInterface.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int which) {
                        Toast.makeText(getApplicationContext(),"You        clicked        on
                            Cancel",Toast.LENGTH_SHORT).show();
                         }
                         });
```

8) alertDialogBuilder.create();
9) alertDialogBuilder.show();

Example :

```
final AlertDialog.Builder adb = new AlertDialog.Builder(this);
adb.setTitle("Confirm Exit !");
adb.setMessage("are you sure, you want to Exit ?");
adb.setCancelable(true);
adb.setPositiveButton("OK", new DialogInterface.OnClickListener() {
   @Override
   public void onClick(DialogInterface dialog, int which) {
      finish();
   }
});
adb.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
   @Override
   public void onClick(DialogInterface dialog, int which) {
      Toast.makeText(getApplicationContext(),"You pressed Cancel  ",Toast.LENGTH_LONG).show();
   }
});
adb.setNeutralButton("Retry", new DialogInterface.OnClickListener() {
   @Override
   public void onClick(DialogInterface dialog, int which) {
      Toast.makeText(getApplicationContext(),"You Presses cancel ",Toast.LENGTH_LONG).show();
   }
});
button.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
         adb.create();
```

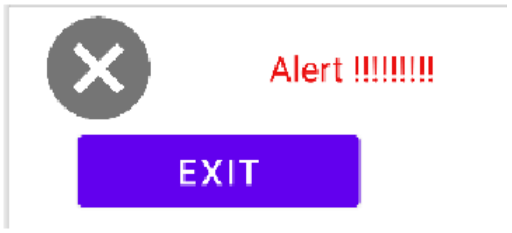
```
            adb.show();
        }
    });
```

Custom Alert Dialog: The custom dialog uses DIALOG to create custom alert in android studio. Dialog display a small window that is a popup which draws the user attention over the activity before they continue moving forward. The following are the steps to create custom alert Dialog.

Step 1: Create a new project and name it **CustomAlertDialogDemo**.

Step 2: Open res -> layout -> activity_main.xml and create XML for button and textview. On button click custom alert dialog will appear & textview is for asking user to click on button.

Step 3 : Now create a layout file name as custom.xml . In this file we are going to define the XML for custom dialog appearance. For example a textbox , imageview and a button.



Step 4 : Now open app -> java -> package -> MainActivity.java and add the below code. In this code onclick is added over the button click and Dialog is used to create the custom alert dialog.

```
button.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View arg0) {
            final Dialog dialog = new Dialog(context); // custom dialog
            dialog.setContentView(R.layout.custom);
            // if button is clicked, close the custom dialog
           dialog.setOnClickListener(new View.OnClickListener() {
               @Override
               public void onClick(View v) {
                   dialog.dismiss();
Toast.makeText(getApplicationContext(),"Dismissed..!!",Toast.LENGTH_SHORT).show();
               }
           });
```

TimePicker: In Android, TimePicker is a widget used for selecting the time of the day in either AM/PM mode or 24 hours mode. The displayed time consist of hours, minutes and clock format. If we need to show this view as a Dialog then we have to use a TimePickerDialog class.

```
<TimePicker
    android:id="@+id/simpleTimePicker"
    android:layout_width="wrap_content"
   android:layout_height="wrap_content"
    android:timePickerMode="spinner"/>  or  clock
```

Some Methods of TimePicker:

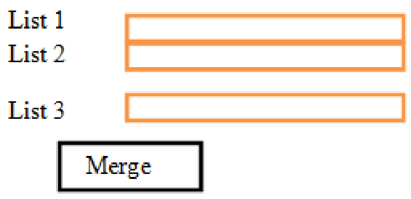
1) **setCurrentHour(Integer currentHour):** This method is used to set the current hours in a time picker.

2) **setHour(Integer hour):** setCurrentHour() method was deprecated in API level 23. From api level 23 we have to use setHour(Integer hour). In this method there is only one parameter of integer type which is used to set the value for hours.
3) **setCurrentMinute(Integer currentMinute):** This method is used to set the current minutes in a time picker.
4) **getCurrentHour():** getCurrentHour() method was deprecated in API level 23. From api level 23 you have to use getHour(). This method returns an integer value.
5) **getCurrentMinute():**This method is used to get the current minutes from a time picker.
6) **getMinute():** getCurrentMinute() method was deprecated in API level 23. From api level 23 we have to use getMinute(). This method returns an integer value.
7) **setIs24HourView(Boolean is24HourView):**This method is used to set the mode of the Time picker either 24 hour mode or AM/PM mode. In this method we set a Boolean value either true or false. True value indicate 24 hour mode and false value indicate AM/PM mode.
8) **is24HourView():**This method is used to check the current mode of the time picker. This method returns true if its 24 hour mode or false if AM/PM mode is set.

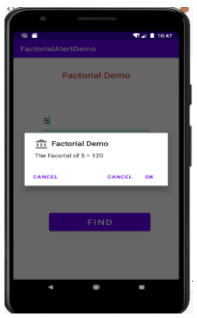
```
TimePicker simpleTimePicker = (TimePicker)findViewById(R.id.simpleTimePicker);
 // initiate a time picker
 simpleTimePicker.setOnTimeChangedListener(new TimePicker.OnTimeChangedListener()
 {
 @Override
public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
                    time.setText(hourOfDay + ":" + minute);
 } });
```

Set A

1. Create an Android application that demonstrate the Alert Dialog.
2. Write an Android code to merge given two Array/List



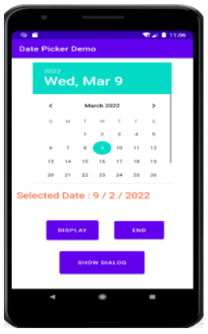
3. . Create an Android application that demonstrate the Custom Alert Dialog.
4. Create an Android Application to find the factorial of a number and Display the Result on Alert Box.



The Facorial of 5 = 120

Set B

1. Develop an Android application that create custom Alert Dialog containing Friends Name and onClick of Friend Name Button greet accordingly.
2. Create an Android Application that Demonstrate DatePicker and DatePickerDailog.

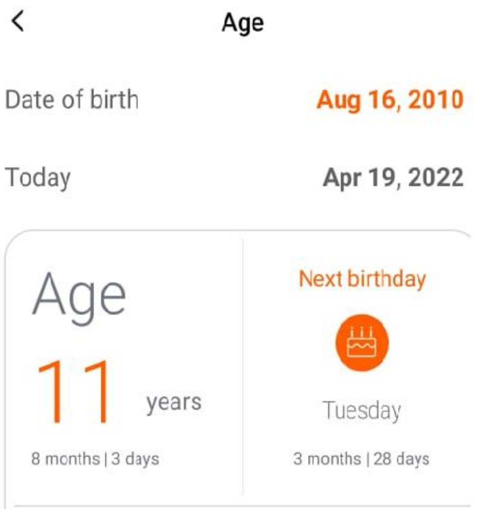


Selected Date : 9 / 2 / 2022

3. Create an Android Application that Demonstrate TimePicker and TimePickerDailog.

Set C

1) Create a Simple Android Application to calculate age of a person. (Use Table Layout)



Date of birth Aug 16, 2010

Today Apr 19, 2022

Age 11 years
8 months | 3 days

Next birthday
Tuesday
3 months | 28 days

Signature of the instructor: ------------------------- Date:------------------------

Assignment Evaluation

0: Not Done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs Improvement		5: Well Done	

Assignment 5 : Android Adapter and Menu

In Android, Adapter is a bridge between UI component and data source that helps us to fill data in UI component. It holds the data and send the data to an Adapter view then view can takes the data from the adapter view and shows the data on different views like as ListView, GridView, Spinner etc. For more customization in Views we uses the base adapter or custom adapters. To fill data in a list or a grid we need to implement Adapter.
There are the some commonly used Adapter in Android used to fill the data in the UI components.

1) **BaseAdapter** – It is parent adapter for all other adapters
2) **ArrayAdapter** – It is used whenever we have a list of single items which is backed by an array.
3) **Custom ArrayAdapter** – It is used whenever we need to display a custom list.
4) **SimpleAdapter** – It is an easy adapter to map static data to views defined in your XML file
5) **Custom SimpleAdapter** – It is used whenever we need to display a customized list and needed to access the child items of the list or grid

List View: It is widely used in android applications. A very common example of ListView is your phone contact book, where you have a list of your contacts displayed in a ListView and if you click on it then user information is displayed.
Adapter: To fill the data in a ListView we simply use adapters. List items are automatically inserted to a list using an Adapter that pulls the content from a source such as an arraylist, array or database.
Listview is present inside Containers. From there you can drag and drop on virtual mobile screen to create it. Alternatively you can also XML code to create it.
divider: This is a drawable or color to draw between different list items.
dividerHeight: This specify the height of the divider between list items. This could be in dp(density pixel), sp(scale independent pixel) or px(pixel).
XML Code :

```
 <ListView
     android:id="@+id/simpleListView“
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:divider="#f00"
     android:dividerHeight="1dp"
     android:listSelector="#0f0"/>
```

Java code :

```
   ListView simpleList;
  String countryList[] = {"Rajesh", "Ramesh", "Suresh", "Satish", "Rakesh", "Dinesh"};
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
 simpleList = (ListView)findViewById(R.id.simpleListView);
ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this,
R.layout.activity_listview,  R.id.textView, countryList);
simpleList.setAdapter(arrayAdapter);
}
```

GridView : In android GridView is a view group that display items in two dimensional scrolling grid (rows and columns), the grid items are not necessarily predetermined but they are automatically inserted to the layout using a Adapter. Users can then select any grid item by clicking on it. GridView is default scrollable. An example of GridView is your default Gallery, where you have number of images displayed using grid. To fill the data in a GridView we simply use adapter and grid items are automatically inserted to a GridView using an Adapter which pulls the content from a source such as an arraylist, array or database

XML Code :

```
<GridView
   android:id="@+id/simpleGridView" android:layout_width="fill_parent"
   android:layout_height="wrap_content" android:numColumns="3“
   android:verticalSpacing="50dp“
   android:horizontalSpacing="50dp“
   android:columnWidth="80dp"
   android:listSelector="#0f0"
/>
```

Spinner :

In Android, Spinner provides a quick way to select one value from a set of values. Android spinners are nothing but the drop down-list seen in other programming languages. In a default state, a spinner shows its currently selected value. It provides a easy way to select a value from a list of values.

XML Code :

```
<Spinner
  android:id="@+id/simpleSpinner"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content”
  android:layout_centerHorizontal="true"
  android:layout_marginTop="100dp" />
```

Java code :

```
String[] bankNames={"BOI","SBI","HDFC","PNB","OBC"};
Spinner spin = (Spinner) findViewById(R.id.simpleSpinner);
ArrayAdapter aa = new
ArrayAdapter(this,android.R.layout.simple_spinner_item,bankNames);
 aa.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
 spin.setAdapter(aa);
 spin.setOnItemSelectedListener(this);
public void onItemSelected(AdapterView<?> arg0, View arg1, int position,long id) {
Toast.makeText(getApplicationContext(), bankNames[position],
Toast.LENGTH_LONG).show(); }
```

Menu in Android : In android, there are three types of Menus available to define a set of options and actions in our android applications.

1) **Android Options Menu :** The options menu is the primary collection of menu items for an activity. It's where you should place actions that have a overall impact on the app, such as Search, Compose Email and Settings.
2) **Android Context Menu** : A context menu is a floating menu that appears when the user performs a long-click on an element. It provides actions that affect the selected content or context frame.

3) **Android Popup Menu** : A popup menu displays a list of items in a vertical list that is anchored(sticked) to the view that invoked the menu. It's good for providing an overflow of actions that relate to specific content or to provide options for a second part of a command.

How to create a Menu?

For all menu types, Android provides a standard XML format to define menu items. you should define a menu and all its items in an XML menu resource. You can then inflate the menu resource i.e load the XML files as a Menu object in your activity.

1) A new menu directory would be made under res directory.
2) Add menu_file.xml file in menu directory by right clicking on **menu --> New --> Menu resource file**.
3) Give the name as **menu_file.xml** and click on Ok.
4) The **menu_file.xml** file contains the following tags:
    - **a) <menu> :** It defines a Menu, which is a container for menu items. A <menu> element must be the root node for the file and can hold one or more <item> and <group> elements.
    - **b) <item>:** It creates a MenuItem, which represents a single item in a menu. This element may contain a nested <menu> element in order to create a submenu.
    - **c) <group> :**It is an optional, invisible container for <item> elements. It allows you to categorize menu items so they share properties such as active state and visibility.

menu_file.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <menu
   xmlns:android="http://schemas.android.com/apk/res/android">
 <item
        android:id="@+id/item1"
        android:title="item1“
         android:icon="@drawable/item" >
<!-- "item" submenu -->
        <menu>
                <item
                        android:id="@+id/a"
                        android:title="subitem a"
                        android:icon="@drawable/subitem_a"/>
                <item
                        android:id="@+id/b“
                         android:title="subitem b“
                         android:icon="@drawable/subitem_b" />
        </menu>
</item>
 </menu>
```

Making an Option Menu

- To make an option menu, we need to Override **onCreateOptionsMenu()** method as follows:

```
@Override
public boolean onCreateOptionsMenu(Menu menu)
{
        MenuInflater inflater = getMenuInflater();
        inflater.inflate(R.menu.menu_file, menu);
        return true;
```

}
MenuInflater inflater = getMenuInflater();

This gives a MenuInflater object that will be used to inflate(convert our XML file into Java Object) the menu_file.xml file.

inflater.inflate(R.menu.menu_file, menu);

inflate() method is used to inflate the menu_file.xml file.

Handling Click Events

- When the user selects an item from the options menu, the system calls your activity's onOptionsItemSelected() method.
- This method passes the MenuItem selected.
- You can identify the item by calling getItemId() method, which returns the unique ID for the menu item (defined by the android:id attribute in the menu resource).
- You can match this ID against known menu items to perform the appropriate action.

```
@Override

public boolean onOptionsItemSelected(MenuItem item) {

    //Handle item selection

     switch (item.getItemId()) {

          case R.id.i1:

         //perform any action; return true;

         case R.id.a:

         //perform any action; return true;

          case R.id.b:

         //perform any action; return true;

         default: return super.onOptionsItemSelected(item);

}

 }
```

Making Contextual Menu

- To make a floating context menu, you need to follow the following steps:
- Register the View to which the context menu should be associated by calling registerForContextMenu() and pass it the View.
- If your activity uses a ListView or GridView and you want each item to provide the same context menu, you can register yout all items for a context menu by passing the ListView or GridView object to registerForContextMenu() method.
- Implement the onCreateContextMenu() method in your Activity.

- When the registered view receives a long-click event, the system calls your onCreateContextMenu() method.
- This is where you define the menu items, usually by inflating a menu resource.

For example:

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v, ContextMenuInfo menuInfo) {

        super.onCreateContextMenu(menu, v, menuInfo);

        MenuInflater inflater = getMenuInflater();

         inflater.inflate(R.menu.menu_file, menu);

}
```

MenuInflater allows you to inflate the **menu_file.xml** file from a menu resource. The method parameters include the View that the user selected and aContextMenu. **ContextMenuInfo** object provides additional information about the item selected. If your activity has several views such that each provide a different context menu, you might use these parameters to determine which context menu to inflate.

Handling Click Events

```
@Override

public boolean onContextItemSelected(MenuItem item)

{

         switch (item.getItemId())
         {
                 case R.id.i1:
                 //Perform any action; return true;
                 case R.id.a:
                 //Perform any action; return true;
                  case R.id.b:
                 //Perform any action; return true;
                 default: return super.onContextItemSelected(item);
         }
 }
```

Making Popup Menu

- If you have define your menu_file.xml file in XML, here's how you can show the popup menu:
- Make an object of PopupMenu, whose constuctor takes the current application Context and the View to which the menu should be anchored.
- Use MenuInflater to inflate your menu resource into the Menu object returned by PopupMenu.getMenu()

- Call PopupMenu.show()

For example,

```
<Button android:id="@+id/button" android:layout_width="wrap_content"
android:layout_height="wrap_content" android:text="Button"
    android:onClick="pop" />
```

The activity can then show the popup menu like this:

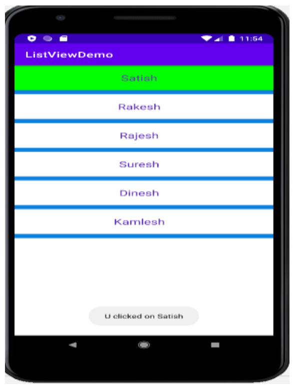
```
public void pop(View v)
{         PopupMenu popup = new PopupMenu(this,v);
    MenuInflater inflater = getMenuInflater(); inflater.inflate(R.menu.menu_file,popup.getMenu());
    popup.show();
 }
```

Listener

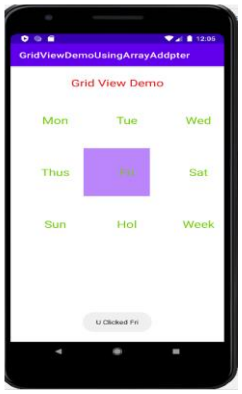
```
b1.setOnClickListener(new View.OnClickListener() {
   @Override
   public void onClick(View v) {
      final PopupMenu popup = new PopupMenu(getApplicationContext(),v);
      MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.menu_file,popup.getMenu());
      popup.show();

 popup.setOnMenuItemClickListener(new PopupMenu.OnMenuItemClickListener() {
          @Override
          public boolean onMenuItemClick(MenuItem item) {
  Toast.makeText(getApplicationContext(),"selected
"+item.getTitle(),Toast.LENGTH_LONG).show();
              return true;
          }
      });
```

Set A

1. Create an Android Application that Demonstrate ListView and Onclick of List Display the Toast.



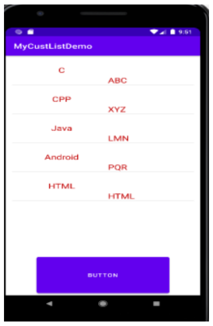
2. Create an Android Application that Demonstrate GridView and Onclick of Item Display the Toast.



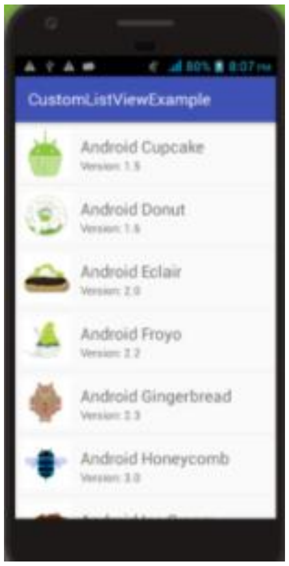
3. Create an Android Application that Demonstrate GridView with Images

Set B

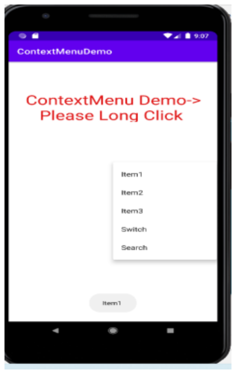
1. Create an Android Application that Demonstrate Custom ListView which shows the BookName and Author Name



2. Create a Custom ListView in Android Application

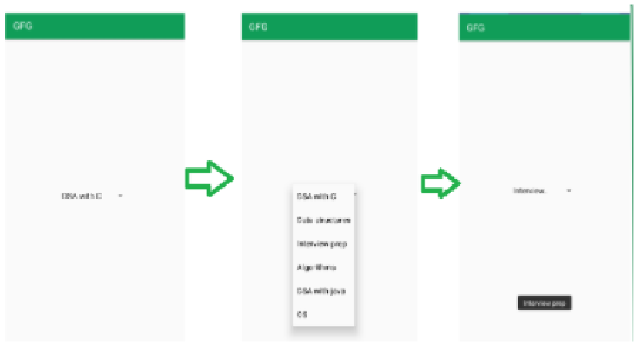


3. Create an Android Application that Demonstrate ContextMenu.

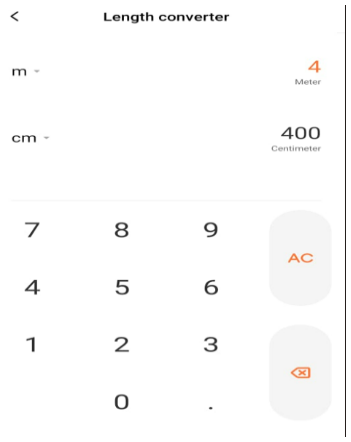


Set C

1. Create the following layout using spinner



2. Create an Android Application to demonstrate Length converter.



Signature of the instructor: ------------------------ Date:-----------------------

Assignment Evaluation

0: Not Done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs Improvement		5: Well Done	

Assignment 6: Thread and Services

Objective:

- To provide a way to improve application performance through parallelism.
- To ensure that the application remains active in the background so that the user can operate multiple applications at the same time.

Thread and Notification:

A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently. When an application is launched, the system creates a thread of execution for the application, called "main." This thread is very important because it is in charge of dispatching events to the appropriate user interface widgets, including drawing events.

Worker threads

Worker threads are background threads. They are the threads that are created separately, other than the UI thread. Since blocking the UI thread is restricted according to the rule, user should run the child processes and tasks in worker threads.

Example

```
public void onClick(View v) {
   new Thread(new Runnable() {
       public void run() {
           Bitmap b = loadImageFromNetwork("http://example.com/image.png");
           mImageView.setImageBitmap(b);
       }
   }).start();
}
```

In the above example code, the download operation is handled by a second thread other than the UI thread. But the program violates the second rule. The imageView from UI thread is manipulating from this worker thread. In the above example code, the download operation is handled by a second thread other than the UI thread. But the program violates the second rule. The imageView from UI thread is manipulating from this worker thread.

According to the second rule, UI could not be accessed from outside the UI thread. Solution for such a restriction is runOnUiThread(Runnable) method. The main or UI thread can be accessed from other threads using runOnUiThread(Runnable) method. As a result, the specified runnable action passed through this method will run on the UI thread. The action will execute immediately, if the current thread is in the UI itself. Else the action will be posted to the event queue.

Example:

```
MainClassName.this.runOnUiThread(new Runnable() {
                    public void run() {
                        textViewObject.setText("Set this " + value from worker thread + "!");
                    }
                });
```

Handlers & Runnable:

A handler is basically a message queue. You post a message to it, and it will eventually process it by calling its run method and passing the message to it. Since these run calls will always occur in the order of messages received on the same thread, it allows you to serialize events.

Uses of Handler:

Handler has two main uses

1. Scehduling of messages and runnables that need to be executed in the future.
2. Enqueueing actions that need to be performed in the background thread.

Handler is a class fundamental to how we do threading in android, at the infrastructure level. It works hand in hand with the Looper. Together they underpin everything that the main thread does and including the invocation of the Activity lifecycle methods.

Looper will take care of dispatching work on its message-loop thread. On the other hand Handler will serve two roles:

1. First it will provide an interface to submit messages to its Looper queue.
2. Secondly it will implement the callback for processing those messages when they are dispatched by the Looper.

Example:

```
public void onClick(View v) {
            nstoploop = true;
            new Thread(new Runnable() {
               @Override
               public void run() {
                  while (nstoploop){
                     try
                     {
                        Thread.sleep(500);
                        count++;

                     }catch (InterruptedException e){
                        Log.i(TAG,e.getMessage());
                     }
                     handler.post(new Runnable() {
                        @Override
                        public void run() {
                           tv.setText(" "+count);
                        }
                     });
                  }
               }
            }).start();
         }
      });
```

We can’t touch background thread to main thread directly so handler is going to collect all events which are available in main thread in a queue and posses this queue to looper class.

In android Handler is mainly used to update the main thread from background thread or other than main thread. There are two methods are in handler.

- **Post()** − it going to post message from background thread to main thread using looper.
- **sendmessage()** − if you want to organize what you have sent to ui (message from background thread) or ui functions. you should use sendMessage().

AsynTask:

Android AsyncTask going to do background operation on background thread and update on main thread. In android we can’t directly touch background thread to main thread in android development. asynctask help us to make communication between background thread to main thread.

Android AsyncTask is an abstract class provided by Android which gives us the liberty to perform heavy tasks in the background and keep the UI thread light thus making the application more responsive.

Android application runs on a single thread when launched. Due to this single thread model tasks that take longer time to fetch the response can make the application non-responsive. To avoid this we use android AsyncTask to perform the heavy tasks in background on a dedicated thread and passing the results back to the UI thread. Hence use of AsyncTask in android application keeps the UI thread responsive at all times.

Methods of AsyncTask

- **onPreExecute()** –
  Before doing background operation we should show something on screen like progressbar or any animation to user. we can directly comminicate background operation using on doInBackground().

- **doInBackground(Params)** –
  In this method we have to do background operation on background thread. Operations in this method should not touch on any mainthread activities or fragments.

- **onProgressUpdate(Progress…)** –
  While doing background operation, if you want to update some information on UI, we can use this method.

- **onPostExecute(Result)** –
  In this method we can update ui of background operation result.

Generic Types in Async Task

- **TypeOfVarArgParams** –
  It contains information about what type of params used for execution.

- **ProgressValue** –

It contains information about progress units. While doing background operation we can update information on ui using onProgressUpdate().

- **ResultValue** –
  It contains information about result type.

Example:

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
Button b1,b2;
TextView tv;
Boolean stoploop;
int count = 0;
String TAG = "Thread";
private MyAsynctask async;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        b1 = (Button)findViewById(R.id.btn1);
        b2 = (Button)findViewById(R.id.btn2);
        tv = (TextView)findViewById(R.id.textView);

        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
               stoploop = true;
                async = new MyAsynctask();
                async.execute(count);
            }
        });
        b2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
            stoploop = false;
            }
        });
    }
    private class MyAsynctask extends AsyncTask<Integer,Integer ,Integer>
    {
        private int ccount;

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            ccount=0;
        }

        @Override
        protected Integer doInBackground(Integer... integers) {
            ccount = integers[0];
            while(stoploop){
                try
                {
                    Thread.sleep(1000);
                    ccount++;
                    publishProgress(ccount);
                }
                catch (InterruptedException e){
                    Log.i(TAG," "+e.getMessage());
                }
            }
            return ccount;
        }

        @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
            tv.setText(""+values[0]);
        }

        @Override
        protected void onPostExecute(Integer integer) {
            super.onPostExecute(integer);
            tv.setText(""+integer);
            count = integer;
        }
    }
}
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:app="http://schemas.android.com/apk/res-auto"

    xmlns:tools="http://schemas.android.com/tools"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintHorizontal_bias="0.58"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.102" />

<Button
    android:id="@+id/btn1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Start Thread"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.6"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView"
    app:layout_constraintVertical_bias="0.116" />

<Button
    android:id="@+id/btn2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Stop Thread"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.6"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btn1"
```

```
        app:layout_constraintVertical_bias="0.145" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Broadcast Receiver :

Broadcast in android is the system-wide events that can occur when the device starts, when a message is received on the device or when incoming calls are received, or when a device goes to airplane mode, etc. Broadcast Receivers are used to respond to these system-wide events. Broadcast Receivers allow us to register for the system and application events, and when that event happens, then the register receivers get notified. There are mainly two types of Broadcast Receivers:

- **Static Broadcast Receivers:**
  These types of Receivers are declared in the manifest file and works even if the app is closed.
- **Dynamic Broadcast Receivers:**
  These types of receivers work only if the app is active or minimized.

Creating the Broadcast Receiver:

```
class AirplaneModeChangeReceiver:BroadcastReceiver() {
    override fun onReceive(context: Context?, intent: Intent?) {
        // logic of the code needs to be written here
    }
}
```

Registering a BroadcastReceiver:

```
IntentFilter(Intent.ACTION_AIRPLANE_MODE_CHANGED).also {
            // receiver is the broadcast receiver that we have registered
            // and it is the intent filter that we have created
            registerReceiver(receiver,it)
    }
```

Intent	Description Of Event
android.intent.action.BATTERY_LOW :	Indicates low battery condition on the device.
android.intent.action.BOOT_COMPLETED	This is broadcast once after the system has finished booting
android.intent.action.CALL	To perform a call to someone specified by the data

Intent	Description Of Event
android.intent.action.DATE_CHANGED	Indicates that the date has changed
android.intent.action.REBOOT	Indicates that the device has been a reboot
android.net.conn.CONNECTIVITY_CHANGE	The mobile network or wifi connection is changed(or reset)
android.intent. ACTION_AIRPLANE_MODE_CHANGED	This indicates that airplane mode has been switched on or off.

Example:

```
MainActivity.Java
public class MainActivity extends AppCompatActivity {
    private EditText txtPhone;
    private Button btn;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        txtPhone = (EditText)findViewById(R.id.mblTxt);
        btn = (Button)findViewById(R.id.btnCall);
        btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                callPhoneNumber();

            }
        });

    }

    @Override
    public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
        if(requestCode == 101)
        {
            if(grantResults[0] == PackageManager.PERMISSION_GRANTED)
            {
                callPhoneNumber();
            }
        }
    }
    public void callPhoneNumber()
    {
        try
        {
            if(Build.VERSION.SDK_INT > 22)
            {
                if (ActivityCompat.checkSelfPermission
                        (this, Manifest.permission.CALL_PHONE) !=
PackageManager.PERMISSION_GRANTED) {
                    ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.CALL_PHONE}, 101);
                    return;
                }

                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
                startActivity(callIntent);

            }
            else {
                Intent callIntent = new Intent(Intent.ACTION_CALL);
                callIntent.setData(Uri.parse("tel:" + txtPhone.getText().toString()));
                startActivity(callIntent);
            }
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }
    }
}
```

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/fstTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:layout_marginTop="150dp"
        android:text="Mobile No"
        />
    <EditText
        android:id="@+id/mblTxt"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:ems="10">
    </EditText>
    <Button
        android:id="@+id/btnCall"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="100dp"
        android:text="Call" />
</LinearLayout>
```

Services:

A service is a component that runs in the background to perform long-running operations without needing to interact with the user and it works even if application is destroyed.

Life Cycle of Android Service

There can be two forms of a service.The lifecycle of service can follow two different paths: started or bound.

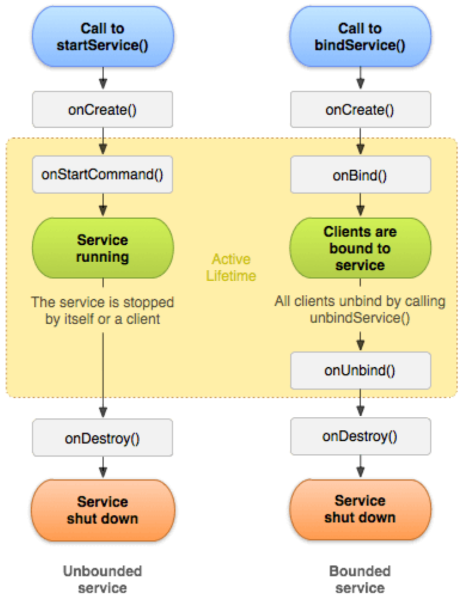
1. Started
2. Bound

1) Started Service

A service is started when component (like activity) calls **startService()** method, now it runs in the background indefinitely. It is stopped by **stopService()** method. The service can stop itself by calling the **stopSelf()** method.

2) Bound Service

A service is bound when another component (e.g. client) calls **bindService()** method. The client can unbind the service by calling the **unbindService()** method.



Fundamentals of Android Services:

Methods	**Description**
onStartCommand()	The Android service calls this method when a component(eg: activity) requests to start a service using startService(). Once the service is started, it can be stopped explicitly using stopService() or stopSelf() methods.
onBind()	This method is mandatory to implement in android service and is invoked whenever an application component calls the bindService() method in order to bind itself with a service. User-interface is also provided to communicate with the service effectively by returning an IBinder object. If the binding of service is not required then the method must return null.
onUnbind()	The Android system invokes this method when all the clients get disconnected from a particular service interface.
onRebind()	Once all clients are disconnected from the particular interface of service and there is a need to connect the service with new clients, the system calls this method.
onCreate()	Whenever a service is created either using onStartCommand() or onBind(), the android system calls this method. This method is necessary to perform a one-time-set-up.
onDestroy()	When a service is no longer in use, the system invokes this method just before the service destroys as a final clean up call.

Methods	Description
	Services must implement this method in order to clean up resources like registered listeners, threads, receivers, etc.

Example of Android Services:

Step 1: Working with the activity_main.xml file

Open the **activity_main.xml** file and add 2 Buttons in it which will start and stop the service. Below is the code for designing a proper activity layout.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:id="@+id/linearLayout"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_centerVertical="true"
        android:orientation="vertical"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="1.0"
        tools:ignore="MissingConstraints">

        <TextView
            android:id="@+id/textView1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="170dp"
            android:text="@string/heading"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Large"
            android:textColor="@android:color/holo_green_dark"
            android:textSize="36sp"
            android:textStyle="bold" />

        <Button
```

```
            android:id="@+id/startButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:text="@string/startButtonText"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />

        <Button
            android:id="@+id/stopButton"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:layout_marginStart="20dp"
            android:layout_marginTop="10dp"
            android:layout_marginEnd="20dp"
            android:layout_marginBottom="20dp"
            android:background="#4CAF50"
            android:text="@string/stopButtonText"
            android:textAlignment="center"
            android:textAppearance="@style/TextAppearance.AppCompat.Display1"
            android:textColor="#FFFFFF"
            android:textStyle="bold" />


    </LinearLayout>

</androidx.constraintlayout.widget.ConstraintLayout>
```

Step 2: Creating the custom service class

A custom service class will be created in the same directory where the **MainActivity** class resides and this class will extend the **Service class**.

```
public class NewService extends Service {
  private MediaPlayer player;

    @Override
    public int onStartCommand(Intent intent, int flags, int startId) {
        player = MediaPlayer.create( this, R.raw.ss);

        // providing the boolean
        // value as true to play
        // the audio on loop
        player.setLooping( true );

        // starting the process
        player.start();

        // returns the status
        // of the program
        return START_STICKY;
    }

    @Override
    public void onDestroy() {
        super.onDestroy();
        player.stop();
    }

    @Nullable
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
}
```

Step 5: Working with the MainActivity file

Now, the button objects will be declared and the process to be performed on clicking these buttons will be defined in the MainActivity class.

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    private Button start, stop;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        start = (Button) findViewById( R.id.startButton );

        // assigning ID of stopButton
        // to the object stop
        stop = (Button) findViewById( R.id.stopButton );

        // declaring listeners for the
        // buttons to make them respond
        // correctly according to the process
        start.setOnClickListener(this );
        stop.setOnClickListener(this );
    }
    public void onClick(View view) {

        // process to be performed
        // if start button is clicked
        if(view == start){

            // starting the service
            startService(new Intent( this, NewService.class ) );
        }

        // process to be performed
        // if stop button is clicked
        else if (view == stop){

            // stopping the service
            stopService(new Intent( this, NewService.class ) );

        }
    }
}
```

Step 6: Modify the AndroidManifest.xml file

To implement the services successfully on any android device, it is necessary to mention the created service in the **AndroidManifest.xml** file.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.servicedemo">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".NewService"/>
    </application>
</manifest>
```

Notification:

A **notification** is a message you can display to the user outside of your application's normal UI. When you tell the system to issue a notification, it first appears as an icon in the notification area. To see the details of the notification, the user opens the notification drawer. Both the notification area and the notification drawer are system-controlled areas that the user can view at any time.

The NotificationCompat.Builder Class

Method	Description
Notification build()	**Combine all of the options that have been set and return a new Notification object.**
NotificationCompat.Builder setAutoCancel (boolean autoCancel)	**Setting this flag will make it so the notification is automatically canceled when the user clicks it in the panel.**
NotificationCompat.Builder setContent (RemoteViews views)	**Supply a custom RemoteViews to use instead of the standard one.**
NotificationCompat.Builder setContentTitle (CharSequence title)	**Set the text (first row) of the notification, in a standard notification.**

Create and Send Notifications

Step 1 - Create Notification Builder

As a first step is to create a notification builder using *NotificationCompat.Builder.build()*. You will use Notification Builder to set various Notification properties like its small and large icons, title, priority etc.

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this)
```

Step 2 - Setting Notification Properties

Once you have **Builder** object, you can set its Notification properties using Builder object as per your requirement. But this is mandatory to set at least following –

- A small icon, set by **setSmallIcon()**
- A title, set by **setContentTitle()**
- Detail text, set by **setContentText()**

Example:

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Notification Alert, Click Me!");
mBuilder.setContentText("Hi, This is Android Notification Detail!");
```

Step 3 - Attach Actions

This is an optional part and required if you want to attach an action with the notification. An action allows users to go directly from the notification to an **Activity** in your application, where they can look at one or more events or do further work.

The action is defined by a **PendingIntent** containing an **Intent** that starts an Activity in your application. To associate the PendingIntent with a gesture, call the appropriate method of *NotificationCompat.Builder*. For example, if you want to start Activity when the user clicks the notification text in the notification drawer, you add the PendingIntent by calling **setContentIntent()**.

A PendingIntent object helps you to perform an action on your applications behalf, often at a later time, without caring of whether or not your application is running.

Example:

```
Intent resultIntent = new Intent(this, ResultActivity.class);
TaskStackBuilder stackBuilder = TaskStackBuilder.create(this);
stackBuilder.addParentStack(ResultActivity.class);

// Adds the Intent that starts the Activity to the top of the stack
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent =
stackBuilder.getPendingIntent(0,PendingIntent.FLAG_UPDATE_CURRENT);
mBuilder.setContentIntent(resultPendingIntent);
```

Step 4 - Issue the notification

Finally, you pass the Notification object to the system by calling NotificationManager.notify() to send your notification. Make sure you call **NotificationCompat.Builder.build()** method on builder object before notifying it. This method combines all of the options that have been set and return a new **Notification** object.

Example:

```
NotificationManager mNotificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);

// notificationID allows you to update the notification later on.
mNotificationManager.notify(notificationID, mBuilder.build());
```

Accessing Phone services(Call,SMS) :

Messaging and E-mail

Sending SMS messages

SMS messages can be send using SmsManager Class or Built-in SMS application.

Using SmsManager class: Using the SmsManager class, you can send SMS messages from within your application without the need to involve the built-in Messaging application.

Steps to send SMS

Create a new Android project and name it SMS. Replace the TextView with the following statements in the ***main.xml*** file:

Example:

```
<Button android:id=‖@+id/btnSendSMS‖
            android:layout_width=‖fill_parent‖
            android:layout_height=‖wrap_content‖
            android:text=‖Send SMS‖
            android:onClick=‖onClick‖ />
```

In the AndroidManifest.xml fi le, add the following statements:

```
  <uses-permission android:name=‖android.permission.SEND_SMS/>
```

Add the following statements to the ***SMSActivity.java*** file:

```
  import android.telephony.SmsManager;
  import android.view.View;
  public void onClick(View v) {
              sendSMS(―5556‖, ―Hello my friends!‖);
  }
  //---sends an SMS message to another device--
    private void sendSMS(String phoneNumber, String message)
                                          {

              SmsManager sms = SmsManager.getDefault();
              sms.sendTextMessage(phoneNumber, null, message, null, null);
  }
```

Methods of SmsManager Class:

Methods	Description
ArrayList<String> divideMessage(String text)	This method divides a message text into several fragments, none bigger than the maximum SMS message size.
static SmsManager getDefault()	This method is used to get the default instance of the SmsManager
void sendDataMessage (String destAddress , String scAddress, short destinationPort, byte[]data, PendingIntent sentIntent, PendingIntent deliveryIntent)	This method is used to send a data based SMS to a specific application port.
void sendMultipartTextMessage (String destAddress, String scAddress, ArrayList<String> parts, ArrayList<PendingIntent> sentIntents,	Send a multi-part text based SMS.

ArrayList<PendingIntent> deliveryIntents)	
void sendTextMessage(String destinationAddress, String scAddress, String text, PendingIntent sentIntent, PendingIntent deliveryIntent)	Send a text based SMS.

Sending SMS messages using Intent:

We can also use built-in messaging application to send the message.

Steps to send SMS

Create an Intent Object (Action to send SMS)
ACTION_VIEW action is used to launch an SMS client installed on your Android device.

Intent smsIntent = new Intent(Intent.ACTION_VIEW);

Using Intent Object set Data/Type to send SMS

smsto: is used as a URI to send message using setData() method.

data type is set to vnd.android-dir/mms-sms using setType() method.

Example:

smsIntent.setData(Uri.parse("smsto:"));

smsIntent.setType("vnd.android-dir/mms-sms");

Intent Object used to set a message for one or more phone number with putExtra( ) method and a message can be send to multiple receivers separated by _:‘.

smsIntent.putExtra(―address‖,new String(―0123456789;3398765587‖));

OR

smsInent.putExtra(―address‖,‖5556 ; 5553 ; 5566‖);

Sending E-mail:

We can send Email by using Email/Gmail application available on Android. An Email account is configured by using POP3 or IMAP.

Steps to send Email

Create Android project and name it as Emails

Add the following statements in main.xml file:

```
<Button android:id=”@+id/btnSendEmail”

android:layout_width=”fill_parent” android:layout_height=”wrap_content”

android:text=”Send Email”

android:onClick=”onClick” />
```

Add the following statements in ***MainActivity.java*** file:

```
public class MainActivity extends Activity {
        Button btnSendEmail;
        /** Called when the activity is first created. */
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                btnSendEmail = (Button) findViewById(R.id.btnSendEmail);
                btnSendEmail.setOnClickListener(new View.OnClickListener() {
                        public void onClick(View v) {

                                String[] to = {―weimenglee@learn2develop.net‖,
―weimenglee@gmail.com‖};
                                String[] cc = {― ‖};

                                sendEmail(to, cc, ―Hello‖, ―Hello my friends!‖);
                        }
                });
        }
        //---sends an SMS message to another device---
        private void sendEmail(String[] emailAddresses, String[] carbonCopies, String subject,
String message)
        {
                Intent emailIntent = new Intent(Intent.ACTION_SEND);
                emailIntent.setData(Uri.parse(―mailto:‖));
                String[] to = emailAddresses;
                String[] cc = carbonCopies;
                emailIntent.putExtra(Intent.EXTRA_EMAIL, to);
                emailIntent.putExtra(Intent.EXTRA_CC, cc);
                emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
                emailIntent.putExtra(Intent.EXTRA_TEXT, message);
                emailIntent.setType(―message/rfc822‖);
                startActivity(Intent.createChooser(emailIntent, ―Email‖));

        }

}
```

Lab Assignments:

SET A

1. Create an Android application to service in android that plays an audio in the background. Audio will not be stopped even if you switch to another activity. To stop the audio, you need to stop the service.
2. Create application to send and receive messages using SMSManager.
3. Create application to send email.
4. Create a Notification in Android and display the notification message on second activity.

SET B

1. Create application to design login form, validate it. Write and send email with appropriate message.
2. Create an Android application to demonstrate Progress Dialog Box using AsyncTask
3. Create an Android application to demonstrate phone call in android using Implicit Intent.

SET C

1. Create an Android application to demonstrates how to use a service to download a file from the Internet based on a button click from an activity. Once done, the service notifies the activity via a broadcast receiver that the download is complete.
2. Create application to send email with attachment.

Signature of the instructor: ------------------------ Date:----------------
----

Assignment Evaluation

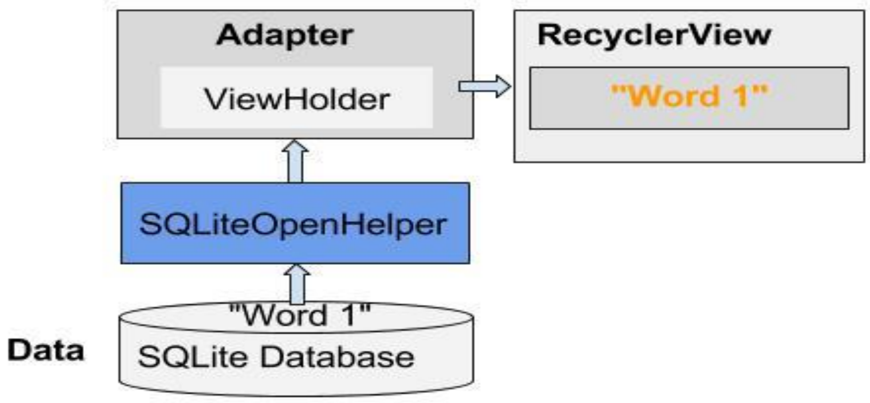
0: Not Don ☐	2: Late Complete ☐	4: ☐
Complete ☐	☐	☐
1: Incomplete	3: Needs Improvement	5: Well Done

Assignment 7: SQLite Database Connection

Objective:

- How to use database in an application.
- How to create and execute database queries

SQLite is an **open-source relational database** i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database. It is embedded in android bydefault. So, there is no need to perform any database setup or administration task. SQLite is an open source SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation. SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like ODBC.



SQLiteOpenHelper

Android database sqlite.SQLiteOpenHelper manages database creation, up gradation, down gradation, version management and opening it. We need to create sub class of SQLiteOpenHelper and override onCreate and onUpgrade and optionally onOpen. If database is not created, onCreate is called where we write script for database creation. If already created, then onOpen is called which opens database.

There are two constructors of SQLiteOpenHelper class.

Constructors	**Description**
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	Create a helper object to create, open, and/or manage a database
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	Create a helper object to create, open, and/or manage a database.

Public Methods	**Description**
void close()	Close any open database object.
String getDatabaseName()	Return the name of the SQLite database being opened, as given to the constructor.
SQLiteDatabase getReadableDatabase()	Create and/or open a database.
SQLiteDatabase getWritableDatabase()	Create and/or open a database that will be used for reading and writing.
void onConfigure(SQLiteDatabase db)	Called when the database connection is being configured, to enable features such as write-ahead logging or foreign key support.
abstract void onCreate(SQLiteDatabase db)	Called when the database is created for the first time.
void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	Called when the database needs to be downgraded.
void onOpen(SQLiteDatabase db)	Called when the database has been opened.

void setIdleConnectionTimeout(long idleConnectionTimeoutMs)	Sets the maximum number of milliseconds that SQLite connection is allowed to be idle before it is closed and removed from the pool.
public synchronized void close ()	Closes the database object.

Example:

```
public class DBHelper extends SQLiteOpenHelper{

        public DBHelper(){

                super(context,DATABASE_NAME,null,1);
        }
        public  void  onCreate(SQLiteDatabase db){

    Creates new database
    Create the tables
    execute query with the help of execSQL();
    Add initial data
          }

          public void  onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion){
          }
}
```

SQLite Database:

The main package is android.database.sqlite that contains the classes to manage your own databases:

Database – Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. Its syntax is given below:

```
SQLiteDatabase mydatabase = openOrCreateDatabase("Database name",MODE_PRIVATE,null);
```

Database – Insertion

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below public void insert(String name, String desc) { ContentValues contentValue = new ContentValues(); contentValue.put(DatabaseHelper.SUBJECT, name); contentValue.put(DatabaseHelper.DESC, desc); database.insert(DatabaseHelper.TABLE_NAME, null, contentValue); }

Content Values creates an empty set of values using the given initial size.

Database – Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

Example:

```
Cursor resultSet=mydb.rawQuery(―Select * from
 Login‖,null); resultSet.moveToFirst();
String username= resultSet.getString(0);
String password= resultSet.getString(1);
```

Methods	**Description**
int getColumnCount()	Returns the total number of columns of the table.
int getColumnIndex(String columnName)	Returns the index number of a column by specifying the name of the column.
String getColumnName (int columnIndex)	Returns the name of the column by specifying the index of the column.
String[ ] getColumnNames()	Returns the array of all the column names of the table.
int getCount()	Returns the total number of rows in the cursor.
int getPosition()	Returns the current position of the cursor in the table.
boolean isClosed()	Returns true if the cursor is closed.

Lab Assignments

SET A

1. Create table Customer (id, name, address, phno). Create Android Application for performing the following operation on the table. (using sqlite database)

i) Insert New Customer Details.
ii) Show All the Customer Details

2. Create Table Employee(Eno,Ename,Designation,Salary). Create Android Application for performing the following operation on the table. (Using SQLite Database)

i) Insert New Employee Details.
ii) Display specific employee details.
iii) Display all the Emplyee details.

3. Create simple application shown below. Create table Student(Sid ,Sname ,phno). Use autoincrement for Sid and Perform following Operation.

a. Add Student and display its information.
b. Delete Student.

4. Create sample application with login module (Check username and password). On successful login, pass username to next screen And on failing login, alert user using Toast (Hint :Use Login(username, password) Table.

SET B

1. Create Table project (pno, p_name, ptype, duration) and employee (id, e_name, qulification, joindate)

Project – employee have many to many relationship.

Using database perform following operation.

1) Add new record into table.
2) Display all the project Details.

2. Create Table Employee(Eno,Ename,Designation,Salary). Create Android Application for performing the following operation on the table. (Using SQLite Database)

i) Insert New Employee Details.
ii) Display maximum and minimum salary from employees table
iii) Display average salary and number of employees

3. Create Table Student(Studno,Studname,StudTotalmarks,Studno_of_subjects). Create Android Apllication for performing the following operation on the table. (Using SQLite Database)

i) Insert New student Details.
ii) Display student details.
iii) Display the percent of all students

SET C

1. Create table Game(no,name,type, no_of_players). Create Application to perform the following operations.

    i) Insert Game Details.
    ii) Update no_of_players to four where game is Badminton.
    iii) Display all the records.

2. Create Table Employee(empno,emp_name,dept,salary,branch). Create Android Application to perform following operations.

i) Display all Fields of employee table.

ii) Display the name of employee in descending order.

iii) Display the total salary of employee which is greater than > 120000

Signature of the instructor: ------------------------- Date:------------------------

Assignment Evaluation

0: Not Done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs Improvement		5: Well Done	

Assignment 8:

Location-Based Services and Google Map

Objectives:

- Study the location based services in android
- Use various operations of Google Map

Location based services: ―Location-based services‖ is used to create an application which uses current location, location updates, and location information. The two main LBS elements are:

Location Manager — these services allow applications to obtain periodic updates of the device's geographical location.

Location Providers — provides periodic reports on the geographical location of the device.

Google Map: Google Maps is web based service developed by Google. It provides many facilities such as Satellite view, Street view, real time traffic condition, and navigations for travelling by foot, car, bicycle and public transportation. To use Google Maps in your Android applications programmatically.

<u>**Google API (A**pplication **P**rogramming **I**nterface)</u>

Google APIs is a set of application programming interfaces (APIs) developed by Google which allow communication with Google Services and their integration to other services. Your application needs an API key to access the Google Maps servers. The key is free. You can use it with any of your applications that call the Google Maps Android API.

Get Google Maps API Key

Use the link provided in the ***google_maps_api.xml*** file.

Copy the link provided in the ***google_maps_api.xml*** file and pastes it into your browser. The link takes you to the Google API Console and supplies the required information to the Google API Console via URL parameters.

Follow the instructions to create a new project on the Google API Console or select an existing project.
Create an Android-restricted API key for your project.
Copy the resulting API key, go back to Android Studio, and paste the API key into the <string> element in the ***google_maps_api.xml*** file.

Displaying the Map

Classes and Interface used for displaying Map are given below:

Interfaces:

Package Name: com.google.android.gms.maps

OnMapReadyCallback – This interface is used to execute method when Map is ready.

Public Method	**Description**
abstract void onMapReady (GoogleMap googleMap)	This method execute automatically when map is ready

Package Name: android.app

FragmentManager: FragmentManager is a class used to create transactions for adding, removing or replacing fragments. It used to interact with Fragment inside the activity.

Public Methods	**Description**
Fragment findFragmentById(int id)	Finds a fragment that was identified by the given id.

Fragment findFragmentByTag(String tag)	Finds a fragment that was identified by the given tag.

GoogleMap: This is the main class of the Google Maps Android API. You cannot instantiate a GoogleMap object directly, rather, you must obtain one from the getMapAsync() method on a MapFragment. This class provides methods to set type of map, add markers, add polyline or move camera etc.

Constants	**Description**
MAP_TYPE_HYBRID MAP_TYPE_NONE MAP_TYPE_NORMAL	Satellite maps with a transparent layer of major streets. No base map tiles. Basic maps.
MAP_TYPE_SATELLITE	Satellite maps with no labels.
MAP_TYPE_TERRAIN	Terrain maps (Topographic data).

Lab Assignments

SET A

Write a program to perform Zoom In, Zoom Out operation and display Satellite view.

Write a program to perform Zoom In, Zoom Out operation and display Terrain view of current location on Google Map.

Write a program to find the specific location of an Android device and display details of the place like Address line, city with Geocoding.

Write a program to search a specific location on Google Map.

SET B

Write a program to calculate distance between two locations on Google Map.

Write a program to demonstrate navigation using Google Map.

Write a program to add Marker for indicating specific location using Google Map

SET C

Write a program to track an android device using mobile number and display the location on Google Map.

Write a program to modify the above program to draw the path along a route on Google Map.

Signature of the instructor: ------------------------ Date:-----------------------

Assignment Evaluation

0: Not Done		2: Late Complete		4: Complete	
1: Incomplete		3: Needs Improvement		5: Well Done	

Section II: Dot Net

ASSIGNMENT -A: DOT NET FRAMEWORK

Basic:

The **.NET Framework** is a software development platform that was introduced by Microsoft in the late 1990 under the NGWS. On 13 February 2002, Microsoft launched the first version of the .NET Framework, referred to as the **.NET Framework 1.0**.

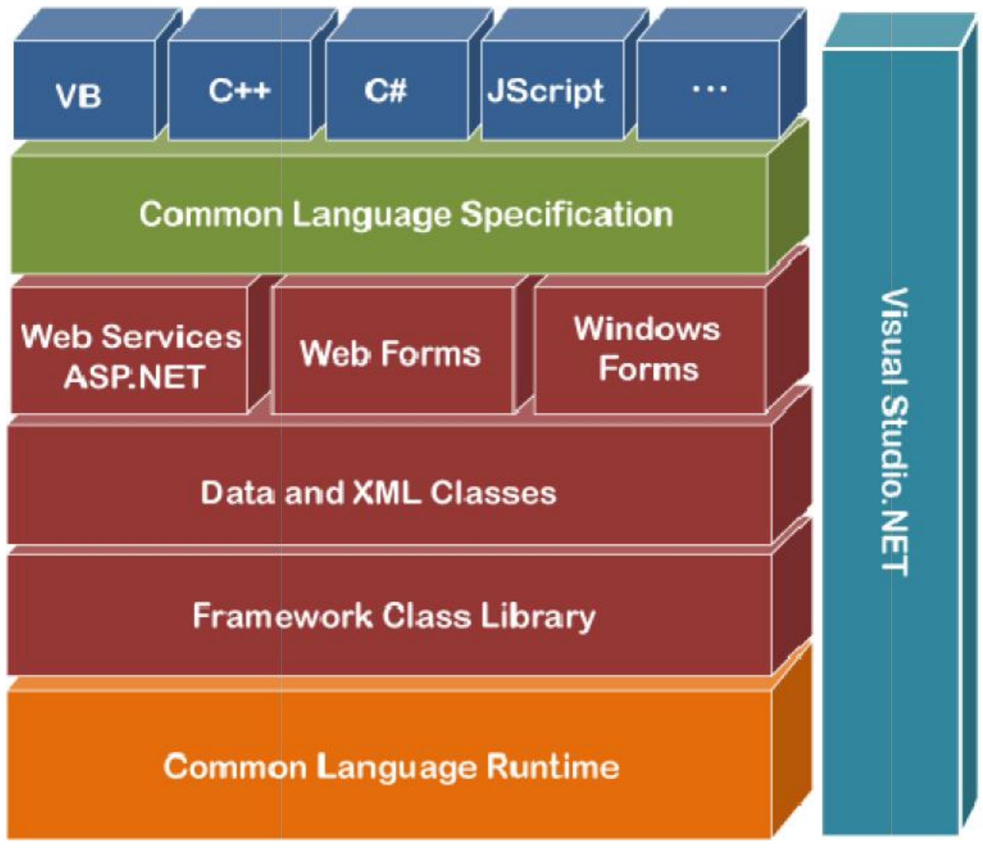
.NET Framework is a virtual machine that provide a common platform to run an application that was built using the different language such as C#, VB.NET, Visual Basic, etc. It is also used to create a form based, console-based and web-based application or services that are available in Microsoft environment. The .NET framework is a pure object oriented, that similar to the Java language . But it is not a platform independent as the Java. So, its application runs only to the windows platform.

The main objective of this framework is to develop an application that can run on the windows platform.

IDE (Integrated Development Environment)

Visual Studio is the Integrated Development Environment (IDE) provided by Microsoft in which developers can write and execute their program to develop various types of applications such as Windows, Web-based, console-based application easily. It has a rich collection of tools that are used to write and modify the programs and also help to detect and correct the errors in your programs. It is not a language-specific IDE, it means Visual Studio is not only for VB.NET language, but you can use this to write code in C#, visual basic, C++, Python, JavaScript and many more languages supported by Visual Studio. It supports 36 different languages that are used to develop an application. It has a built-in compiler to run the application, and it is also available for Windows and Mac OS.

Architecture of Dot Net Framework



Components of .NET Framework

There are following components of .NET Framework:

1. CLR (Common Language Runtime)
2. CTS (Common Type System)
3. BCL (Base Class Library)
4. CLS (Common Language Specification)
5. FCL (Framework Class Library)
6. .NET Assemblies
7. XML Web Services
8. Window Services

CLR (common language runtime)

It is an important part of a .NET framework that works like a virtual component of the .NET Framework to executes the different languages program like C#, Visual Basic, etc. A CLR also helps to convert a source code into the byte code, and this byte code is known as CIL (Common Intermediate Language) or MSIL (Microsoft Intermediate Language). After converting into a byte code, a CLR uses a JIT compiler at run time that helps to convert a CIL or MSIL code into the machine or native code.

CTS (Common Type System)

It specifies a standard that represent what type of data and value can be defined and managed in computer memory at runtime. A CTS ensures that programming data defined in various languages should beinteract with each other to share information. For example, in C# we define data type as int, while in VB.NET we define integer as a data type.

BCL (Base Class Library)

The base class library has a rich collection of libraries features and functions that help to implement many programming languages in the .NET Framework, such as C #, Visual C++, and more. Furthermore, BCL divides into two parts:

1. **User defined class library**
    - **Assemblies -** It is the collection of small parts of deployment an application's part. It contains either the DLL (Dynamic Link Library) or exe (Executable) file.
2. **Predefined class library**
    - **Namespace -** It is the collection of predefined class and method that present in .Net. In other languages such as, C we used header files, in java we used package similarly we used "using system" in .NET, where using is a keyword and system is a namespace.

CLS (Common language Specification)

It is a subset of common type system (CTS) that defines a set of rules and regulations which should be followed by every language that comes under the .net framework. In other words, a CLS language should be cross-language integration or interoperability. For example, in C# and VB.NET language, the C# language terminate each statement with semicolon, whereas in VB.NET it is not end with semicolon, and when these statements execute in .NET Framework, it provides a common platform to interact and share information with each other.

FCL (Framework Class Library)

It provides the various system functionality in the .NET Framework, that includes classes, interfaces and data types, etc. to create multiple functions and different types of application such as desktop, web, mobile application, etc. In other words, it can be defined as, it provides a base on which various applications, controls and components are built in .NET Framework.

ASSIGNMENT -B: VB.Net

Overview of VB.NET

The VB.NET stands for Visual Basic .Network Enabled Technologies. It is a simple, high-level, object-oriented programming language developed by Microsoft in 2002 to replace Visual Basic 6.
VB.NET is an object-oriented programming language. This means that it supports the features of object-oriented programming which include encapsulation, polymorphism, abstraction and inheritance.

Types of Applications that can be developed using VB.NET are :

- Windows Application
- Console Application
- Web Application

Microsoft Visual Studio Interface :

- Menubar
- Toolbar
- Toolbox
- Form Designer Window
- Code Window
- Properties Window
- Server Explorer
- Solution Explorer

A VB.NET define the following structure to create a program:

- Namespace declaration
- Procedure can be multiple
- Define a class or module
- Variables
- The Main procedure
- Statement and Expression
- Comments

Comments
In VB .NET, you write a comment by writing an apostrophe ' or writing REM. This means the rest of the line will not be taken into account by the compiler.

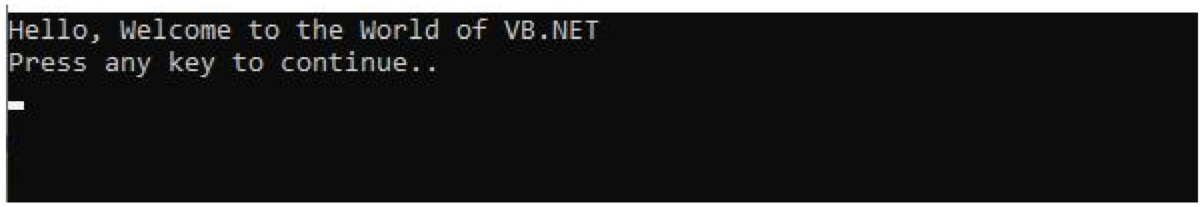
Hello World Program in VB.Net

Create a Hello_Program.vb file in MYConsoleApp project and write the following code:

```
Imports System       'System is a Namespace

Module Hello_Program

    Sub Main()
        Console.WriteLine("Hello, Welcome to the world of VB.NET")
        Console.WriteLine("Press any key to continue...")
        Console.ReadKey()
      End Sub
 End Module
```

Compile and run the above program by pressing the F5 key, we get the follwoing output.

Output:



```
Hello, Welcome to the World of VB.NET
Press any key to continue..
```

Data Types in VB.Net

Data types determine the type of data that any variable can store. Variables belonging to different data types are allocated different amounts of space in the memory.VB.Net provides a wide range of data types. The following table shows all the data types

Data Type	Storage Allocation	Value Range
Boolean	Depends on implementing platform	**True** or **False**
Byte	1 byte	0 through 255 (unsigned)
SByte	1 byte	-128 through 127 (signed)
UInteger	4 bytes	0 through 4,294,967,295 (unsigned)
ULong	8 bytes	0 through 18,446,744,073,709,551,615 (unsigned)
UShort	2 bytes	0 through 65,535 (unsigned)
Short	2 bytes	-32,768 through 32,767 (signed)

Integer	4 bytes	-2,147,483,648 through 2,147,483,647 (signed)
Long	8 bytes	-9,223,372,036,854,775,808 through 9,223,372,036,854,775,807(signed)
Single	4 bytes	-3.4028235E+38 through -1.401298E-45 for negative values; 1.401298E-45 through 3.4028235E+38 for positive values
Double	8 bytes	-1.79769313486231570E+308 through -4.94065645841246544E-324, for negative values 4.94065645841246544E-324 through 1.79769313486231570E+308, for positive values
Decimal	16 bytes	0 through +/-79,228,162,514,264,337,593,543,950,335 (+/-7.9...E+28) with no decimal point; 0 through +/-7.9228162514264337593543950335 with 28 places to the right of the decimal
Char	2 bytes	0 through 65535 (unsigned)
String	Depends on implementing platform	0 to approximately 2 billion Unicode characters
Date	8 bytes	0:00:00 (midnight) on January 1, 0001 through 11:59:59 PM on December 31, 9999
Object	4 bytes on 32-bit platform 8 bytes on 64-bit platform	Any type can be stored in a variable of type Object

Summary of Data Types :

Integer Values ▸ **Byte** ▸ **Sbyte** ▸ **Integer** ▸ **Uinteger** ▸ **Long** ▸ **Ulong** ▸ **Short** ▸ **Ushort**	**Floating Point Values** ▸ **Decimal** ▸ **Single** ▸ **Double**
Boolean Type	**Character Type**

▸ **Boolean**	▸ **Char** ▸ **String**
Date Type ▸ **Date**	**Any type of Value** ▸ **Object**

Example of Datatypes :

```
Module DataTypes
  Sub Main()
    Dim b As Byte
    Dim n As Integer
    Dim si As Single
    Dim d As Double
    Dim c As Char
    Dim s As String
    Dim bl As Boolean

    b = 1
    n = 1234567
    si = 0.12345678901234566
    d = 0.12345678901234566
   c = "U and me"
    s = "Me"

      Console.Write(c & " and," & s & vbCrLf)
      Console.WriteLine("We will learn VB.Net seriously")
      Console.WriteLine("The Boolean Value", b1)
      Console.WriteLine("The Integer Value", n)
      Console.WriteLine("Lets see what happens to the floating point variables:")
      Console.WriteLine("The Single: {0}, The Double: {1}", si, d)


       Console.ReadKey()
  End Sub
End Module
```

Type Conversion Functions
There are functions that we can use to convert from one data type to another. They include:

- **CBool** (expression): converts the expression to a Boolean data type.

- **CDate**(expression): converts the expression to a Date data type.
- **CDbl**(expression): converts the expression to a Double data type.
- **CByte** (expression): converts the expression to a byte data type.
- **CChar**(expression): converts the expression to a Char data type.
- **CLng**(expression): converts the expression to a Long data type.
- **CDec**(expression): converts the expression to a Decimal data type.
- **CInt**(expression): converts the expression to an Integer data type.
- **CObj**(expression): converts the expression to an Object data type.
- **CStr**(expression): converts the expression to a String data type.
- **CSByte**(expression): converts the expression to a Byte data type.
- **CShort**(expression): converts the expression to a Short data type.

Variable Declaration

In VB.NET, the declaration of a variable involves giving the variable a name and defining the data type to which it belongs. We use the following syntax:

**Syntax : ** Dim <Variable_Name> As <Data_Type>

**Example : ** Dim x As Integer

In the above example, 'x' is the variable name while Integer is the data type to which variable x belongs.

Variable Initialization

Initializing a variable means assigning a value to the variable. The following example demonstrates this:

Example-1 :

```
Dim x As Integer
 x = 10
```

Example-2 :

```
Dim name As String
 name  = “Akshit”
```

Example-3 :

```
Dim flag As Boolean
 flag  = True
```

Input Output Functions used in Console Application

Output Functions :

1) Write( ) :

Write() method displays the output but do not provide a new line character.

Example :

```
Console.Write("One")
Console.Write("Two")
Output : OneTwo
```

2) WriteLine( )

WriteLine() method displays the output and also provides a new line character it the end of the string, This would set a new line for the next output.

Example :

```
Console.WriteLine("One")
Console.WriteLine("Two")
Output : One
         Two
```

Input Functions : ReadLine( ), Read( ), ReadKey( )

Console.ReadLine()

It read all the characters from user input. (and finish when press enter).

Note: It return a STRING so data type should be STRING.

Example 1:

```
Dim name As String
Console.WriteLine("Enter student name : ")
name = Console.ReadLine()
Console.WriteLine( name)                         o/p ------------ Sachin Saxena
          or
Console.WriteLine("Student Name is = {0}", name)
                                        o/p ------------ My Name is = Sachin Saxena
```

In this example, if you enter the string "**Sachin Saxena**" it read as it is. It means *it reads all characters until the end of line.*

Example 2:

```
Dim sum As Integer
Console.WriteLine("Enter two numbers : ")
sum = CInt(Console.ReadLine()) + CInt(Console.ReadLine())
Console.WriteLine("Addition = {0}", sum)
```

Example 3:

```
Dim num1, num2, sum As Integer
Console.WriteLine("Enter two numbers : ")
num1 = Console.ReadLine()
num2 = Console.ReadLine()
sum = num1 + num2
Console.WriteLine("num1 = {0} num 2 = {1}", num1, num2)
Console.WriteLine("Addition = {0}", sum)
```

Console.Read()

It only accept single character from user input and return its ASCII Code.

Note: Data type should be integer because it returns an integer value as ASCII.

Example :

```
Dim value As Integer
value = Console.Read( )
Console.WriteLine("Value Read from console is : ", value)
```

As input given is "**a**" and it return its ASCII value is 97

Console.ReadKey()

It reads that which key is pressed by user and returns its name.

Example :

```
Dim key As ConsoleKeyInfo
key = Console.ReadKey()
Console.WriteLine("You Pressed : {0}", key.Key)
```

Note: It is STRUCT Data type which is ConsoleKeyInfo.

If you press different Buttons from keyboard like Escape, Spacebar, Enter, R, P, Z, Minus and it return name.

Operators

Operator is a symbol that is used to perform various operations on variables. VB.NET has different types of Operators that help in performing logical and mathematical operations on data values.

Following are the different types of Operators available in VB.NET:

- **Arithmetic Operators**
- **Comparison Operators**
- **Assignment Operators**
- **Logical Operators**

Arithmetic Operators :

You can use arithmetic operators to perform various mathematical operations in VB.NET.

Assume variable **A = 2** and variable **B = 7**, then –

Operator	Description	Example
^	Raises one operand to the power of another	B^A will give 49
+	Adds two operands	A + B will give 9
-	Subtracts second operand from the first	A - B will give -5
*	Multiplies both operands	A * B will give 14
/	Divides one operand by another and returns a **floating point result**	B / A will give 3.5
\	Divides one operand by another and returns an **integer result**	B \ A will give 3
MOD	Modulus Operator and remainder of after an integer division	B MOD A will give 1

Example :

```
Module Module1
   Sub Main()
      Dim no1 As Integer = 11
      Dim no2As Integer = 5
      Dim no3 As Integer = 2
      Dim res1 As Integer
      Dim res2 As Single
      res1 = no1 + res2
```

```
        Console.WriteLine(" Result of 11 + 5 is {0} ", res1)
                res1 = no1 – no2
        Console.WriteLine(" Result of 11 - 5 is {0} ", res1)
                res1 = no1 * no2
        Console.WriteLine(" Result of 11 * 5 is {0} ", res1)
                res2 = no1 / no2
        Console.WriteLine(" Result of 11 / 5 is {0}", res2)
                res1 = no1 \ no2
        Console.WriteLine(" Result of 11 \ 5 is {0}", res1)
                res1 = no1 Mod no2
        Console.WriteLine(" Result of 11 MOD 5 is {0}", res1)
                res1 = no2^ no2
        Console.WriteLine(" Result of 5 ^ 5 is {0}", res1)
        Console.ReadLine()

    End Sub
End Module
```

Comparison Operators

These operators are used for making comparisons between variables.
Assume variable **A** holds 10 and variable **B** holds 20, then –

Operator	Description	Example
=	Checks if the values of two operands are equal or not; if yes, then condition becomes true.	(A = B) is not true.
<>	Checks if the values of two operands are equal or not; if values are not equal, then condition becomes true.	(A <> B) is true.
>	Checks if the value of left operand is greater than the value of right operand; if yes, then condition becomes true.	(A > B) is not true.
<	Checks if the value of left operand is less than the value of right operand; if yes, then condition becomes true.	(A < B) is true.
>=	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then condition becomes true.	(A >= B) is not true.
<=	Checks if the value of left operand is less	(A <= B) is true.

	than or equal to the value of right operand; if yes, then condition becomes true.	

Assignment Operators :

There are following assignment operators supported by VB.Net –

Operator	Description	Example
=	Simple assignment operator, Assigns values from right side operands to left side operand	C = A + B will assign value of A + B to C
+=	Add and assignment operator, It adds right operand to the left operand and assigns the result to left operand	C += A is equivalent to C = C + A
-=	Subtract and assignment operator, It subtracts right operand from the left operand and assigns the result to left operand	C -= A is equivalent to C = C - A
*=	Multiply and assignment operator, It multiplies right operand with the left operand and assigns the result to left operand	C *= A is equivalent to C = C * A
/=	Divide AND assignment operator, It divides left operand with the right operand and assigns the result to left operand (floating point division)	C /= A is equivalent to C = C / A
\=	Divide and assignment operator, It divides left operand with the right operand and assigns the result to left operand (Integer division)	C \= A is equivalent to C = C \A
^=	Exponentiation and assignment operator. It raises the left operand to the power of the right operand and assigns the result to left operand.	C^=A is equivalent to C = C ^ A
&=	Concatenates a String expression to a String variable or property and assigns the result to the variable or property.	Str1 &= Str2 is same as Str1 = Str1 & Str2

Logical Operators

Following table shows all the logical operators supported by VB.Net.

Assume variable A holds Boolean value True and variable B holds Boolean value False, then −

Operator	Description	Example
And	It is the logical as well as bitwise AND operator. **If both the operands are true,** then condition becomes true. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A And B) is False.
Or	It is the logical as well as bitwise OR operator. If **any of the two operands is true, then condition becomes true**. This operator does not perform short-circuiting, i.e., it evaluates both the expressions.	(A Or B) is True.
Not	It is the logical as well as bitwise NOT operator. Use to **reverses the logical state** of its operand. If a condition is true, then **Logical NOT** operator will make false.	Not(A And B) is True.
Xor	It is the logical as well as bitwise Logical Exclusive OR operator. **It returns True if both expressions are True or both expressions are False**; otherwise it returns False. This operator does not perform short-circuiting, it always evaluates both expressions and there is no short-circuiting counterpart of this operator.	A Xor B is True.
AndAlso	It is the **logical AND** operator. It works only on Boolean data. **It performs short-circuiting**.	(A AndAlso B) is False.
OrElse	It is the **logical OR** operator. It works only on Boolean data. It **performs short-circuiting.**	(A OrElse B) is True.

Example :

X=10, Y = 30

(X < 10) AndAlso (Y < 20)

Control Structure

1. **Decision Making Statements**
2. **Loops Statements**

Decision Making Statements :

1. If statement :
   It executes a set of code when a condition is true.
2. If...Then... Else statement :
   It selects one of two sets of lines to execute.
3. If...Then...ElseIf statement (Nested If statement) :
   It selects one of many sets of lines to execute.
4. Select Case statement :
   It select one of many sets of lines to execute.

1) If Then Statement ***If Then statement*** is a control structure which executes a set of code only when the given condition is True.

Syntax:

```
If [Condition] Then
   [Statements]
```

In the above syntax when the **Condition** is true then the **Statements** after **Then** are executed.

Example:

```
Dim x As Integer
x = 9
If (x Mod 2) = 0 Then
   Console.WriteLine("x is an even number")
End If
```

2) If Then Else Statement ***If Then Else*** statement is a control structure which executes different set of code statements when the given condition is true or false.

Syntax:

```
If [Condition] Then
   [Statements]
Else
   [Statements]
```

In the above syntax when the **Condition** is true, the **Statements** after **Then** are executed. If the condition is false then the statements after the **Else** part is executed.

Example:

```
Dim x As Integer
x = 9

If (x Mod 2) = 0 THEN
   Console.WriteLine("x is an even number")
Else
```

```
    Console.WriteLine("x is an odd number")
End If
```

3) Nested If Then Else Statement ***Nested If..Then..Else*** statement is used to check multiple conditions using if then else statements nested inside one another.

Syntax :

```
If condition1 Then
        Statement1
        statement2
ElseIf condition2 Then
        Statement3
        Statement4
Else
        Statement5
        Statement6
End If
```

Example:

```
Dim x As Integer
x = Console.ReadLine()
If x > 0 Then
    Console.WriteLine("It is positive.")
Elseif x<0
    Console.WriteLine("It is negative.")
Else
    Console.WriteLine("It is zeo.")
End If
```

Case Statement ***Select case statement*** is used when the expected results for a condition can be known previously so that different set of operations can be done based on each condition.

Syntax:

```
Select Case Expression
Case Expression1
    Statement1
Case Expression2
    Statement2
Case Expressionn
    Statementn
    ...
Case Else
    Statement
End Select
```

In the above syntax, the value of the **Expression** is checked with **Expression1..n** to check if the condition is true. If none of the conditions are matched the statements under the **Case Else** is executed.

Example - 1:

```
Dim grade As Char
    grade = "B"
    Select grade
       Case "A"
          Console.WriteLine("Excellent!")
       Case "B", "C"
          Console.WriteLine("Well done")
       Case "D"
          Console.WriteLine("You passed")
       Case "F"
          Console.WriteLine("Better try again")
       Case Else
          Console.WriteLine("Invalid grade")
    End Select
```

Example - 2:

```
Dim no As Integer
Console.WriteLine("Enter no. between 1-10")
no = Console.ReadLine()
Select Case no
   Case 1, 3, 5, 7, 9
      Console.WriteLine("Odd number")
   Case 2, 4, 6, 8, 10
      Console.WriteLine("Even number")
   Case Else
      Console.WriteLine("Invalid number-OutofRange")
End Select
```

Looping Statements : *A loop* statement allows us to execute a statement or group of statements multiple times.

1) For Next Loop Statement ***For Next Loop*** Statement executes a set of statements repeatedly in a loop for the given initial, final value range with the specified step by step increment or decrement value.

Syntax:

```
For counter = start To end [Step]
     [Statement]
Next [counter]
```

In the above syntax the **Counter** is range of values specified using the **Start**, **End** parameters. The **Step** specifies step increment or decrement value of the counter for which the statements are executed.

Example:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
Dim i As Integer
Dim j As Integer
j = 0
For i = 1 To 10 Step 1
j = j + 1
MsgBox("Value of j is::" & j)
Next i
End Sub
```

Description:

In the above For Next Loop example the counter value of i is set to be in the range of 1 to 10 and is incremented by 1. The value of j is increased by 1 for 10 times as the loop is repeated.

For Each...Next *For Each...Next* repeats a group of statements for each element in a collection. For-Each, requires a collection. This loop is used for accessing and manipulating all elements in an array or a VB.Net collection.

Syntax :

```
For Each element [As datatype ] In group
  statements
 Next [ element ]
```

Example : Print elements of an array

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load

    Dim arr() As Integer = {1, 3, 5, 7, 9}
    Dim x As Integer

   For Each x In arr            'displaying the values
      MsgBox("Value of x is::" & x)
    Next
  End Sub
```

Description:

In the above For Each….Next Loop example the value of array is stored in variable x and displayed in message box. The loop is repeated until all the values are fetched. Total number of iteration of For Each….Next loop is equal to the total number of elements in array. In above example For Each….Next loop executes 5 times as there are 5 values in an array.

3) Do While Loop Statement ***Do While Loop*** Statement is used to execute a set of statements only if the condition is satisfied. But the loop get executed once for a false condition once before exiting the loop. This is also known as **Entry Controlled** loop.

Syntax:

```
Do While [Condition]
[Statements]
Loop
```

In the above syntax the **Statements** are executed till the **Condition** remains true.

Example:

```
Private Sub Form1_Load(ByVal sender As System.Object,ByVal e As
System.EventArgs) Handles MyBase.Load
Dim a As Integer
a = 1
Do While a < 100
a = a * 2
MsgBox("Product is::" & a)
Loop
End Sub
```

Description:

In the above Do While Loop example the loop is continued after the value 64 to display 128 which is false according to the given condition and then the loop exits.

4) While Wend Statement ***While Wend Statement*** is a looping statement where a condition is checked first, if it is true a set of statements are executed. The condition can also become false atleast once while this statement is used.

Syntax:

```
While condition
  [statements]
Wend
```

In the above syntax the **Statements** are executed when the **Condition** is true. The condition may also become false while the statements are executed.

Example:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
Dim n As Integer
n = 1
     While n <= 1
        n = n + 1
        MsgBox("First incremented value is:" & n)
     End While
End Sub
```

Description:

In the above example, the value of **n** is 2 in the loop, which is false according to the condition but the loop is continued until the condition is checked. Thus the while wend statement can be used.

5) While Statement *Do Loop While* Statement executes a set of statements and checks the condition, this is repeated until the condition is true. .It is also known as an **Exit Control** loop

Syntax:

```
    Do
        [Statements]
    Loop While [Condition]
```

In the above syntax the **Statements** are executed first then the **Condition** is checked to find if it is true.

Example:

```
    Private Sub Form1_Load(ByVal sender As System.Object,
    ByVal e As System.EventArgs) Handles MyBase.Load
    Dim cnt As Integer
        Do
            cnt = 10
            MsgBox("Value of cnt is::" & cnt)
        Loop While cnt <= 9
    End Sub
```

Description:

In the above Do Loop While example, a messa

6) Do Loop Until Statement *Do Loop Until* Statement executes a set of statements until a condition becomes false, this is an infinite loop might have to be terminated using **Ctrl + Break** .

Syntax:

```
    Do
        [Statements]
    Loop Until [Condition]
```

In the above syntax the **Statements** are executed until the **Condition** becomes false.

Example:

```
Private Sub Form1_Load(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles MyBase.Load
Dim X As String
    Do
        X = InputBox$("Correct Password Please")
    Loop Until X = "Ranger"
End Sub
```

Description:

In the above Do Until Loop example, a input box is displayed until the correct password is typed.

Windows Applications

Form Control :

Properties – Following properties can be set or read during application execution.

Sr. No	Properties	Description
1	**BackColor**	Sets the form background color.
2	**FormBorderStyle**	The BorderStyle property determines the style of the form's border and the appearance of the form − ⟩ **None** − Borderless window that can't be resized. ⟩ **Sizable** − This is default value and will be used for resizable window that's used for displaying regular forms. ⟩ **Fixed3D** − Window with a visible border, "raised" relative to the main area. In this case, windows can't be resized. ⟩ **FixedDialog** − A fixed window, used to create dialog boxes. ⟩ **FixedSingle** − A fixed window with a single line border. ⟩ **FixedToolWindow** − A fixed window with a Close button only. It looks like the toolbar displayed by the drawing and imaging applications. ⟩ **SizableToolWindow** − Same as the FixedToolWindow but resizable. In addition, its caption font is smaller than the usual.
3	**ControlBox**	By default, this property is True and you can set it to False to hide the icon and disable the Control menu.
4	**Height**	This is the height of the Form in pixels.
5	**MinimizeBox**	By default, this property is True and you can set it to False to hide the Minimize button on the title bar.

6	**MaximizeBox**	By default, this property is True and you can set it to False to hide the Maximize button on the title bar.
7	**MinimumSize**	This specifies the minimum height and width of the window you can minimize.
8	**MaximumSize**	This specifies the maximum height and width of the window you maximize.
9	**Name**	This is the actual name of the form.
10	**StartPosition**	This property determines the initial position of the form when it's first displayed. It will have any of the following values − ⌡ **CenterParent** − The form is centered in the area of its parent form. ⌡ **CenterScreen** − The form is centered on the monitor. ⌡ **Manual** − The location and size of the form will determine its starting position. ⌡ **WindowsDefaultBounds** − The form is positioned at the default location and size determined by Windows. ⌡ **WindowsDefaultLocation** − The form is positioned at the Windows default location and has the dimensions you've set at design time.
11	**Text**	The text, which will appear at the title bar of the form.
12	**TopMost**	This property is a True/False value that lets you specify whether the form will remain on top of all other forms in your application. Its default property is False.
13	**Width**	This is the width of the form in pixel.
14	**AcceptButton**	The button that's automatically activated when you press Enter, no matter which control has the focus at the time. Usually the OK button on a form is set as AcceptButton for a form.
15	**CancelButton**	The button that's automatically activated when you hit the Esc key.

		Usually, the Cancel button on a form is set as CancelButton for a form.
16	**AutoScroll**	This Boolean property indicates whether scroll bars will be automatically attached to the form if it is resized to a point that not all its controls are visible.

Form Events:

Following table lists down various important events related to a form −

Sr. No.	Event	Description
1	**Click**	Occurs when the form is clicked.
2	**Closed**	Occurs before the form is closed.
3	**DoubleClick**	Occurs when the form control is double-clicked.
4	**GotFocus**	Occurs when the form control receives focus.
5	**LostFocus**	Occurs when the form loses focus.
6	**KeyDown**	Occurs when a key is pressed while the form has focus.
7	**KeyPress**	Occurs when a key is pressed while the form has focus.
8	**KeyUp**	Occurs when a key is released while the form has focus.
9	**Load**	Occurs before a form is displayed for the first time.
10	**MouseDown**	Occurs when the mouse pointer is over the form and a mouse button is pressed.
11	**MouseEnter**	Occurs when the mouse pointer enters the form.
12	**MouseLeave**	Occurs when the mouse pointer leaves the form.
13	**MouseMove**	Occurs when the mouse pointer is moved over the form.

14	**MouseUp**	Occurs when the mouse pointer is over the form and a mouse button is released.

Examples on Form Event :

```
Private Sub Form1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Click

            MsgBox("Form Click Event raised")

End Sub

Private Sub Form1_DoubleClick(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Me.DoubleClick
        MsgBox("Form Double Click Event raised")
End Sub

Private Sub Form1_FormClosed(ByVal sender As Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs) Handles Me.FormClosed
        MsgBox("Form Closed Event raised")

End Sub
```

Button:

Properties – Following properties are commonly used properties of the Button control −.

Sr. No.	Property & Description
1	**AutoSizeMode** Gets or sets the mode by which the Button automatically resizes itself
2	**BackColor** Gets or sets the background color of the control.
3	**ForeColor** Gets or sets the foreground color of the control.
4	**Image**

	Gets or sets the image that is displayed on a button control.
5	**Location** Gets or sets the coordinates of the upper-left corner of the control relative to the upper-left corner of its container.
6	**TabIndex** Gets or sets the tab order of the control within its container.
7	**Text** Gets or sets the text associated with this control.

TextBox:

The TextBox Control allows you to enter text on your form during runtime.

Properties – Following properties are commonly used properties of the TextBox control –

Sr. No.	**Property & Description**
1	**Text** Gets or sets the current text in the TextBox.
2	**Font** Gets or sets the font of the text displayed by the control.
3	**ForeColor** Gets or sets the foreground color of the control.
4	**Enabled** For enabling the textbox control
5	**Multiline** Gets or sets a value indicating whether this is a multiline TextBox control.
6	**PasswordChar** Gets or sets the character used to mask characters of a password in a single-line TextBox control.

7	**ReadOnly** Gets or sets a value indicating whether text in the text box is read-only.
8	**ScrollBars** Gets or sets which scroll bars should appear in a multiline TextBox control. This property has values − None, Horizontal, Vertical, Both
9	**TabIndex** Gets or sets the tab order of the control within its container.
10	**TextAlign** Gets or sets how text is aligned in a TextBox control. This property has values – Left, Right, Center
11	**TextLength** Gets the length of text in the control.
12	**WordWrap** Indicates whether a multiline text box control automatically wraps words to the beginning of the next line when necessary.

The Methods of the TextBox Control :

The following are some of the commonly used methods of the TextBox control –

Sr.No.	Method Name & Description
1	**AppendText** Appends text to the current text of a text box.
2	**Clear** Clears all text from the text box control.
3	**Copy** Copies the current selection in the text box to the **Clipboard**.
4	**Cut** Moves the current selection in the text box to the **Clipboard**.
5	**Paste** Replaces the current selection in the text box with the contents of the **Clipboard**.
6	**Undo**

	Undoes the last edit operation in the text box.

- SelectionStart - for setting or getting the starting point for the TextBox Control.
- SelectionLength - for setting or getting the number of characters that have been selected in the TextBox Control.
- SelectedText - returns the TextBox Control that is currently selected.

Events of the TextBox Control :

The following are some of the commonly used events of the Text control –

Sr.No.	Event & Description
1	**Click** Occurs when the control is clicked.
2	**DoubleClick** Occurs when the control is double-clicked.
3	**TextChanged** Occurs when the Text within the textbox changes.

Creating TextBox at Runtime:

```
    Dim txt1, txt2 As New TextBox    //Declare variables of type

    txt1.Name = "txtNo1"        //Set the properties
  txt1.Left = 100
  txt1.Top = 50
  txt1.Visible = True

  Me.Controls.Add(txt1)         //Add the control on form

  txt2.Name = "txtNo2"
  txt2.Left = 100
  txt2.Top = 150
  txt2.Visible = True
  Me.Controls.Add(txt2)
```

Public Class Form1

```
    Dim txtNo1 As New TextBox
    Dim btn1 As New Button()

Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button2.Click

        txtNo1.Left = 100
        txtNo1.Top = 200
        txtNo1.Visible = True
        Me.Controls.Add(txtNo1)
    End Sub

Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Dim btn1 As Button = New Button()

        btn1.Parent = Me
        btn1.Name = "btn1"
        btn1.Top = 10
        btn1.Text = "Btn1"
        Me.Controls.Add(btn1)

        'adding handler for click event
        AddHandler btn1.Click, AddressOf HandleDynamicButtonClick

    End Sub

    Private Sub HandleDynamicButtonClick(ByVal sender As Object, ByVal e As
EventArgs)
        MsgBox(txtNo1.Text)
    End Sub

End Class
```

Label :

The Label Control allows you to enter text on your form during runtime.

Properties – Following properties are commonly used properties of the Label control –

Sr.No.	Property & Description
1	**Autosize**

	Gets or sets a value specifying if the control should be automatically resized to display all its contents.
2	**BorderStyle** Gets or sets the border style for the control.
3	**FlatStyle** Gets or sets the flat style appearance of the Label control
4	**Font** Gets or sets the font of the text displayed by the control.
5	**FontHeight** Gets or sets the height of the font of the control.
6	**ForeColor** Gets or sets the foreground color of the control.
7	**PreferredHeight** Gets the preferred height of the control.
8	**PreferredWidth** Gets the preferred width of the control.
10	**Text** Gets or sets the text associated with this control.
11	**TextAlign** Gets or sets the alignment of text in the label.

Anchoring and Docking Controls to a Form :

Anchoring Control:

- The Anchor property lets you attach one or more edges of the control to corresponding edges of the form.
- The anchored edges of the control maintain the same distance from the corresponding edges of the form.
- Place a TextBox control on a new form, set its MultiLine property to True, and then open the control's Anchor property in the Properties window.

Docking Control:

- Dock property, which determines how a control will dock on the form. The default value of this property is None.

CheckBox :

The CheckBox control allows the user to set true/false or yes/no type options. The user can select or deselect it. When a check box is selected it has the value True, and when it is cleared, it holds the value False.

Properties of the CheckBox Control :

The following are some of the commonly used properties of the CheckBox control −

Sr. No.	Property & Description
1	**Appearance** Gets or sets a value determining the appearance of the check box.
2	**CheckAlign** Gets or sets the horizontal and vertical alignment of the check mark on the check box.
3	**Checked** Gets or sets a value indicating whether the check box is selected.
4	**CheckState** Gets or sets the state of a check box.
5	**Text** Gets or sets the caption of a check box.
6	**ThreeState** Gets or sets a value indicating whether or not a check box should allow three check states rather than two.

Events of the CheckBox Control :

The following are some of the commonly used events of the CheckBox control −

Sr. No.	Event & Description
1	**AppearanceChanged** Occurs when the value of the Appearance property of the check box is changed.
2	**CheckedChanged** Occurs when the value of the Checked property of the CheckBox

	control is changed.
3	**CheckStateChanged** Occurs when the value of the CheckState property of the CheckBox control is changed.

Following Example shows use of CheckState Property and CheckStateChanged Event

```
Private Sub CheckBox1_CheckStateChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles CheckBox1.CheckStateChanged

         If CheckBox1.CheckState = CheckState.Checked Then
            MsgBox("Yes")
        End If

        If CheckBox1.CheckState = CheckState.Indeterminate Then
            MsgBox("May Be")
        End If

        If CheckBox1.CheckState = CheckState.Unchecked Then
            MsgBox("No")
        End If

End Sub
```

Example 2:

Add four check boxes on form. The check boxes will allow the users to choose the programming languages. When the user clicks the Submit button, he/she gets an appropriate message.

Solution:

```
If CheckBox1.Checked = True Then
        str &= CheckBox1.Text
        str &= " "
End If
```

RadioButton :

The RadioButton control is used to provide a set of mutually exclusive options.
The user can select one radio button in a group.
If you need to place more than one group of radio buttons in the same form, you should place them in different container controls like a GroupBox control.

Properties :
The following are some of the commonly used properties of the RadioButton control –

Sr. No.	Property & Description
1	**Appearance** Gets or sets a value determining the appearance of the radio button.
2	**CheckAlign** Gets or sets the location of the check box portion of the radio button.
3	**Checked** Gets or sets a value indicating whether the control is checked.
4	**Text** Gets or sets the caption for a radio button.

ListBox:

The ListBox represents a Windows control to display a list of items to a user. A user can select an item from the list. It allows the programmer to add items at design time by using the properties window or at the runtime.
Properties:
The following are some of the commonly used properties of the ListBox control –

Sr. No.	Property & Description
1	**AllowSelection :** Gets a value indicating whether the ListBox currently enables selection of list items.
2	**HorizontalScrollBar :**Gets or sets the value indicating whether a horizontal scrollbar is displayed in the list box.

3	**Items :** Gets the items of the list box.
4	**MultiColumn :** Gets or sets a value indicating whether the list box supports multiple columns.
5	**ScrollAlwaysVisible :**Gets or sets a value indicating whether the vertical scroll bar is shown at all times.
6	**SelectedIndex :**Gets or sets the zero-based index of the currently selected item in a list box.
7	**SelectedIndices :** Gets a collection that contains the zero-based indexes of all currently selected items in the list box.
8	**SelectedItem:** Gets or sets the currently selected item in the list box.
9	**SelectedItems :**Gets a collection containing the currently selected items in the list box.
10	**SelectionMode** Gets or sets the method in which items are selected in the list box. This property has values − ⌡ None – No item can be selected ⌡ One – Only one item can be selected ⌡ MultiSimple – Multiple items can be selected ⌡ MultiExtended – Multiple items can be selected with the help of Shift Ctrl and arrow key
11	**Sorted** Gets or sets a value indicating whether the items in the list box are sorted alphabetically.
12	**Text** Gets or searches for the text of the currently selected item in the list box.
13	**TopIndex** Gets or sets the index of the first visible item of a list box.

Adding Items to ListBox at Runtime :

Add( ) method : It is used to add items in ListBox at Runtime
Syntax :

```
ListBoxName.Items.Add("DataItme")
```

Example :

```
lstCity.Items.Add("Amravati")
```

Insert( ) method : It is used to insert items at particular position in ListBox at Runtime
Syntax :
ListBoxName.Items.Insert(Index, “DataItme”)
Example :
lstCity.Items.Insert(2,“Amravati”)

Count( ) method :
It returns the total number of items in the ListBox

Syntax :
Variable name = ListBoxName.Items.Count
Example :
n = lstCity.Items.Count

Clear( ) method : It is used to clear the ListBox
Syntax :
ListBoxName.Items.Clear( )
Example :
lstCity.Items.Clear( )

Removing Items from ListBox at Runtime :

Remove( ) and RemoveAt( ) methods :
These two methods are used to remove items from ListBox at Runtime
i) Remove( ) : Removes item by passing corresponding item.
Syntax :
ListBoxName.Items.Remove(ItemName)
Example :
i) lstCity.Items.Remove(“Pune”) : It removes city Pune from ListBox lstCity
ii) lstCity.Items.Remove(lstCity.SelectedItem) : It removes city Pune from ListBox lstCity
ii) RemoveAt( ) : Removes item by passing corresponding index.
Syntax :
ListBoxName.Items.RemoveAt(Index)
Example :
i) lstCity.Items.RemoveAt(2) : It removes city placed at index 2 from ListBox lstCity
ii) lstCity.Items.RemoveAt(lstCity.SeletedIndex) : It removes currently selected City from ListBox lstCity

Events of the ListBox Control :

The following are some of the commonly used events of the ListBox control –

Sr.No.	Event & Description
1	**Click** Occurs when a list box is selected.
2	**SelectedIndexChanged** Occurs when the SelectedIndex property of a list box is changed.

ComboBox Control :

- The ComboBox control is used to display a drop-down list of various items.
- It is a **combination of a text box** in which the user enters an item and a drop-down list from which the user selects an item.

Properties of the ComboBox Control :

Sr.No.	Property & Description
1	**DropDownStyle :** It specifies the appearance and behavior of the ComboBox Values : 1) Simple 2) DropDown 3) DropDownList
2	**ItemHeight** **Gets** or sets the height of an item in the combo box.
3	**Items** Gets an object representing the collection of the items contained in this ComboBox.
4	**MaxDropDownItems** Gets or sets the maximum number of items to be displayed in the drop-down part of the combo box. To apply this first set the property IntegralHeight = False
5	**MaxLength** Gets or sets the maximum number of characters a user can enter in the editable area of the combo box.
6	**SelectedIndex** Gets or sets the index specifying the currently selected item.
7	**SelectedItem** Gets or sets currently selected item in the ComboBox.

8	**SelectedText** Gets or sets the text that is selected in the editable portion of a ComboBox.
9	**Sorted** Gets or sets a value indicating whether the items in the combo box are sorted.
10	**Text** Gets or sets the text associated with this control.

RichTextBox :

The RichTextBox control allows the user to display, enter, and edit text while also providing more advanced formatting features than the conventional TextBox control.

Methods :

1] SaveFile : The SaveFile method saves the contents of the control to a disk file.

Syntax : RichTextBox1.SaveFile(path, filetype)

Example :

2] LoadFile : The LoadFile method loads a text or RTF file to the control.

Syntax : RichTextBox1.LoadFile(path, filetype)

Example :

Properties :

Following properties are used to read or change the alignment of one or more paragraphs:

- SelectionIndent : Sets / Gets the left indentation of paragraph.
- SelectionRightIndent : Sets / Gets the right indentation of paragraph.
- SelectionHangingIndent : Indents all other lines of paragraph with respect to SelectionIndent.

Example:

```
RichTextBox1.SelectionIndent            = 20
RichTexBox1.SelectionHangingIndent = -25
RichTextBox1.RightIndent                    = 10
```

Formatting Text of RichTextBox :

```
Dim fd As New FontDialog
fd.ShowDialog()
TextBox1.Font = fd.Font
```

MsgBox ( ) Function

The objective of MsgBox is to produce a pop-up message box and prompt the user to click on a command button before he /she can continues. This format is as follows:

myMsg=MsgBox(Prompt, Style Value, Title)

The first argument, Prompt, will display the message in the message box. The Style Value will determine what type of command buttons appear on the message box, please refer to the below Table for types of command button displayed. The Title argument will display the title of the message board.

Style Value	Named Constant	Buttons Displayed
0	vbOkOnly	Ok button
1	vbOkCancel	Ok and Cancel buttons
2	vbAbortRetryIgnore	Abort, Retry and Ignore buttons.
3	vbYesNoCancel	Yes, No and Cancel buttons
4	vbYesNo	Yes and No buttons
5	vbRetryCancel	Retry and Cancel buttons

We can use named constants in place of integers for the second argument to make the programs more readable.

For example:

myMsg=MsgBox("Click OK to Proceed", 1, "Startup Menu")

and

myMsg=Msg("Click OK to Proceed". vbOkCancel,"Startup Menu")

are the same.

myMsg is a variable that holds values that are returned by the MsgBox ( ) function. The values are determined by the type of buttons being clicked by the users. It has to be declared as Integer data type in the procedure or in the general declaration section. Below table shows the values, the corresponding named constant and buttons.

Value	Named Constant	Button Clicked
1	vbOk	Ok button
2	vbCancel	Cancel button
3	vbAbort	Abort button
4	vbRetry	Retry button
5	vbIgnore	Ignore button

6	vbYes	Yes button
7	vbNo	No button

Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim testmsg As Integer
        testmsg = MsgBox("Click to test", 1, "Test message")
        If testmsg = 1 Then
                MessageBox.Show("You have clicked the OK button")
        Else
                MessageBox.Show("You have clicked the Cancel button")
        End If
End Sub
```

You can add an icon besides the message. There are four types of icons available as shown in Table

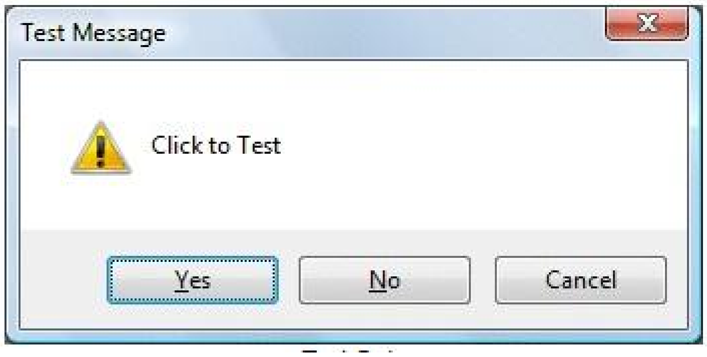
Value	Named Constant	Icon
16	vbCritical	✖
3	vbQuestion	❓
48	vbExclamation	⚠
64	vbInformation	ℹ

Example:

```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim testMsg As Integer
        testMsg = MsgBox("Click to Test", vbYesNoCancel + vbExclamation, "Test
Message")
        If testMsg = 6 Then
                MessageBox.Show("You have clicked the yes button")
        ElseIf testMsg = 7 Then
                MessageBox.Show("You have clicked the NO button")
        Else
                MessageBox.Show("You have clicked the Cancel button")
        End If
```

End Sub

The first argument, Prompt, will display the message



The InputBox( ) Function

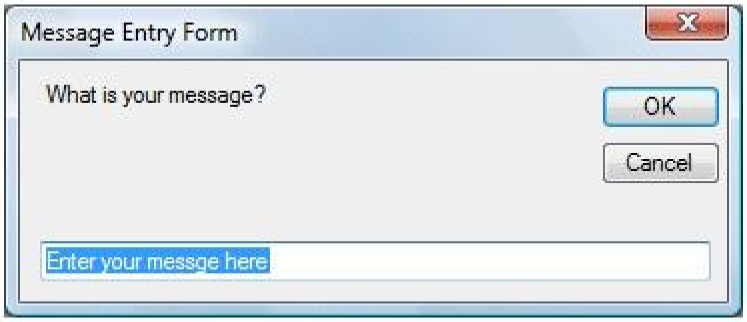
An InputBox( ) function will display a message box where the user can enter a value or a message in the form of text.
Format of InputBox Function:
myMessage=InputBox(Prompt, Title, default_text, x-position, y-position)

myMessage is a variant data type but typically it is declared as string, which accept the message input by the users.
The arguments are explained as follows:

Prompt - the message displayed normally as a question asked.
Title - The title of the Input Box.
default-text - The default text that appears in the input field where users can use it as his intended input or he may change to the message he wish to enter.
x-position and y-position - the position or tthe coordinates of the input box.
InputBox(Prompt, Title, default_text, x-position, y-position)
The parameters remain the same.

Example:
```
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        Dim userMsg As String
        userMsg = Microsoft.VisualBasic.InputBox("What is your message?", "Message Entry
Form", "Enter your messge here", 500, 700)
        If userMsg <> "" Then
                MessageBox.Show(userMsg)
        Else
                MessageBox.Show("No Message")
        End If
End Sub
```

The InputBox will appear as shown in the figure below when you press the command button.



Predefined Dialog Box in VB.Net:

- There are many built-in dialog boxes to be used in Windows forms for various tasks like opening and saving files, printing a page, providing choices for colors, fonts, page setup, etc., to the user of an application.
- The ShowDialog method is used to display all the dialog box controls at run-time.
- The ShowDialog method returns a value of the type of DialogResult enumeration. The values of DialogResult enumeration are −
  - Abort − returns DialogResult. Abort value, when user clicks an Abort button.
  - Cancel − returns DialogResult. Cancel, when user clicks a Cancel button.
  - Ignore − returns DialogResult. Ignore, when user clicks an Ignore button.
  - No − returns DialogResult. No, when user clicks a No button.
  - None − returns nothing and the dialog box continues running.
  - OK − returns DialogResult. OK, when user clicks an OK button
  - Retry − returns DialogResult. Retry , when user clicks an Retry button
  - Yes − returns DialogResult. Yes, when user clicks an Yes button

1] ColorDialog : It represents a common dialog box that displays available colors along with controls that enable the user to define custom colors.

Example :

```
Dim ColorDialog1 As New ColorDialog
Private Sub Button1_Click(sender As Object, e As EventArgs) Handles
Button1.Click
  If ColorDialog1.ShowDialog < > Windows.Forms.DialogResult.Cancel Then
    Label1.ForeColor = ColorDialog1.Color
  End If
End Sub
```

2] FontDialog : It prompts the user to choose a font from among those installed on the local computer and lets the user select the font, font size, and color.

Example :

```
Dim FontDialog1 As New FontDialog
If FontDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
    RichTextBox1.ForeColor = FontDialog1.Color
    RichTextBox1.Font = FontDialog1.Font
  End If
```

3] OpenFileDialog : It prompts the user to open a file and allows the user to select a file to open.

Example:

```
Dim OpenFileDialog1 As New OpenFileDialog
 If OpenFileDialog1.ShowDialog <> Windows.Forms.DialogResult.Cancel Then
     PictureBox1.Image = Image.FromFile(OpenFileDialog1.FileName)
   End If
```

4] SaveFileDialog : It prompts the user to select a location for saving a file and allows the user to specify the name of the file to save data.

Example :

```
SaveFileDialog1.Filter = "TXT Files (*.txt*)|*.txt"
    If SaveFileDialog1.ShowDialog = Windows.Forms.DialogResult.OK _
          Then
      My.Computer.FileSystem.WriteAllText _
      (SaveFileDialog1.FileName, RichTextBox1.Text, True)
    End If
```

5] PrintDialog : It lets the user to print documents by selecting a printer and choosing which sections of the document to print from a Windows Forms application.

Example :

```
PrintDialog1.Document = PrintDocument1
  PrintDialog1.PrinterSettings = PrintDocument1.PrinterSettings
  PrintDialog1.AllowSomePages = True

  If PrintDialog1.ShowDialog = DialogResult.OK Then
    PrintDocument1.PrinterSettings = PrintDialog1.PrinterSettings
    PrintDocument1.Print()
  End If
```

Assignment 1:

Set A

1. Write a Vb.net program to accept string and count the number of words and display the count.
2. Write VB.Net program to covert decimal number into binary, hexadecimal, and octal number.
3. Write a Vb.net program to print second highest value in an array.

Set B

1. Write a Vb.net program to find Prime Numbers between 1 to 500.
2. Write a Vb.net program to find Spy Numbers between 100 to 500.
3. Write a Vb.net program to check whether entered string is palindrome or not.

Set C

1. Write a Vb.net program to check whether entered number is Spy Number or not Using Console Application.
   [Note : A positive integer is called a spy number if the sum and product of its digits are equal.]

Assignment Evaluation

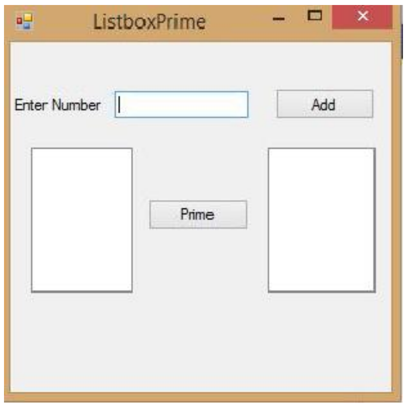
0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 2

Set A

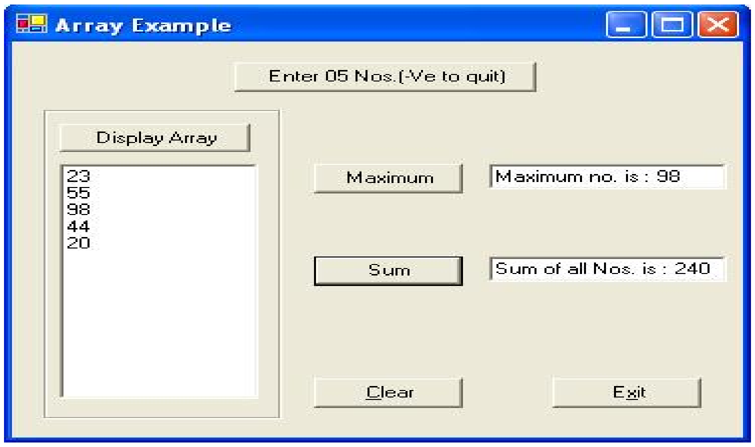
1. Write a Vb.net program to design the following form, accept the numbers through textbox and add them into the ListBoxe1 by clicking on Add button. When user click on Prime button then all the prime numbers from ListBox1 should get added into ListBox2.



2. Write a Vb.net program for blinking an image.
3. Design a Vb.net form to pick a date from DateTimePicker Control and display day, month and year in separate text boxes.

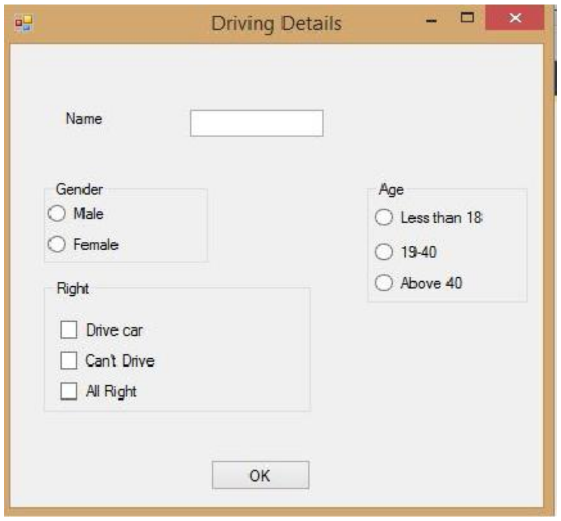
Set B

1. Write VB.Net program to add three text boxes at runtime. Allow user to enter values and then display Maximum and Minimum value using message box after click on command button.
2. Write VB .Net program to take three text boxes and two buttons on the form. Enter different strings in first and second textbox. On clicking the First command button, concatenation of two strings should be displayed in third text box and on clicking second command button, reverse of string should display in third text box.
3. Write VB.Net Program to Design following form to store numbers in single dimensional array and to find Maximum no & sum of all numbers in array.



Set C

1. Write a Vb.net program to design the following form, allow the user to select radio buttons from Gender and Age Panel. After Selection appropriate CheckBox from Right Panel should be selected automatically. Display appropriate message into the MessageBox by clicking on Ok button.



2. Write VB.net program to design a calculator with proper validation.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Assignment 3

1. Write a Vb.net program to check whether entered number is Armstrong or.
2. Write a Vb.net program to accept a character from keyboard and check whether it is vowel or not. Also display the case of that character.
3. Write a Vb.net program to accept number from user into the TextBox. Calculate the square root of that number also convert the entered number into binary number and display result into the Message Box
4. Write a Vb.net program to accept a number from an user through InputBox and display its multiplication table into the ListBox.
5. Write a Vb.Net program to move the Text "Arts Commerce Science College" continuously from Left to Right.
6. Write a VB.NET program to do the following operations on RichTextBox values
   i) Font Style
   ii) Font Color
   iii) Save
   iv) Open

7. Write a Vb.net program to design screen to accept the details from the user. Clicking on Submit button Net Salary should be calculated and displayed into the TextBox. Display the MessageBox informing the Name and Net Salary of employee.

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**

3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

ASSIGNMENT - C: C#.NET

C# is a general-purpose, object-oriented, modern programming language, which is developed by Microsoft and is a part of ".Net framework".

C# programming supports multi-paradigm, strong typing, lexically scope, imperative, declarative, functional, etc.

Most of the syntaxes of the C# programming language are similar to the C and C++ programming languages.

we created a C# file called Program1.cs, and we used the following code to print "Hello World" to the screen:

```
using System;

namespace HelloWorld
{
  class Program1
  {
    static void Main(string[] args)
    {
      Console.WriteLine("Hello World!");
    }
  }
}
```

```
Hello World!
```

Example explained

Line 1: using System means that we can use classes from the **System** namespace.

Line 2: namespace is used to organize your code, and it is a container for classes and other namespaces.

Line 3: The curly braces **{}** marks the beginning and the end of a block of code.

Line 4: class is a container for data and methods, which brings functionality to your program. Every line of code that runs in C# must be inside a class. In our example, we named the class Program1.

Line 6: The **Main** method. Any code inside its curly brackets {} will be executed. It is an entry point method, Program execution is stared from main method.

Line 7: Console is a class of the **System** namespace, which has a **WriteLine()** method that is used to output/print text. In our example it will output "Hello World!".

Note:

1.If you omit the **using System** line, you would have to write **System.Console.WriteLine()** to print/output text.

2. Every C# statement ends with a semicolon (;)

3**.** C# is case-sensitive: "MyClass" and "myclass" has different meaning.

C# Comments: Comments are the statements ignored by the compiler.

Types of C# compiler:

1. **Single-line Comments**

Single-line comments start with two forward slashes (//).
Any text between // and the end of the line is ignored by C# compiler (will not be executed).
Example

```
// This is a comment
Console.WriteLine("Hello World!");
```

2. **C# Multi-line Comments**

Multi-line comments start with /* and ends with */.
Any text between /* and */ will be ignored by C# compiler.
Example

```
/* This is the multi line comment and
the output of this example is Hello World
on the screen. */
Console.WriteLine("Hello World!");
```

C# Variables
Variables are the identifier, used for the storing values.

Syntax:
```
datatype variableName = value;
```

Example:
```
string name = "Vikas";
Console.WriteLine(name);
```

Constants
Constant is a variable that can't be change its value during the program once initialized.
Examlple :	const doublePI = 3.142;

C# Data Types

Data Type tells us that which kind of value store into a variable and how many bytes allocated for that variable.

Data Type	Size	Description
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
Double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits
Bool	1 bit	Stores true or false values
Char	2 bytes	Stores a single character/letter, surrounded by single quotes
String	2 bytes per character	Stores a sequence

Note: A floating point number can also be a scientific number with an "e" to indicate the power of 10:

```
float f1 = 35e3F;
double d1 = 12E4D;
Console.WriteLine(f1);
Console.WriteLine(d1);
```

```
Output
35000
120000
```

C# Type Casting

Type Casting is the converting the value of variable into one data type to another data type.

In C#, there are two types of casting:

- **Implicit Casting** (automatically) - converting a smaller type to a larger type size
  char -> int -> long -> float -> double
  Example:

```
int intNum = 10;
double doubleNum = intNum;      // Automatic casting: int to double
Console.WriteLine(intNum);     // Outputs 10
Console.WriteLine(doubleNum);   // Outputs 10
```

- **Explicit Casting** (manually) - converting a larger type to a smaller size type
  double -> float -> long -> int -> char

  Example:

```
    double doubleNum= 13.22;
    int intNum= (int) doubleNum;    // Manual casting: double to int
    Console.WriteLine(doubleNum);   // Outputs 13.22
    Console.WriteLine(intNum);      // Outputs 13
```

Type Conversion Methods

There are some built-in methods which converts data types explicitly, such as Convert.ToBoolean, Convert.ToDouble, Convert.ToString, Convert.ToInt32 (int) and Convert.ToInt64 (long):

```
int intNum = 10;
double doubleNum= 13.22;
bool boolNum = true;
Console.WriteLine(Convert.ToString(intNum));    // convert int to string
Console.WriteLine(Convert.ToDouble(intNum));    // convert int to double
Console.WriteLine(Convert.ToInt32(doubleNum));  // convert double to int
Console.WriteLine(Convert.ToString(boolNum));   // convert bool to string
```

Get User Input
Console.ReadLine() is the method that read the line of text from keyboard.
Example:

```
using System;

namespace MyApplication
{
  class Program2
  {
    static void Main(string[] args)
    {
Console.WriteLine("Enter your full name:");
string fullName = Console.ReadLine();
Console.WriteLine("Enter your age:");
int age = Console.ReadLine();
Console.WriteLine("Your Full Name is: " + fullName+” and Age is:”+age);

}
  }
}
```

C# Operators

Operators are used to perform operations on variables and values.

Operator	Name	Example
Arithmetic Operators		
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
++	Increment	x++
--	Decrement	x--
Comparison Operators		
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y
Logical Operators		
&&	Logical and	x < 5 && x < 10
||	Logical or	x < 5 || x < 4
!	Logical not	!(x < 5 && x < 10)
Assignment Operator		
=	Assigning the RHS value to LHS	x = 5
Short hand assignment operator		
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3

>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Control Structure:

There are three types of control structures in c# these are:

1. **Decision Making Control Statements:**

i. **if Statement:**When condition is true then statements within if block are executed.
Syntax:

```
if (condition)
{
  // block of code to be executed if the condition is True
}
```

```
using System;

namespace MyApplication {
  class Program3
  {
    static void Main(string[] args)
    {
      if (20>10)
      {
        Console.WriteLine("20 is greater than 10");
      }
    }
  }
}
```

ii. **if-else statement:** if condition is true then statements within if block are executed otherwise statements within else block are executed.
Syntax:

```
if (condition)
{
  // block of code to be executed if the condition is True
}
else
{
  // block of code to be executed if the condition is False
}
```

```
using System;

namespace MyApplication {
  class Program4
  {
    static void Main(string[] args)
    {
            Console.Write("Enter number:");
            int num = Convert.ToInt32(Console.ReadLine());
           if (num>=0)
 {
Console.WriteLine(num+" is +ve");
}
        else
           {
Console.WriteLine(num+" is -ve");
}
    }
  }
}
```

iii. **Nested if statement:**if with ladder or if within if are called nested if statement.

Syntax:

```
if (condition1)
{
  // block of code to be executed if condition1 is True
}
else if (condition2)
{
  // block of code to be executed if the condition1 is false and condition2 is True
}
else
{
  // block of code to be executed if the condition1 is false and condition2 is False
}
```

```
using System;

namespace MyApplication
{
  class Program5
  {
    static void Main(string[] args)
    {
          double per;
Console.Write("Enter the Percentage : ");
                        per = Convert.ToDouble(Console.ReadLine());
      if (per >=75 10)
      {
        Console.WriteLine("First class with Distinction.");
       }
       else if (per >=60 && per<75)
       {
        Console.WriteLine("First Class.");
       }
      else if (per >=50 && per<60)
       {
        Console.WriteLine("Second Class.");
       }
      else if (per >=40 && per<50)
       {
        Console.WriteLine("Pass Class.");
       }
else {
        Console.WriteLine("Fail.");
       }
     }
   }
}
```

iv. **Switch-case statement:**select one of many code blocks to be executed.

Syntax:

```
switch(expression)
{
  case literal1:     // code block
                     break;
  case literal2:
               // code block
                     break;
  default:
               // code block
}
```

```
using System;

namespace MyApplication
{
class Program6
{

static void Main(string[] args)
{
int monthNumber;
Console.Write("Enter Month Number (1 - 12): ");
monthNumber = Convert.ToInt32(Console.ReadLine());
switch (monthNumber)
{
case 1:
Console.WriteLine("January");
break;
case 2:
Console.WriteLine("February");
break;
case 3:
Console.WriteLine("March");
break;
case 4:
Console.WriteLine("April");
break;
case 5:
Console.WriteLine("May");
break;
case 6:
Console.WriteLine("June");
break;
case 7:
Console.WriteLine("July");
break;
case 8:
Console.WriteLine("August");
break;
case 9:
Console.WriteLine("September");
break;
case 10:
Console.WriteLine("October");
break;
case 11:
Console.WriteLine("November");
break;
case 12:
Console.WriteLine("December");
break;
default:
Console.WriteLine("you did not enter correct value for month name");
break;
}

}
}
```

```
}
```

2. **Loop Control Structures :**
   Loops are the statements executed frequently until a specified condition becomes false.
   i. While loop: it is an entry point loop it executed as long as a specified condition is true.

   Syntax:
   ```
   while (condition) {
     // code block to be executed
   }
   ```
   Example:Find the Sum of First N Numbers in C#

```
using System;

namespace MyApplication
{
class Program7
    {
        static void Main(string[] args)
        {
            int num, sum=0;

            Console.Write("Enter a Number : ");
            num = Convert.ToInt32(Console.ReadLine());

            if(num <0)
            {
                Console.Write("Please Enter Positive Number");
            }
            else
            {
                while(num >0)
                {
                    sum =sum+num;
                    num = num - 1;
                }
            }
            Console.WriteLine("The sum is "+sum);

            Console.ReadKey();
        }
    }
}
```

ii. **do-while loop:** it is an exit control loop it executed at least once even though condition is false.

Syntax:

```
do
{
  // code block to be executed
} while (condition);
```

Example: Find the Sum of Digits of a given number in C#

```
using System;

namespace MyApplication
{
class Program8
   {
       static void Main(string[] args)
       {
          int number, sum=0,lastDigit;

          Console.Write("Enter a Number : ");
          number = Convert.ToInt32(Console.ReadLine());

              do
              {
                 lastDigit=number%10;
                  sum = sum+ lastDigit;
                  number =number/10;
              }while(number>0);

          Console.WriteLine("The sum is "+sum);

          Console.ReadKey();
       }
   }
}
```

iii. **for loop:** When we know the number of iterations in advance then for loop is used instead of while loop.

Syntax:

```
for (Initialization ; Condition ; Updation)
{
  // code block to be executed
}
```

Example: C# Program to Find the Factorial of a Number

```
using System;
```

```
namespace MyApplication
{
class Program9
    {
static void Main(string[] args)
        {
            int i, number, fact;
            Console.WriteLine("Enter the Number");
            number = int.Parse(Console.ReadLine());
            fact = number;
            for (i = 1; i <= number; i++)
            {
                fact = fact * i;
            }
            Console.WriteLine("\nFactorial of Given Number is: "+fact);
            Console.ReadLine();

      }
    }
}
```

3. **Jump Control Structures:** Transfer the control of flow unorderly.

i. **break statement:** it is use for abnormal termination of loop; it means break transfer the control out of loop.

```
using System;

namespace MyApplication
{
  class Program10
  {
    static void Main(string[] args)
    {
      for (int i = 0; i < 10; i++)
      {
        if (i == 4)
        {
          break;
        }
        Console.WriteLine(i);
      }
    }
  }
}
```

Output

```
0
1
2
3
```

ii. **continue statement:** it is use for breaks one iteration in the loop, if a specified condition occurs, and continues with the next iteration in the loop.

```
using System;

namespace MyApplication
{
  class Program11
  {
    static void Main(string[] args)
    {
      for (int i = 0; i < 10; i++)
      {
        if (i == 4)
        {
          continue;
        }
        Console.WriteLine(i);
      }
    }
  }
}
```

```
Output
0
1
2
3
5
6
7
8
9
```

Array
Array is use to storing the heterogeneous elements into single variable.
Array element is accessed using an index, it starts from 0 to size-1.

Syntax:
Datatype[] array_name=new Datatype[size];

Example:program to find min and max in an array.

```
using System;
```

```
namespace MyApplication
{
 class Program12
 {
       static void Main(string[] args)
       {
          int[] numbers = new int[10];
          Random rnd = new Random();
          int min, max;

          for (int i = 0; i < numbers.Length; i++)
          {
             numbers[i] = rnd.Next(1, 100);
             Console.WriteLine(numbers[i]);
          }
          min = numbers[0];
          max = numbers[0];
          for (int i = 1; i < numbers.Length; i++)
          {
             if (min > numbers[i])
                min = numbers[i];
             if (max < numbers[i])
                max = numbers[i];
          }
          Console.WriteLine("====================================");
          Console.WriteLine("The highest number in the array: " + max);
          Console.WriteLine("The lowest number in the array:" + min);
          Console.ReadKey();
       }
   }
}
```

Output

```
56
95
23
75
63
45
19
7
32
83
The highest number in the array:95
The lowest number in the array:7
```

Method: It is a block of statements used to perform a specific task and it may return value to the calling program.
Syntax to define method:

```
static return_type method_name([parameter_list])
{
        //method statements
}
```

Note: [] indicates it is optional.

Example: Check whether given number is even or odd using method

```
using System;

namespace MyApplication
{
class Program13
  {
      static bool IsEvenNumber(int num)
      {
         if (num % 2 == 0)
         {
            return true;
         }
         else
         {
            return false;
         }
      }
      static void Main(string[] args)
      {
         int n;
         Console.Write("Enter an integer : ");
         n = Int32.Parse(Console.ReadLine());

         if (IsEvenNumber(n))
         {
            Console.WriteLine("{0} is even", n);
         }
         else
         {
            Console.WriteLine("{0} is odd", n);
         }

         Console.ReadKey();
      }
  }
}
```

```
Output
Enter an integer: 37
37 is odd
```

String class:

In C#, string is keyword which is an alias for System.String class. That is why string and String are equivalent. We are free to use any naming convention.

string s1 = "india";//creating string using string keyword

String s2 = "Bharat";//creating string using String class

String methods

Method Name	**Description**
Clone()	It is used to return a reference to this instance of String.
Compare(String, String)	It is used to compares two specified String objects. It returns an integer that indicates their relative position in the sort order.
CompareOrdinal(String, String)	It is used to compare two specified String objects by evaluating the numeric values of the corresponding Char objects in each string..
CompareTo(String)	It is used to compare this instance with a specified String object. It indicates whether this instance precedes, follows, or appears in the same position in the sort order as the specified string.
Concat(String, String)	It is used to concatenate two specified instances of String.
Contains(String)	It is used to return a value indicating whether a specified substring occurs within this string.
Copy(String)	It is used to create a new instance of String with the same value as a specified String.
CopyTo(Int32, Char[], Int32, Int32)	It is used to copy a specified number of characters from a specified position in this instance to a specified position in an array of Unicode characters.
EndsWith(String)	It is used to check that the end of this string instance matches the specified string.
Equals(String, String)	It is used to determine that two specified String objects have the same value.
Format(String, Object)	It is used to replace one or more format items in a specified string with the string representation of a specified object.
GetEnumerator()	It is used to retrieve an object that can iterate through the individual characters in this string.
GetHashCode()	It returns the hash code for this string.
GetType()	It is used to get the Type of the current instance.
GetTypeCode()	It is used to return the TypeCode for class String.

IndexOf(String)	It is used to report the zero-based index of the first occurrence of the specified string in this instance.
Insert(Int32, String)	It is used to return a new string in which a specified string is inserted at a specified index position.
Intern(String)	It is used to retrieve the system's reference to the specified String.
IsInterned(String)	It is used to retrieve a reference to a specified String.
IsNormalized()	It is used to indicate that this string is in Unicode normalization form C.
IsNullOrEmpty(String)	It is used to indicate that the specified string is null or an Empty string.
IsNullOrWhiteSpace(String)	It is used to indicate whether a specified string is null, empty, or consists only of white-space characters.
Join(String, String[])	It is used to concatenate all the elements of a string array, using the specified separator between each element.
LastIndexOf(Char)	It is used to report the zero-based index position of the last occurrence of a specified character within String.
LastIndexOfAny(Char[])	It is used to report the zero-based index position of the last occurrence in this instance of one or more characters specified in a Unicode array.
Normalize()	It is used to return a new string whose textual value is the same as this string, but whose binary representation is in Unicode normalization form C.
PadLeft(Int32)	It is used to return a new string that right-aligns the characters in this instance by padding them with spaces on the left.
PadRight(Int32)	It is used to return a new string that left-aligns the characters in this string by padding them with spaces on the right.
Remove(Int32)	It is used to return a new string in which all the characters in the current instance, beginning at a specified position and continuing through the last position, have been deleted.
Replace(String, String)	It is used to return a new string in which all occurrences of a specified string in the current instance are replaced with another specified string.
Split(Char[])	It is used to split a string into substrings that are based on the characters in an array.
StartsWith(String)	It is used to check whether the beginning of this string instance matches the specified string.
Substring(Int32)	It is used to retrieve a substring from this instance. The substring starts at a specified character position and continues to the end of the string.
ToCharArray()	It is used to copy the characters in this instance to a Unicode character array.
ToLower()	It is used to convert String into lowercase.
ToLowerInvariant()	It is used to return convert String into lowercase using the casing rules of the invariant culture.

ToString()	It is used to return instance of String.
ToUpper()	It is used to convert String into uppercase.
Trim()	It is used to remove all leading and trailing white-space characters from the current String object.
TrimEnd(Char[])	It Is used to remove all trailing occurrences of a set of characters specified in an array from the current String object.
TrimStart(Char[])	It is used to remove all leading occurrences of a set of characters specified in an array from the current String object.

Example:

```
using System.Text;
namespace MyApplication
{
   class Program13
   {
      static void Main(string[] args)
      {
         string firstname;
         string lastname;

         firstname = "Steven Clark";
         lastname = "Clark";


  Console.WriteLine(firstname.Clone());
// Make String Clone
         Console.WriteLine(firstname.CompareTo(lastname));
//Compare two string value and returns 0 for true and1 for false

 Console.WriteLine(firstname.Contains("ven")); //Check whether specified value exists or not
in string

  Console.WriteLine(firstname.EndsWith("n")); //Check whether specified value is the last
character of string
         Console.WriteLine(firstname.Equals(lastname));
//Compare two string and returns true and false


  Console.WriteLine(firstname.GetHashCode());
//Returns HashCode of String

  Console.WriteLine(firstname.GetType());
//Returns type of string
```

```
  Console.WriteLine(firstname.GetTypeCode());
//Returns type of string

  Console.WriteLine(firstname.IndexOf("e")); //Returns the first index position of specified
value
the first index position of specified value

  Console.WriteLine(firstname.ToLower());
//Covert string into lower case

  Console.WriteLine(firstname.ToUpper());
//Convert string into Upper case

  Console.WriteLine(firstname.Insert(0, "Hello")); //Insert substring into string

  Console.WriteLine(firstname.IsNormalized());
//Check Whether string is in Unicode normalization
from C


   Console.WriteLine(firstname.LastIndexOf("e")); //Returns the last index position of
specified value

 Console.WriteLine(firstname.Length);
//Returns the Length of String

 Console.WriteLine(firstname.Remove(5));
//Deletes all the characters from begining to specified index.

 Console.WriteLine(firstname.Replace('e','i')); // Replace the character

  string[] split = firstname.Split(new char[] { 'e' }); //Split the string based on specified value


            Console.WriteLine(split[0]);
            Console.WriteLine(split[1]);
            Console.WriteLine(split[2]);

  Console.WriteLine(firstname.StartsWith("S"));
//Check wheter first character of string is same as specified value

  Console.WriteLine(firstname.Substring(2,5));
//Returns substring

  Console.WriteLine(firstname.ToCharArray());
//Converts an string into char array.
```

```
  Console.WriteLine(firstname.Trim());
//It removes starting and ending white spaces fromstring.
        }
    }
}
```

```
Output
Steven Clark
1
True
False
False
1470518261
System.String
String
2
steven clark
STEVEN CLARK
HelloSteven Clark
True
4
12
Steve
Stivin Clark
St
v
n Clark
True
even
Steven Clark
Steven Clark
```

Object Oriented Concepts

Object-oriented programming has several advantages over procedural programming:

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the C# code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time.

Object and Classes:

In C#, Object is a real world entity, for example, chair, car, pen, mobile, laptop etc.

```
Student s1 = new Student();//creating an object of Student
```

It is a template from which objects are created. It can have fields, methods, constructors etc.

Example:

```
using System;
   public class Student
    {
        public int id;
        public String name;
        public void insert(int i, String n)
        {
            id = i;
            name = n;
        }
        public void display()
        {
            Console.WriteLine(id + " " + name);
        }
   }
   class TestStudent{
       public static void Main(string[] args)
        {
            Student s1 = new Student();
            Student s2 = new Student();
            s1.insert(1001, "Abhijeet");
            s2.insert(1002, "Reshma");
            s1.display();
            s2.display();

        }
    }
```

Output

```
1001 Abhijeet
1002 Reshma
```

Access Modifiers / Specifiers

Access modifiers or specifiers are the keywords that are used to specify accessibility or scope of variables and functions in the C# application.

C# provides five types of access specifiers.

Access Specifier	**Description**
Public	It specifies that access is not restricted.
Protected	It specifies that access is limited to the containing class or in derived

	class.
Internal	It specifies that access is limited to the current assembly.
protected internal	It specifies that access is limited to the current assembly or types derived from the containing class.
Private	It specifies that access is limited to the containing type.

Constructors

In C#, constructor is a special method its name should be same as its class name.
which is invoked automatically at the time of object creation.
It is used to initialize the data members of new object generally.
It does not have any return type even though void.

There can be two types of constructors in C#.

1. **Default constructor:** constructor does not have any parameters or arguments.
2. **Parameterized constructor:** constructor have at least one parameter.

```
    using System;
      public class Employee
       {
          public int id;
          public String name;
          public float salary;
public Employee()
     {
    id = 101;
             name = “Rahul Bhosale”;
             salary = 750000f;


        Console.WriteLine("Default Constructor Invoked");
     }

          public Employee(int i, String n,float s)
          {
             id = i;
             name = n;
             salary = s;
    Console.WriteLine("Parameterised Constructor Invoked");
          }
          public void display()
          {
             Console.WriteLine(id + " " + name+" "+salary);
          }
        }
        class TestEmployee{
           public static void Main(string[] args)
            {
               Employee e1 = new Employee();
               Employee e2 = new Employee(102, "Mahesh Raut", 460000f);
               e1.display();
               e2.display();

            }
         }
```

```
Output
101Rahul Bhosale 750000
102Mahesh Raut 460000
```

Inheritance
In C#, inheritance is a process in which object of one class acquires the properties of object of another class.
Inheritance provides reusability of code and extends or modifies the attributes, behaviors which is defined in other class.
The class whose members are inherited is called base class or parent class or super class and the class which inherits the members of another class is called derived class or child class or sub class.
Types of inheritance:

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance

Example of Multilevel Inheritance:

```
using System;
  public class Animal
   {
      public void eat() { Console.WriteLine("Eating..."); }
  }
  public class Dog: Animal
  {
      public void bark() { Console.WriteLine("Barking..."); }
  }
  public class BabyDog : Dog
  {
      public void weep() { Console.WriteLine("Weeping..."); }
  }
  class TestInheritance{
      public static void Main(string[] args)
       {
            BabyDog d1 = new BabyDog();
            d1.eat();
            d1.bark();
            d1.weep();
        }
    }
```

```
Output
Eating...
Barking...
Weeping...
```

Abstract Classes and Methods

The abstract keyword is used for classes and methods:

Abstract class: It can have abstract and non-abstract methods. It cannot be instantiated. Its implementation must be provided by derived classes. Here, derived class is forced to provide the implementation of all the abstract methods.

Abstract method: A method which is declared abstract and has no body is called abstract method. It can be declared inside the abstract class only. Its implementation must be provided by derived classes.

Example:

```
using System;
public abstract class Shape
{
    public abstract void draw();
}
public class Rectangle : Shape
{
    public override void draw()
    {
        Console.WriteLine("drawing rectangle...");
    }
}
public class Circle : Shape
{
    public override void draw()
    {
        Console.WriteLine("drawing circle...");
    }
}
public class TestAbstract
{
    public static void Main()
    {
        Shape s;
        s = new Rectangle();
        s.draw();
        s = new Circle();
        s.draw();
    }
}
```

```
Output
drawing ractangle...
drawing circle..
```

Interface

It is used to achieve multiple inheritance which can't be achieved by class

An interface is a completely "abstract class", which can only contain abstract methods (with empty bodies)

Example:

```
using System;
public interface Drawable
{
    void draw();
}
public class Rectangle : Drawable
{
    public void draw()
    {
        Console.WriteLine("drawing rectangle...");
    }
}
public class Circle : Drawable
{
    public void draw()
    {
        Console.WriteLine("drawing circle...");
    }
}
public class TestInterface
{
    public static void Main()
    {
        Drawable d;
        d = new Rectangle();
        d.draw();
        d = new Circle();
        d.draw();
    }
}
```

```
Output
drawing ractangle...
drawing circle...
```

Method Overloading

Method having same name but different prototyping or signature called as method overloading.

The advantage of method overloading is that it increases the readability of the program because you don't need to use different names for same action.

Example

```
using System;
public class Calculate{
    public static int sum(int a,int b){
        return a + b;
    }
    public static int sum(int a, int b, int c)
    {
        return a + b + c;
    }
}
public class TestMemberOverloading
{
    public static void Main()
    {
        Console.WriteLine(Calculate.sum(10, 20));
        Console.WriteLine(Calculate.sum(10, 20, 30));
    }
}
```

```
Output
30
60
```

Method Overriding

If derived class defines same method as defined in its base class, it is known as method overriding.
It is used to achieve runtime polymorphism.

To perform method overriding, you need to use virtual keyword with base class method and override keyword with derived class method.
Example:

```
using System;
public class Animal{
   public virtual void eat(){
      Console.WriteLine("Eating...");
   }
}
public class Dog: Animal
{
   public override void eat()
   {
      Console.WriteLine("Eating bread...");
   }
}
public class TestOverriding
{
   public static void Main()
   {
      Dog d = new Dog();
      d.eat();
   }
}
```

```
Output
Eating bread...
```

Delegates

In C#, Delegate is a reference to the method. It works like function pointer in C and C++. But it is objected-oriented, secured and type-safe than function pointer.
For static method, delegate encapsulates method only. But for instance method, it encapsulates method and instance both.
The best use of delegate is to use as event.
Internally a delegate declaration defines a class which is the derived class of System.Delegate.

Example:

```
using System;
delegate int Calculator(int n);//declaring delegate
public class DelegateExample
{
   static int num = 100;
   public static int addition(int n)
   {
      num = num + n;
```

```
        return num ;
    }
    public static int multiplication(int n)
    {
        num = num * n;
        return num;
    }
    public static int getNumber()
    {
        return num;
    }
    public static void Main(string[] args)
    {
        Calculator c1 = new Calculator(addition);//instantiating delegate
        Calculator c2 = new Calculator(multiplication);
        c1(20);//calling method using delegate
        Console.WriteLine("After c1 delegate, Number is: " + getNumber());
        c2(3);
        Console.WriteLine("After c2 delegate, Number is: " + getNumber());

    }
}
```

Output

```
After c1 delegate, Number is: 120
After c2 delegate, Number is: 360
```

Assignment 4

Set A:

1. Program to display the addition, subtraction, multiplication and division of two number.
2. Program to display the first 10 natural numbers and their sum.
3. Accept three and find their maximum and minimum.
4. C# Program to Calculate the Power of a Number Without Using Math.Pow.
5. Define a Student class (roll number, name, and percentage). Define a default andparameterized constructor. Override the toString method. Keep a count objects created. Create objects using parameterized constructor and display the object count after each object is created. (Use static member and method).
6. Define a class Employee having members – id, name, department, salary. Define default and parameterized constructors. Create a subclass called "Manager" with member bonus. Define methods accept and display in both the classes. Create no. objects of the Manager class and display the details of the manager having the maximum total salary (salary + bonus)
7. Define an interface "IntOperations" with methods to check whether an integer ispositive, negative, even, odd, prime and operations like factorial and sum of digits. Define a class MyNumber having one private int data member. Write a default constructor toinitialize it to 0 and another constructor to initialize it to a value (Use this). Implement theabove interface. Create an object in main.

Set B:

1. Define an interface "StackOperations" which declares methods for a static stack. Define a class "MyStack" which contains an array and top as data members and implements the above interface. Initialize the stack using a constructor. Write a menu driven program to perform operations on a stack object.
2. Define a class CricketPlayer (name, no_of_innings, no_times_notout, total_runs,bat_avg). Create an array of n player objects. Calculate the batting average for each Player using a static method avg(). Define a static method "sortPlayer" which sorts the array on the basis of average. Display the player details in sorted order.

3. Create a class date with day, month and year as members. Write appropriate member functions. Create another class student, which has id, name, date of birth and marks of 3 subjects as members. Write appropriate constructor for the student which assigns values to the members. Accept the details from keyboard and create a student object using the arguments. Display the student details in a proper format.

Set C:

1. Define a class "Employee" which has members id, name, date of birth. Define another class "Manager" which has members department name and joining date and extends Employee. Create n objects of the manager class.

2. Define an abstract class "Staff" with members name and address. Define two subclasses of this class – "FullTimeStaff" (department, salary) and "PartTimeStaff" (numberof_hours, rate_per_hour). Define appropriate constructors. Create n objects which could be of either FullTimeStaff or PartTimeStaff class by asking the user's choice. Display details of all "FullTimeStaff" objects and all "PartTimeStaff" objects.

3. Create an interface "CreditCardInterface" with methods to viewCreditAmount,viewPin, changePin, useCard and payBalance. Create a class Customer (name, cardnumber, pin, creditAmount – initialized to 0). Implement methods viewCreditAmount,viewPin, changePin and payBalance of the interface. From Customer,create classesRegularCardHolder (maxCreditLimit) and GoldCardHolder (String specialPrivileges)and define the remaining methods of the interface. Create n objects of the RegularCardHolder and GoldCardHolder classes and writea menu driven program to perform the following actions
   1. Use Card
   2. Pay Balance
   3. Change Pin

Assignment Evaluation

0: Not Done [ ] **1: Incomplete [ ]** **2: Late Complete [ ]**

3: Need Improvement [ ] **4: Complete [ ]** **5: Well Done [ ]**

Signature of Instructor

ASSIGNMENT -D: ASP.NET

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC as well as mobile devices. ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation. ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

The architecture of the.Net framework is based on the following key components

1. **Language –** A variety of languages exists for .net framework. They are VB.net and C#. These can be used to develop web applications.
2. **Library –** The .NET Framework includes a set of standard class libraries. The most common library used for web applications in .net is the Web library. The web library has all the necessary components used to develop.Net web-based applications.
3. **Common Language Runtime –** The Common Language Infrastructure or CLI is a platform. .Net programs are executed on this platform. The CLR is used for performing key activities. Activities include Exception handling and Garbage collection.

ASP.Net Page Lifecycle

When an ASP.Net page is called, it goes through a particular lifecycle. This is done before the response is sent to the user. There are series of steps which are followed for the processing of an ASP.Net page.

The various stages of the lifecycle of an ASP.Net web page are as:

1. **Page Request–** This is when the page is first requested from the server. When the page is requested, the server checks if it is requested for the first time. If so, then it needs to compile the page, parse the response and send it across to the user. If it is not the first time the page is requested, the cache is checked to see if the page output exists. If so, that response is sent to the user.

2. **Page Start –** During this time, 2 objects, known as the Request and Response object are created. The Request object is used to hold all the information which was sent when the page was requested. The Response object is used to hold the information which is sent back to the user.
3. **Page Initialization –** During this time, all the controls on a web page is initialized. So if you have any label, textbox or any other controls on the web form, they are all initialized.
4. **Page Load –** This is when the page is actually loaded with all the default values. So if a textbox is supposed to have a default value, that value is loaded during the page load time.
5. **Validation –** Sometimes there can be some validation set on the form. For example, there can be a validation which says that a list box should have a certain set of values. If the condition is false, then there should be an error in loading the page.
6. **Postback event handling –** This event is triggered if the same page is being loaded again. This happens in response to an earlier event. Sometimes there can be a situation that a user clicks on a submit button on the page. In this case, the same page is displayed again. In such a case, the Postback event handler is called.
7. **Page Rendering –** This happens just before all the response information is sent to the user. All the information on the form is saved, and the result is sent to the user as a complete web page.
8. **Unload –** Once the page output is sent to the user, there is no need to keep the ASP.net web form objects in memory. So the unloading process involves removing all unwanted objects from memory.

Working with Views and Windows

You can work with windows in the following ways:

- To change the Web Forms Designer from one view to another, click on the Design or source button.
- To close a window, click on the close button on the upper right corner and to redisplay, select it from the View menu.
- To hide a window, click on its Auto Hide button. The window then changes into a tab.
- To display again, click the Auto Hide button again.
- To change the size of a window, just drag it.

Adding Folders and Files to your Website

When a new web form is created, Visual Studio automatically generates the starting HTML for the form and displays it in Source view of the web forms designer. The Solution Explorer is used to add any other files, folders or any existing item on the web site.

- To add a standard folder, right-click on the project or folder under which you are going to add the folder in the Solution Explorer and choose New Folder.
- To add an ASP.NET folder, right-click on the project in the Solution Explorer and select the folder from the list.
- To add an existing item to the site, right-click on the project or folder under which you are going to add the item in the Solution Explorer and select from the dialog box.

Projects and Solutions

A typical ASP.NET application consists of many items: the web content files (.aspx), source files (.cs files), assemblies (.dll and .exe files), data source files (.mdb files), references, icons, user controls and miscellaneous other files and folders. All these files that make up the website are contained in a Solution.

When a new website is created, dotnet environment automatically creates the solution and displays it in the solution explorer.

Solutions may contain one or more projects. A project contains content files, source files, and other files like data sources and image files. Generally, the contents of a project are compiled into an assembly as an executable file (.exe) or a dynamic link library (.dll) file.

Typically a project contains the following content files:

- Page file (.aspx)
- User control (.ascx)
- Web service (.asmx)
- Master page (.master)
- Site map (.sitemap)
- Website configuration file (.config)

Building and Running a Project:

You can execute an application by:

- Selecting Start
- Selecting Start Without Debugging from the Debug menu
- pressing F5
- Ctrl-F5

The program is built meaning, the .exe or the .dll files are generated by selecting a command from the Build menu.

ASP.NET Application Life Cycle

The application life cycle has the following stages:

1. User makes a request for accessing application resource, a page. Browser sends this request to the web server.

2. A unified pipeline receives the first request and the following events take place:

i. An object of the class ApplicationManager is created.

ii. An object of the class HostingEnvironment is created to provide information regarding the resources. iii. Top level items in the application are compiled.

3. Response objects are created. The application objects such as HttpContext, HttpRequest and HttpResponse are created and initialized.

4. An instance of the HttpApplication object is created and assigned to the request.

5. The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.

ASP.NET Page Life Cycle

The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

EVENT HANDLING

An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification. A process communicates through events. For example, interrupts are system-generated events. When events occur, the application should be able to respond to it and manage it.

Events in ASP.NET are raised at the client machine and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.

Event Arguments

ASP.NET event handlers generally take two parameters and return void. The first parameter represents the object raising the event and the second parameter is event argument. The general syntax of an event is:

private void EventName (object sender, EventArgs e);

Application and Session Events

The most important application events are:

- Application_Start - It is raised when the application/website is started
- Application_End - It is raised when the application/website is stopped. Similarly, the most used Session events are
  - Session_Start - It is raised when a user first requests a page from the application.
  - Session_End - It is raised when the session ends.

Page and Control Events

Common page and control events are:

- DataBinding – It is raised when a control binds to a data source.
- Disposed – It is raised when the page or the control is released.
- Error - It is a page event, occurs when an unhandled exception is thrown.
- Init – It is raised when the page or the control is initialized.
- Load – It is raised when the page or a control is loaded.
- PreRender – It is raised when the page or the control is to be rendered.
- Unload – It is raised when the page or control is unloaded from memory.

Event Handling Using Controls:

All ASP.NET controls are implemented as classes, and they have events which are fired when a user performs a certain action on them. For example, when a user clicks a button the 'Click' event is generated. For handling events, there are in-built attributes and event handlers. Event handler is coded to respond to an event and take appropriate action on it.

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles btnCancel.Click

End Sub
```

Example: Write a Program to calculate factorial of number

```
Partial Class _Default
Inherits System.Web.UI.Page
Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
Handles Button1.Click
Dim i As Integer
Dim a As Double
Dim f As Double
f = 1
i = 1
a = TextBox1.Text
While i <= a
f = f * i
i = i + 1
End While
Label3.Text = f & "unit"
End Sub
End Class
```

Write a Program to bind data using template in data list

```
Imports System.Data.SqlClient
Partial Class _Default
Inherits System.Web.UI.Page
Dim constr As String =
ConfigurationManager.ConnectionStrings("DatabaseConnectionString1").Connection String
Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Me.Load
Dim conpubs As SqlConnection Dim cmdselect As SqlCommand Dim dtrAuthor As
SqlDataReader
conpubs = New SqlConnection(constr)
cmdselect = New SqlCommand("Select * from Author", conpubs) conpubs.Open()
dtrAuthor = cmdselect.ExecuteReader() dtrlstAuthor.DataSource = dtrAuthor
dtrlstAuthor.DataBind() dtrAuthor.Close()
conpubs.Close()
End Sub
End Class
```

Assignment 5

1. Write a Program to convert temperature from Fahrenheit to Celsius or vice versa.
2. Write a Program to display the selected date in the calendar
3. Write a Program to display the Difference between the two dates in the calendar.

Assignment Evaluation

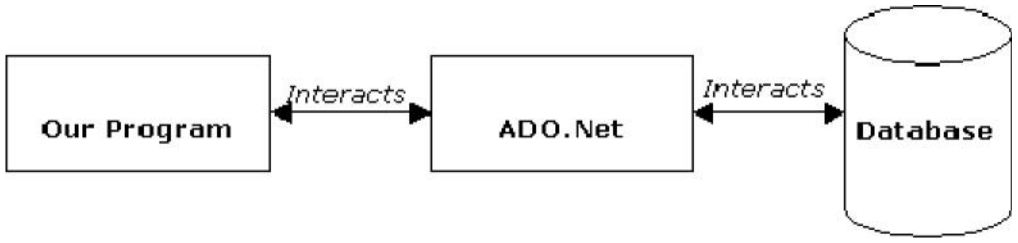
0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

ASSIGNMENT -E: ADO .NET

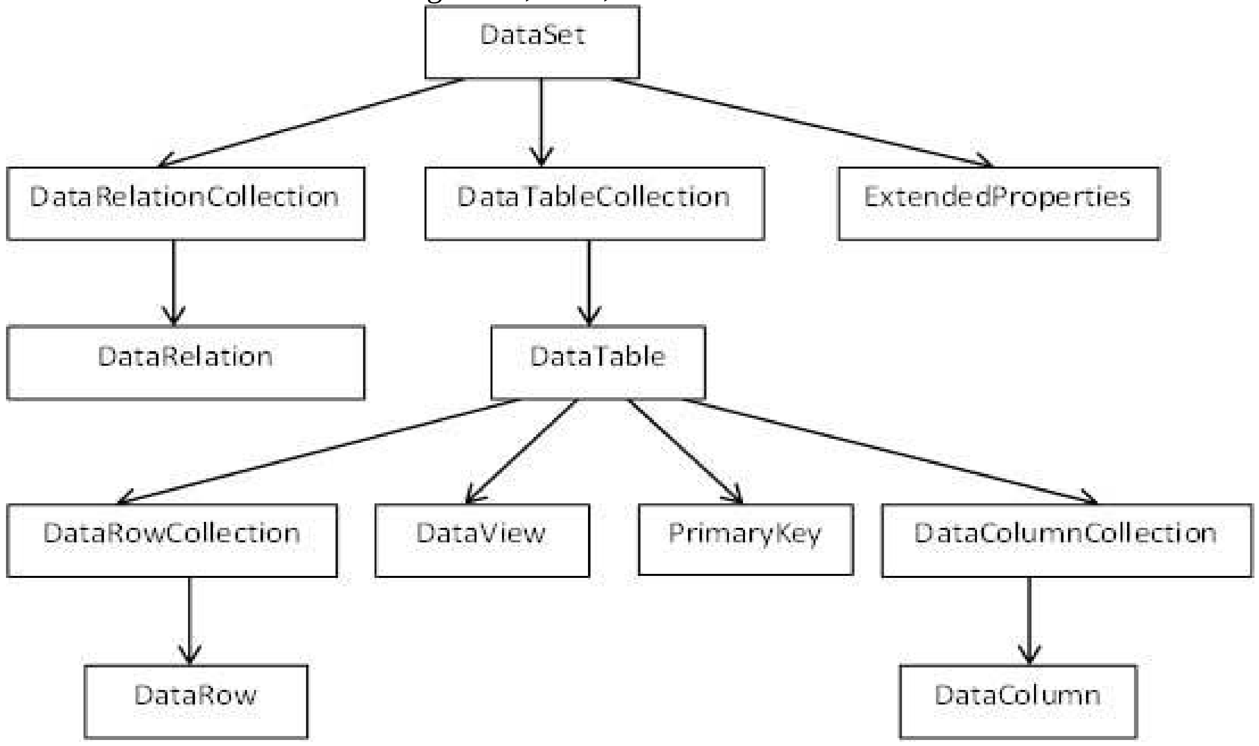
A data-access technology that enables applications to connect to data stores and manipulate data contained in them in various ways.

An object oriented framework that allows you to interact with database systems



ADO.NET components

- 1) Dataset
- 2) Data Provider
- **DataSet** is an in-memory representation of data. It is a disconnected, cached set of records that are retrieved from a database.
- When a connection is established with the database, the data adapter creates a dataset and stores data in it. After the data is retrieved and stored in a dataset, the connection with the database is closed. This is called the 'disconnected architecture'. The dataset works as a virtual database containing tables, rows, and columns.



Connecting to a Database

- The .Net Framework provides two types of Connection classes −
- **SqlConnection** − designed for connecting to Microsoft SQL Server.
- **OleDbConnection** − designed for connecting to a wide range of databases, like Microsoft Access and Oracle.

Steps :

- We have a table stored in Microsoft SQL Server, named Customers, in a database named testDB.
- Select TOOLS → Connect to Database
- Select a server name and the database name in the Add Connection dialog box.
- Click on the Test Connection button to check if the connection succeeded.
- Add a DataGridView on the form.
- Click on the Choose Data Source combo box. Click on the Add Project Data Source link.
- This opens the Data Source Configuration Wizard.
  Select Database as the data source type
- Choose DataSet as the database model.
- Choose the connection already set up.
- Save the connection string.
- Choose the database object, Customers table in our example, and click the Finish button.
- When the application is run using Start button available at the Microsoft Visual Studio tool bar, it will show the following window

2) Data Provider:

A data provider is used for connecting to a database, executing commands and retrieving data, storing it in a dataset, reading the retrieved data and updating the database.

The data provider in ADO.Net consists of the following four objects –

1) Connection

This component is used to set up a connection with a data source.

2) Command

A command is a SQL statement or a stored procedure used to retrieve, insert, delete or modify data in a data source.

3)DataReader

Data reader is used to retrieve data from a data source in a read-only and forward-only mode.

4) DataAdapter

This is integral to the working of ADO.Net since data is transferred to and from a database through a data adapter. It retrieves data from a database into a dataset and updates the database. When changes are made to the dataset, the changes in the database are actually done by the data adapter.

There are following different types of data providers included in ADO.Net

1) The .Net Framework data provider for SQL Server - provides access to Microsoft SQL Server.

2) The .Net Framework data provider for OLE DB - provides access to data sources exposed by using OLE DB.

3) The .Net Framework data provider for ODBC - provides access to data sources exposed by ODBC.

4) The .Net Framework data provider for Oracle - provides access to Oracle data source.

5) The Entity Client provider - enables accessing data through Entity Data Model (EDM) applications.

CONNECTIONS:

ADO.NET provids both Sqlconnection and OLEDB connection classes for use with datasource.

1) Sql Connection :-

To declare and instantiate Sql connection,u can use foll.stmt

```
Dim con AS Sqlconnection

con =  New     SqlConnection("server=YASH;database=test;integrated security=true")
```

2) OleDbConnection

```
Dim con As OleDbConnection

Dim con As OleDbConnection
```

con = New OleDbconnection ("Provider=Microsoft.Jet.OLEDB.4.0;Data Source=yourdatabasename.mdb”)

Command

We use command object to tell database about operation we need to perform.

The typical command on database will be :-

1) Select

2) Insert

3) Delete

4) Update

EXAMPLE:creating OLEDB Data connection

```
   imports System.Data.OleDb

   Public Class Form1

    Dim cn As New OleDbConnection

    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load


        cn = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\user\Documents\Database5.accdb")

        cn.Open()

        MsgBox("oledb connection is successfully  established")

 End Sub

End Class
```

Data Adapter

- DataAdapter is a part of the ADO.NET Data Provider. DataAdapter provides the communication between the Dataset and the Datasource. We can use the DataAdapter in combination with the DataSet Object. That is these two objects combine to enable both data access and data manipulation capabilities.

- The DataAdapter can perform Select , Insert , Update and Delete SQL operations in the Data Source. The Insert , Update and Delete SQL operations , we are using the continuation of the Select command perform by the DataAdapter. That is the DataAdapter uses the Select statements to fill a DataSet and use the other three SQL commands (Insert, Update, delete) to transmit changes back to the Database
- **Types of DataAdapter**

1) SqlDataAdapter

2) OleDbDataAdapter

Types of DataAdapter

1) SqlDataAdapter :-

```
  Dim ad as   SqlDataAdapter
    ad=new  SqlDataAdapter (cmd)
```

2) OleDbDataAdapter :-

```
    Dim ad as OleDbDataAdapter
    ad=new OleDbDataAdapter (cmd)
```

Example

```
Imports System.Data.OleDb
    Public Class Form1
        Dim cn As New OleDbConnection
        Dim cmd As New OleDbCommand
        Dim ad As New OleDbDataAdapter
        Dim dt As New DataTable
          Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
          System.EventArgs) Handles MyBase.Load
            'TODO: This line of code loads data into the 'Database5DataSet.doctor' table. You can
    move, or remove it, as needed.
            Me.DoctorTableAdapter.Fill(Me.Database5DataSet.doctor)
            cn = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
    Source=C:\Users\user\Documents\Database5.accdb")

            cn.Open()

            MsgBox("oledb connection is successfully  established")

            cmd = New OleDbCommand("select  * from doctor", cn)

            ad = New OleDbDataAdapter(cmd)

            ad.Fill(dt)

            DataGridView1.DataSource = dt

         End Sub

    End Class
```

Dataset

- The DataSet contains the copy of the data we requested through the SQL statement. We can use Dataset in combination with SqlDataAdapter class . The SqlDataAdapter object allows us to populate Data Tables in a DataSet.

Example

```
    Imports System.Data.OleDb

    Public Class Form1

   Dim cn As New OleDbConnection

   Dim cmd As New OleDbCommand

   Dim ad As New OleDbDataAdapter

   Dim ds As New DataSet()

      Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
    System.EventArgs) Handles MyBase.Load


      Me.DoctorTableAdapter.Fill(Me.Database5DataSet.doctor)

      cn = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\user\Documents\Database5.accdb")
```

```
        cn.Open()

        MsgBox("oledb connection is successfully  established")

adp = New OleDbDataAdapter(cmd)

        adp = New OleDbDataAdapter("select * from doctor", cn)

       ad.Fill(ds, "doctor")

         DataGridView1.DataSource = ds

        DataGridView1.DataMember = "doctor"

       End Sub

     End Class
```

Data Reader

- To retrieve data using a DataReader, create an instance of the Command object, and then create a DataReader bycalling Command.ExecuteReader to retrieve rows from a data source.
- DataReader Object in ADO.NET is a stream-based , forward-only, read-only retrieval of query results from the Data Source, which do not update the data. The DataReader cannot be created directly from code, they created only by calling the Execute Reader method of a Command Object.

```
DataReader = Command.ExecuteReader()
```

DataTable

- DataTable represents relational data into tabular form. ADO.NET provides a DataTable class to create and use data table independently. It can also be used with DataSet also. Initially, when we create DataTable, it does not have table schema. We can create table schema by adding columns and constraints to the table. After defining table schema, we can add rows to the table.
- We must include System.Data namespace before creating DataTable.

Example

```
      Imports System.Data. DataTable

      Public Class Form1

    Dim table As New DataTable("table")
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load

  table.Columns.Add("ID", Type.GetType("System.Int32"))

  table.Columns.Add("FIRST NAME", Type.GetType("System.Int32"))

  table.Columns.Add("LAST NAME", Type.GetType("System.Int32"))

  table.Columns.Add("SUM", Type.GetType("System.Int32"))

  DataGridView1.DataSource = table

End Sub

    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click

  DataGridView1.DataSource = table

    End Sub
```

Data Validation

- *Data validation* is the process of ensuring, at least as far as is possible, that the data given to a program by a user or from a file is of the correct type, and in the correct format.
- Although the programmer will obviously take every precaution to ensure the correct operation of the program, and will attempt to eliminate bugs that could cause a problem through a rigorous process of testing, they have no real control over mistakes made by the user during data entry. Nor can they guarantee that any data files used by the program will be free of errors and in the correct format.

- Validation is a form of self protection. It is the checking to verify that appropriate values have been entered for textbox or any control .

Validation may include :-

1) Making sure data is numeric or text depending on application.

2) Checking for specific values.

3) Checking for range of values

4) Making sure required items are entered.

Write a program to access data source through ADO.NET.

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;

public partial class marksheet : System.Web.UI.Page
{
        OleDbConnection con = new OleDbConnection("Provider=MSDAORA; User Id=result;
Password=college");
        OleDbCommand cmd = new OleDbCommand();
        OleDbCommand cmd1 = new OleDbCommand();
        OleDbDataReader dr,dr1;
        protected void Page_Load(object sender, EventArgs e)
        {
      cmd.Connection = con;
      cmd1.Connection = con;
        con.Open();

        string s1 = Session["rollno"].ToString();
        string s2 = Session["sem"].ToString();
        string s3 = Session["branch"].ToString();

      cmd.CommandText = "select * from DATABASE where ROLLNO='"+s1+"' and
SEM='"+s2+"' and BRANCH='"+s3+"' ";
        dr = cmd.ExecuteReader();
if (dr.Read())
        {


         string d1 = dr.GetValue(0).ToString();
        string d2 = dr.GetValue(1).ToString();
        string d3 = dr.GetValue(2).ToString();
        string d4 = dr.GetValue(3).ToString();
        string d5 = dr.GetValue(4).ToString();
        string d6 = dr.GetValue(5).ToString();
        string d7 = dr.GetValue(6).ToString();
        string d8 = dr.GetValue(7).ToString();
```

```
            TextBox1.Text = d1;
            TextBox2.Text = d2;
            TextBox3.Text = d3;
            Label16.Text = d4;
            Label17.Text = d5;
            Label18.Text = d6;
            Label19.Text = d7;
            Label20.Text = d8;
            dr.Dispose();
}
cmd1.CommandText = "select * from SEM_BRANCH_SUB where sem='"+s2+"' and
branch='"+s3+"' ";
        OleDbDataReader dr1 = cmd1.ExecuteReader();
           if (dr1.Read())
           {
           string d9 = dr1.GetValue(2).ToString();
           string d10 = dr1.GetValue(3).ToString();

           string d11 = dr1.GetValue(4).ToString();
           string d12 = dr1.GetValue(5).ToString();
           string d13 = dr1.GetValue(6).ToString();

            Label10.Text = d9;
  Label11.Text = d10;
            Label12.Text = d11;
            Label13.Text = d12;
            Label14.Text = d13;

           }

           }
 }
```

Database programs with ASP.NET and ADO.NET Create a Login Module which adds Username and Password in the database. Username in the database should be a primary key.

CODE:

LoginModule.aspx

```
using System;

using System.Collections.Generic;

using System.Linq;
using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

public partial class LoginModule : System.Web.UI.Page

{

protected void Page_Load(object sender, EventArgs e)

{

}

protected void btnSignUp_Click(object sender, EventArgs e)

{

SqlDataSource1.InsertParameters["Username"].DefaultValue = txtUserName.Text;

SqlDataSource1.InsertParameters["Password"].DefaultValue = txtPassword.Text;

SqlDataSource1.Insert();

lblResult.Text = "User Added";

}

}
```

Assignment: 6

1. Create a web application to insert 3 records inside the SQL database table having following fields(DeptId, DeptName, EmpName, Salary). Update the salary for any one employee and increment it to 15% of the present salary. Perform delete operation on 1 row of the database table.
2. Create the table with the given fields. FIELD NAME DATA TYPE (EmpNo number EmpName varchar EmpSal number EmpJob varchar EmpDeptNo number)For the given table design a web page to display the employee information from table to grid control.
3. Create the table with the given fields. FIELD NAME DATA TYPE SRollno int SName string SAddress string SFees int For the given table design a web page to display the employee information from table to grid control.

4. Write a program to connect to the master database in SQL Server, in the Page_Load event. When the connection is established, the message “Connection has been established” should be displayed in a label in the form.

Assignment Evaluation

0: Not Done [ ]	**1: Incomplete [ ]**	**2: Late Complete [ ]**
3: Need Improvement [ ]	**4: Complete [ ]**	**5: Well Done [ ]**

Signature of Instructor

Section III: RIT

T.Y.B.B.A.(C.A.) Sem-VI (CBCS 2019 Pattern)

Subject Code: CA-601

Subject: Recent Trends in IT Practical Assignments

Artificial Intelligence Practical Assignments: (Python / R programming)

1. Write a program to implement Breadth First Search Algorithm.
2. Write a program to implement Depth First Search Algorithm.
3. Write a program to implement water jug problem.
4. Write a program to implement Tower of Hanoi problem.

Data Mining Practical Assignments:

1. Build a classification model and usage of weka to use a Decision Tree algorithm to classify data from the “weather.arff” file. Perform initial preprocessing and create a version of the initial dataset in which all numeric attributes should be converted to categorical data.
2. Use database “labor.arff” Apply Linear Regression and find out total number of instances. (using weka / R).
3. Use Naïve Bayes algorithm to diabetes data from the “diabetes.arff” file. Perform initial preprocessing and create a version of the initial data set in which the ID field should be removed and the "Type" attribute should be converted to categorical data.
   (using weka /R)
4. Build a classification model and usage of WEKA to implement Naïve Bayes algorithm to classify whether data from the “Iris.arff” file on attribute plat growth and leaves. Perform initial pre-processing and create a version of the initial dataset in which all numeric attributes should be converted to categorical data.
5. Use Zero R classification algorithm to supermarket data from the “supermarket.arff” file. Perform initial preprocessing and analyze confusion matrix. (using weka /R)
6. Use the simple K-means algorithm for database “bank.arff” with the default settings and find out final cluster centroids.
7. Use Apriori algorithm to vote data of a transaction and identify all frequent k-item set and min support is 40% from the “vote.arff” file. Perform initial preprocessing and find all the frequent item sets. (using weka /R).
8. Use Hierarchical Clustering classificational algorithm to tic-tac-toe data from the “tic-tac-toe.arff” file. Perform initial pre-processing and create a version of the initial data set in which the ID field should be removed and the "class" attribute should be converted to categorical data. (Using weka /R).

9. Build a classification model and usage of weka to use a Decision Tree algorithm to classify whether data from the “labour.arff” file. Perform initial preprocessing and create a version of the initial dataset in which all numeric attributes should be converted to categorical data.
10. Use the FP Growth algorithm and Apriori algorithm for database “supermarket.arff”, with the default settings and find out which algorithm generates maximum rules.

Spark Practical Assignments:

1. Write a spark program to apply the map function and pass the expression required to perform.
2. Write a spark program to apply filter function and pass the expression required to perform.
3. Write a spark program to apply intersection() function to return the intersection of the elements.
4. Write a spark program to apply reduceByKey() function to aggregate the values.
5. Write a spark program to demonstrate the different ways to create a DataFrame.
6. Write a spark program to add or update column on DataFrame.
7. Write a spark program to remove distinct on multiple selected columns.

Section IV: Soft Skill

T.Y.B.B.A.(C.A.) Semester-VI

Subject: (CA – 607) Soft Skill Practical Assignments

1 Practice on Oral and spoken communication skill & testing – voice & accent, voice clarity, voice modulation & word stress etc.

2 Study of different pictorial expression of non-verbal communication and its analysis

3 Development etiquettes and manners

4 Conduct demo session on interview.

5 Prepare a presentation and Deliver it on specific topic.

6 Prepare your own resume.

7 Take a Group Discussion on some topic like Education, Social media , Online System etc.

8 Team Building Practices through group exercises, team task / role play.