

# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name: Shiva Shree

PRN:72018544M

Date:-

Roll No:-TEB237

---

### Experiment No. 1

#### Problem Statement: -

Write a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.

#### Objectives: -

The objective of this assignment is to learn the concepts of key board event handling.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans

#### Theory :-

#### Java Event Handling

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.

| Event Classes   | Listener Interfaces                   |
|-----------------|---------------------------------------|
| ActionEvent     | ActionListener                        |
| MouseEvent      | MouseListener and MouseMotionListener |
| MouseWheelEvent | MouseWheelListener                    |
| KeyEvent        | KeyListener                           |

|                 |                    |
|-----------------|--------------------|
| ItemEvent       | ItemListener       |
| TextEvent       | TextListener       |
| AdjustmentEvent | AdjustmentListener |
| WindowEvent     | WindowListener     |
| ComponentEvent  | ComponentListener  |
| ContainerEvent  | ContainerListener  |
| FocusEvent      | FocusListener      |

### Java KeyListener Interface

The **Java KeyListener** is notified whenever you change the state of key. It is notified against KeyEvent. The KeyListener interface is found in java.awt.event package, and it has three methods.

#### Interface declaration

Following is the declaration for **java.awt.event.KeyListener** interface:

1. **public interface** KeyListener **extends** EventListener

#### Methods of KeyListener interface

The signature of 3 methods found in KeyListener interface are given below:

| Sr. no. | Method name                                    | Description                                 |
|---------|--|---|
| 1.      | public abstract void keyPressed (KeyEvent e);  | It is invoked when a key has been pressed.  |
| 2.      | public abstract void keyReleased (KeyEvent e); | It is invoked when a key has been released. |

|    |   |  |
|----|---|--|
| 3. | public abstract void keyTyped (KeyEvent e); | It is invoked when a key has been typed. |
|----|---|--|

Methods inherited

This interface inherits methods from the following interface:

- java.awt.EventListener

### KeyListenerExample.java

```
// importing awt libraries
import java.awt.*;
import java.awt.event.*;
// class which inherits Frame class and implements KeyListener interface
public class KeyListenerExample extends Frame implements KeyListener {
// creating object of Label class and TextArea class
Label l;
    TextArea area;
// class constructor
    KeyListenerExample() {
        // creating the label
        l = new Label();
// setting the location of the label in frame
        l.setBounds (20, 50, 100, 20);
// creating the text area
        area = new TextArea();
// setting the location of text area
        area.setBounds (20, 80, 300, 300);
// adding the KeyListener to the text area
        area.addKeyListener(this);
// adding the label and text area to the frame
        add(l);
        add(area);
// setting the size, layout and visibility of frame
        setSize (400, 400);
        setLayout (null);
        setVisible (true);
```

```
}
```

// overriding the keyPressed() method of KeyListener interface where we set the text of the label when key is pressed

```
public void keyPressed (KeyEvent e) {  
    l.setText ("Key Pressed");  
}
```

// overriding the keyReleased() method of KeyListener interface where we set the text of the label when key is released

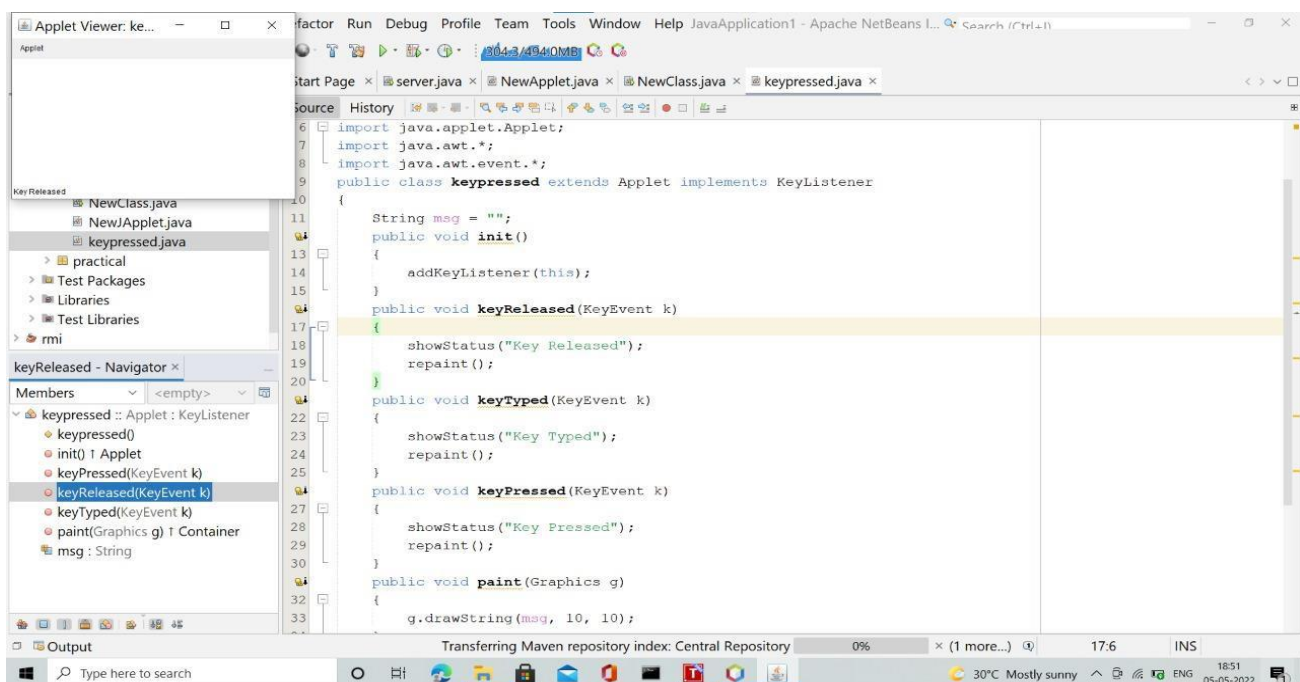
```
public void keyReleased (KeyEvent e) {  
    l.setText ("Key Released");  
}
```

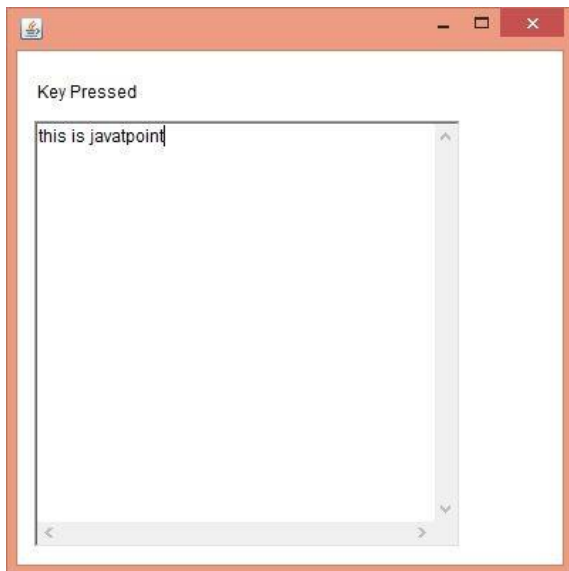
// overriding the keyTyped() method of KeyListener interface where we set the text of the label when a key is typed

```
public void keyTyped (KeyEvent e) {  
    l.setText ("Key Typed");  
}
```

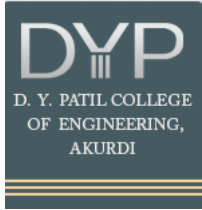
// main method

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





**Conclusion :-**



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name : Shiva Shree

PRN: 72018544M

Date:-

Roll No:-TEB237

### Experiment No. 2

#### Problem Statement: -

Write a program to create a frame using AWT. Implement mouseClicked, mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it.

#### Objectives: -

The objective of this assignment is to learn awt and swing package in Java and mouse event handling.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans

#### Theory :-

**Java AWT** (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Java AWT components are platform-dependent i.e. components are displayed according to the view of operating system. AWT is heavy weight i.e. its components are using the resources of underlying operating system (OS).

The java.awt package provides classes for AWT API such as TextField, Label, TextArea, RadioButton, CheckBox, Choice, List etc.

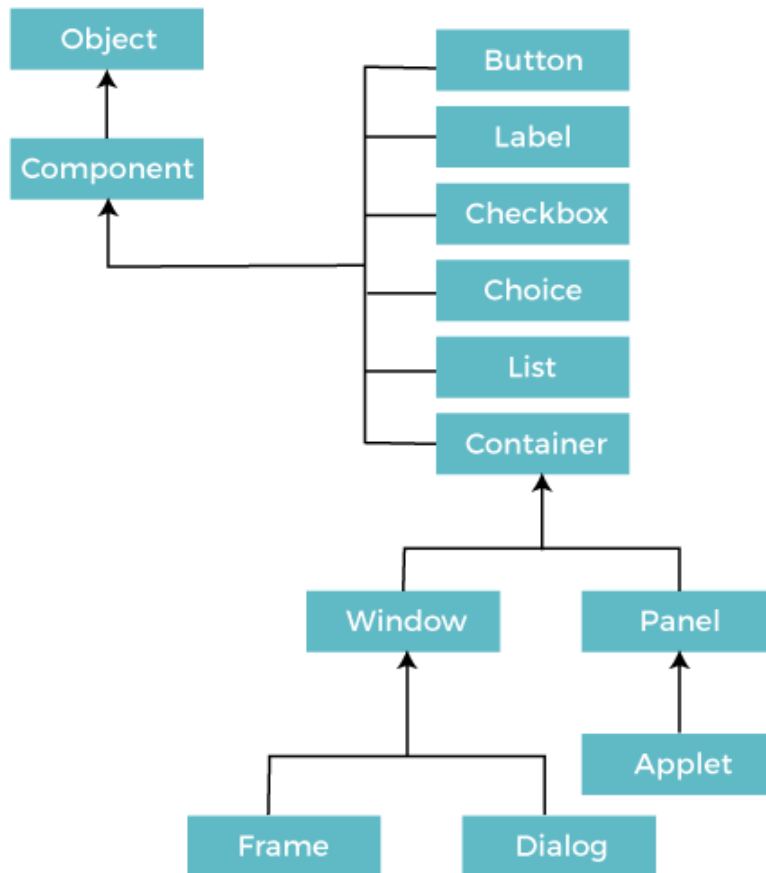
Java AWT calls the native platform calls the native platform (operating systems) subroutine for creating API components like TextField, ChechBox, button, etc.

For example, an AWT GUI with components like TextField, label and button will have different look and feel for the different platforms like Windows, MAC OS, and Unix. The reason for this is the platforms have different view for their native components and AWT directly calls the native subroutine that creates those components.

In simple words, an AWT application will look like a windows application in Windows OS whereas it will look like a Mac application in the MAC OS.

### Java AWT Hierarchy

The hierarchy of Java AWT classes are given below.



### Components

All the elements like the button, text fields, scroll bars, etc. are called components. In Java AWT, there are classes for each component as shown in above diagram. In order to place every component in a particular position on a screen, we need to add them to a container.

### Container

The Container is a component in AWT that can contain another components like [buttons](#), textfields, labels etc. The classes that extends Container class are known as container such as **Frame**, **Dialog** and **Panel**.

It is basically a screen where the where the components are placed at their specific locations. Thus it contains and controls the layout of components.

## Types of containers:

There are four types of containers in Java AWT:

1. Window
2. Panel
3. Frame
4. Dialog

### Window

The window is the container that have no borders and menu bars. You must use frame, dialog or another window for creating a window. We need to create an instance of Window class to create this container.

### Panel

The Panel is the container that doesn't contain title bar, border or menu bar. It is generic container for holding the components. It can have other components like button, text field etc. An instance of Panel class creates a container, in which we can add components.

### Frame

The Frame is the container that contain title bar and border and can have menu bars. It can have other components like button, text field, scrollbar etc. Frame is most widely used container while developing an AWT application.

### Java MouseListener Interface

The Java MouseListener is notified whenever you change the state of mouse. It is notified against MouseEvent. The MouseListener interface is found in java.awt.event package. It has five methods.

### Methods of MouseListener interface

The signature of 5 methods found in MouseListener interface are given below:

```
public abstract void mouseClicked(MouseEvent e);  
public abstract void mouseEntered(MouseEvent e);  
public abstract void mouseExited(MouseEvent e);  
public abstract void mousePressed(MouseEvent e);  
public abstract void mouseReleased(MouseEvent e);
```



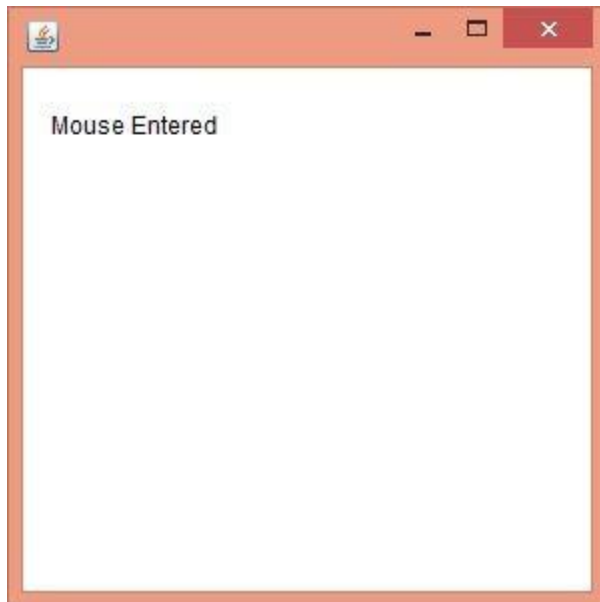
## Java MouseListener Example

```
import java.awt.*;
import java.awt.event.*;
public class MouseListenerExample extends Frame implements MouseListener{
    Label l;
    MouseListenerExample(){
        addMouseListener(this);

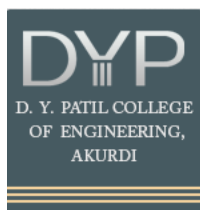
        l=new Label();
        l.setBounds(20,50,100,20);
        add(l);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public void mouseClicked(MouseEvent e) {
        l.setText("Mouse Clicked");
    }
    public void mouseEntered(MouseEvent e) {
        l.setText("Mouse Entered");
    }
    public void mouseExited(MouseEvent e) {
        l.setText("Mouse Exited");
    }
    public void mousePressed(MouseEvent e) {
        l.setText("Mouse Pressed");
    }
    public void mouseReleased(MouseEvent e) {
        l.setText("Mouse Released");
    }
    public static void main(String[] args) {
        new MouseListenerExample();
    }
}
```



Output:



Conclusion :-



# D.Y.Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name: Shiva Shree

PRN: 72018544M

Date:-

Roll No:-TEB237

### Experiment No. 3

#### Problem Statement: -

Develop a GUI which accepts the information regarding the marks for all the subjects of a student in the examination. Display the result for a student in a separate window.

#### Objectives: -

The objective of this assignment is to learn basics of GUI and different components used in GUI.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans

#### Theory :-

#### Java Swing Tutorial

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components.

The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

#### Difference between AWT and Swing

There are many differences between java awt and swing that are given below.

| No. | Java AWT                                     | Java Swing  |
|-----|--|---|
| 1)  | AWT components are platform-dependent.       | Java swing components are platform-independent.   |
| 2)  | AWT components are heavyweight.              | Swing components are lightweight.   |
| 3)  | AWT doesn't support pluggable look and feel. | Swing supports pluggable look and feel.   |
| 4)  | AWT provides less components than Swing.     | Swing provides more powerful components such as tables, lists, scrollpanes, colorchooser, tabbedPane etc. |

|    |   |                    |
|----|---|--------------------|
| 5) | AWT doesn't follow MVC(Model View Controller) where model represents data, view represents presentation and controller acts as an interface between model and view. | Swing follows MVC. |
|----|---|--------------------|

The Java Foundation Classes (JFC) are a set of GUI components which simplify the development of desktop applications.

#### Commonly used Methods of Component class

The methods of Component class are widely used in java swing that are given below.

| Method                                    | Description   |
|---|---|
| public void add(Component c)              | add a component on another component.                         |
| public void setSize(int width,int height) | sets size of the component.                                   |
| public void setLayout(LayoutManager m)    | sets the layout manager for the component.                    |
| public void setVisible(boolean b)         | sets the visibility of the component. It is by default false. |

There are two ways to create a frame:

By creating the object of Frame class (association)

By extending Frame class (inheritance)

We can write the code of swing inside the main(), constructor or any other method.

#### Simple Java Swing Example

Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

#### File: FirstSwingExample.java

```
import javax.swing.*;

public class FirstSwingExample {
    public static void main(String[] args) {
        JFrame f=new JFrame();//creating instance of JFrame

        JButton b=new JButton("click");//creating instance of JButton
        b.setBounds(130,100,100, 40);//x axis, y axis, width, height

        f.add(b);//adding button in JFrame
    }
}
```

```
f.setSize(400,500);//400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
```

### Code:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.JApplet;
import javax.swing.*;

/**
 *
 * @author prajakta
 */

public class NewJApplet extends JApplet implements ActionListener{

    /**
     * Initialization method that will be called after the applet is loaded into
     * the browser.
     */
    JTextField T1,T2,T3,T4,T5,ans;
    JLabel L1, L2, L3, L4, L5;
    JButton res;
    @Override
    public void init() {
        // TODO start asynchronous download of heavy resources
        // JFrame f = new JFrame();
        Container contentPane = getContentPane();
        contentPane.setLayout(new FlowLayout(FlowLayout.CENTER));
        res = new JButton("Result");
        T1 = new JTextField(20);
        T2 = new JTextField(20);
        T3 = new JTextField(20);
        T4 = new JTextField(20);
        T5 = new JTextField(20);
        ans = new JTextField(20);
        L1 = new JLabel("Enter marks of AJP");
        L2 = new JLabel("Enter marks of PDC");
        L3 = new JLabel("Enter marks of CN");
        L4 = new JLabel("Enter marks of OOP");
        L5 = new JLabel("Enter marks of PM");
        contentPane.add(L1);
        contentPane.add(T1);
```

```

contentPane.add(L2);
contentPane.add(T2);
contentPane.add(L3);
contentPane.add(T3);
contentPane.add(L4);
contentPane.add(T4);
contentPane.add(L5);
contentPane.add(T5);
contentPane.add(res);
contentPane.add(ans);
res.addActionListener(this);
}

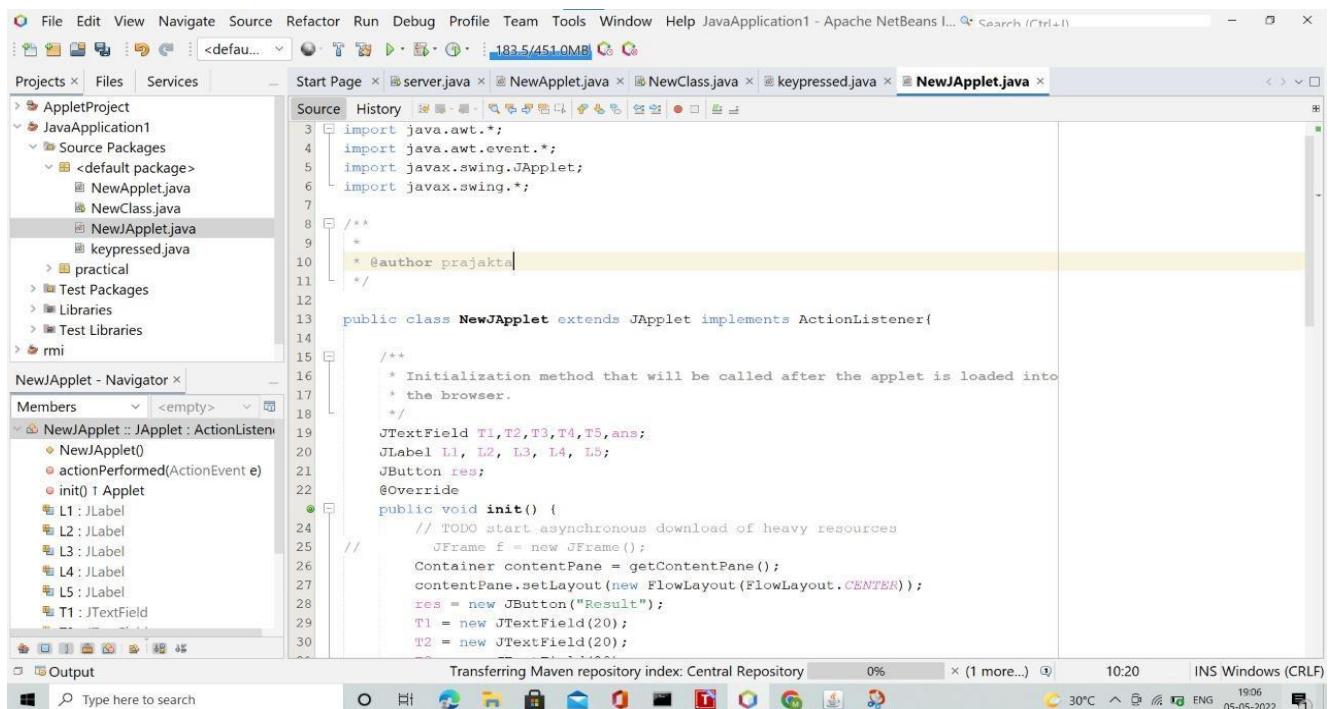
```

@Override

```

public void actionPerformed(ActionEvent e) {
    int m1 = Integer.parseInt(T1.getText());
    int m2 = Integer.parseInt(T2.getText());
    int m3 = Integer.parseInt(T3.getText());
    int m4 = Integer.parseInt(T4.getText());
    int m5 = Integer.parseInt(T5.getText());
    double result = ((m1+m2+m3+m4+m5)/500.0)*100;
    String format = String.format("%.2f", result);
    ans.setText(format);
}
}

```



Start Page x server.java " NewApplet.java x NewClass.java x keypressed.java x |||| NewJApplet.java x

vO

```
Source History IL</      IQ. 8 1! "  /  /      • □  5
u      IL      new UT extr i e L Q IL UJ ;
31      T3 = new JTextField(20);
32      T4 = new JTextField(20);
33      TS = new JTextField(20);
34      ans = new JTextField(20);
35      L1 = new JLabel("Enter marks of AJP");
36      L2 = new JLabel("Enter marks of PDC");
37      L3 = new JLabel("Enter marks of CN");
38      L4 = new JLabel("Enter marks of OOP");
39      L5 = new JLabel("Enter marks of PM");
40      contentPane.add(L1);
41      contentPane.add(L2);
42      contentPane.add(L3);
43      contentPane.add(L4);
44      contentPane.add(L5);
45      contentPane.add(T1);
46      contentPane.add(T2);
47      contentPane.add(T3);
48      contentPane.add(T4);
49      contentPane.add(T5);
50      contentPane.add(res);
51      contentPane.add(ans);
52      //      f.setSize(100,30);
53      //      f.setVisible(true);
54      //      f.addKeyListener(this);
55      res.addActionListener(this);
56
57
.s.a_      _!a1111P...r id l=<
```

Transferring Maven repository index: Central Repository

0%

x (1 more...) G;

70:1

INS

Start Page x ||||server.java x ||| NewApplet.java x ||| NewClass.java x ||| keypressed.java x |||1 NewJApplet.java x

```
Source History 39      f1'e,gt C: f 1.      • □  11
2      contentPane.add(L2);
43      contentPane.add(L3);
44      contentPane.add(L4);
45      contentPane.add(L5);
46      contentPane.add(T1);
47      contentPane.add(T2);
48      contentPane.add(T3);
49      contentPane.add(T4);
50      contentPane.add(T5);
51      contentPane.add(res);
52      contentPane.add(ans);
53      //      f.setSize(300,300);
54      //      f.setVisible(true);
55      //      f.addKeyListener(this);
56      res.addActionListener(this);
57
58      @Override
59      public void actionPerformed(ActionEvent e) {
60          int m1 = Integer.parseInt(T1.getText());
61          int m2 = Integer.parseInt(T2.getText());
62          int m3 = Integer.parseInt(T3.getText());
63          int m4 = Integer.parseInt(T4.getText());
64          int m5 = Integer.parseInt(T5.getText());
65          double result = (m1+m2+m3+m4+m5)/500*.0*100;
66          String format = String.format("%.2f", result);
67          ans.setText(format);
68
69
```

Transferring Maven repository index: Central Repository

0%

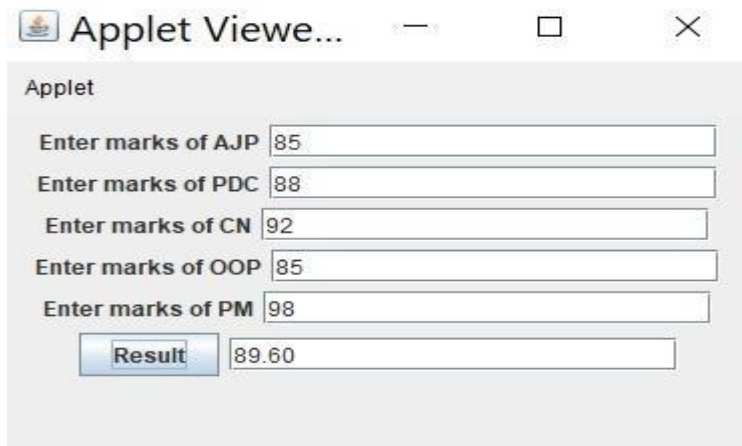
x (1 more...) Q;

70:1

INS



## OUTPUT



Applet

Enter marks of AJP 85

Enter marks of PDC 88

Enter marks of CN 92

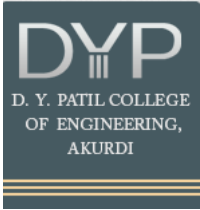
Enter marks of OOP 85

Enter marks of PM 98

Result 89.60

Applet started.

**Conclusion :-**



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE  
Name: Shiva Shree  
Date:-

Subject : Advanced Java programming  
PRN: 72018544M  
Roll No:-TEB237

---

### Experiment No. 4

#### Problem Statement: -

Write a program to insert and retrieve the data from the database using JDBC.

#### Objectives: -

The objective of this assignment is to learn the concepts of database handling.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans, Tomcat software, MySQL.

#### Theory :-

In general, to process any SQL statement with JDBC, you follow these steps:

- Establishing a connection.
- Create a statement.
- Execute the query.
- Process the ResultSet object.
- Close the connection.

```
public static void viewTable(Connection con) throws SQLException {
```

```
String query = "select COF_NAME, SUP_ID, PRICE, SALES, TOTAL from COFFEES";
```

```

try (Statement stmt = con.createStatement()) {

    ResultSet rs = stmt.executeQuery(query);

    while (rs.next()) {

        String coffeeName = rs.getString("COF_NAME");

        int supplierID = rs.getInt("SUP_ID");

        float price = rs.getFloat("PRICE");

        int sales = rs.getInt("SALES");

        int total = rs.getInt("TOTAL");

        System.out.println(coffeeName + ", " + supplierID + ", " +
            price + ", " + sales + ", " + total);

    }

} catch (SQLException e) {

    JDBCTutorialUtilities.printStackTrace(e);

}

}

```

- **Establishing Connections**

First, establish a connection with the data source you want to use. A data source can be a DBMS, a legacy file system, or some other source of data with a corresponding JDBC driver. This connection is represented by a Connection object.

- **Creating Statements**

A Statement is an interface that represents a SQL statement. You execute Statement objects, and they generate ResultSet objects, which is a table of data representing a database result set. You need a Connection object to create a Statement object.

For example, CoffeesTable.viewTable creates a Statement object with the following code:

```
stmt = con.createStatement();
```

There are three different kinds of statements:

**Statement:** Used to implement simple SQL statements with no parameters.

**PreparedStatement:** (Extends Statement.) Used for precompiling SQL statements that might contain input parameters.

**CallableStatement:** (Extends PreparedStatement.) Used to execute stored procedures that may contain both input and output parameters.

- **Executing Queries**

To execute a query, call an execute method from Statement such as the following:

**execute:** Returns true if the first object that the query returns is a ResultSet object. Use this method if the query could return one or more ResultSet objects. Retrieve the ResultSet objects returned from the query by repeatedly calling Statement.getResultSet.

**executeQuery:** Returns one ResultSet object.

**executeUpdate:** Returns an integer representing the number of rows affected by the SQL statement. Use this method if you are using INSERT, DELETE, or UPDATE SQL statements.

For example, CoffeesTable.viewTable executed a Statement object with the following code:

```
ResultSet rs = stmt.executeQuery(query);
```

- **Processing ResultSet Objects**

You access the data in a ResultSet object through a cursor. Note that this cursor is not a database cursor. This cursor is a pointer that points to one row of data in the ResultSet object. Initially, the cursor is positioned before the first row. You call various methods defined in the ResultSet object to move the cursor.

```
ResultSet rs = stmt.executeQuery(query);
```

```
while (rs.next()) {
```

```

String coffeeName = rs.getString("COF_NAME");

int supplierID = rs.getInt("SUP_ID");

float price = rs.getFloat("PRICE");

int sales = rs.getInt("SALES");

int total = rs.getInt("TOTAL");

System.out.println(coffeeName + ", " + supplierID + ", " + price +

                    ", " + sales + ", " + total);

}

// ...

```

- **Closing Connections**

When you are finished using a Connection, Statement, or ResultSet object, call its close method to immediately release the resources it's using.

Alternatively, use a try-with-resources statement to automatically close Connection, Statement, and ResultSet objects, regardless of whether an SQLException has been thrown. (JDBC throws an SQLException when it encounters an error during an interaction with a data source) An automatic resource statement consists of a try statement and one or more declared resources.

```

public static void viewTable(Connection con) throws SQLException {

    String query = "select COF_NAME, SUP_ID, PRICE, SALES, TOTAL from COFFEES";

    try (Statement stmt = con.createStatement()) {

        ResultSet rs = stmt.executeQuery(query);

        while (rs.next()) {

            String coffeeName = rs.getString("COF_NAME");

            int supplierID = rs.getInt("SUP_ID");

```

```

float price = rs.getFloat("PRICE");

int sales = rs.getInt("SALES");

int total = rs.getInt("TOTAL");

System.out.println(coffeeName + ", " + supplierID + ", " + price +

                    ", " + sales + ", " + total);

}

} catch (SQLException e) {

    JDBCTutorialUtilities.printSQLException(e);

}

}

```

### **Code:**

```

import java.sql.*;

public class MySQLdatabase {

    public static void main(String[] args) {

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

            Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/manisha",

"root", "Root2223");

            Statement s = con.createStatement();

            s.execute("create table student ( stud_id integer,stud_name varchar(20),stud_address

varchar(30) )"); // create a table

            s.execute("insert into student values(001,'ARman','Delhi')"); // insert first row into the

table

            s.execute("insert into student values(002,'Robert','Canada')"); // insert second row into the

table

            s.execute("insert into student values(003,'Ahuja','Karnal')"); // insert third row into the

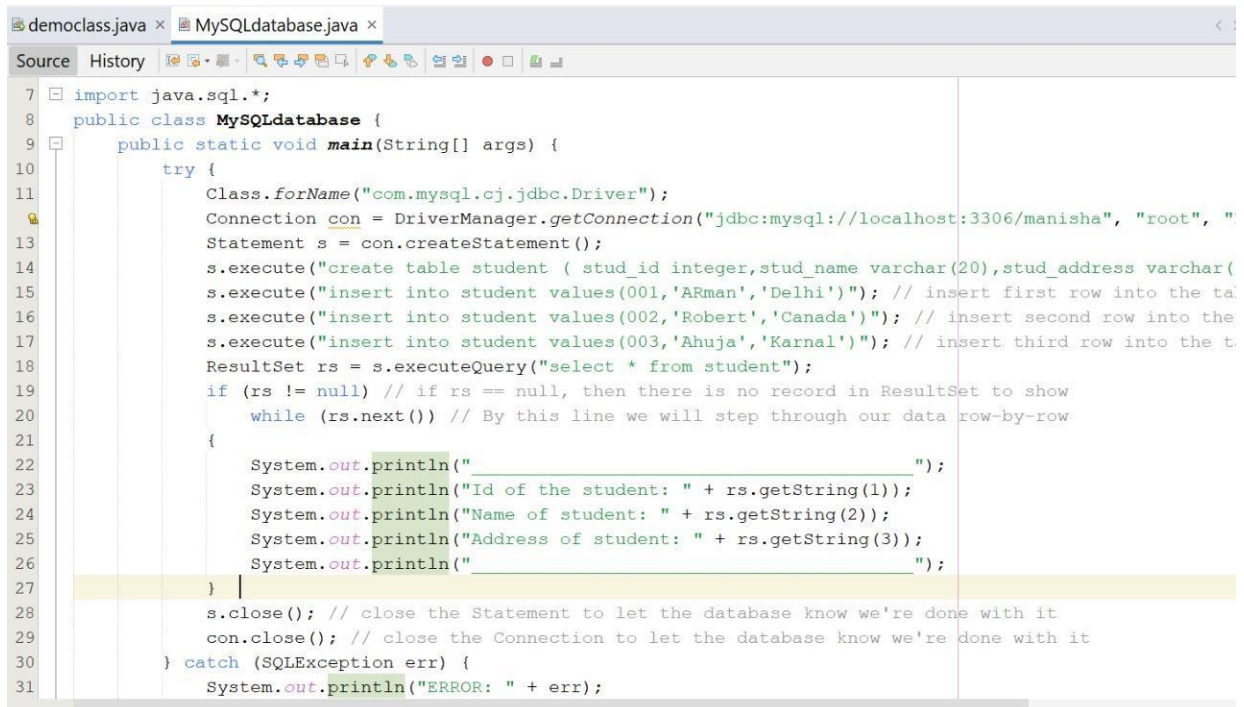
table

```

```

        ResultSet rs = s.executeQuery("select * from student");
        if (rs != null) // if rs == null, then there is no record in ResultSet to show
            while (rs.next()) // By this line we will step through our data row-by-row
            {
                System.out.println("_____");
                System.out.println("Id of the student: " + rs.getString(1));
                System.out.println("Name of student: " + rs.getString(2));
                System.out.println("Address of student: " + rs.getString(3));
                System.out.println("_____");
            }
        s.close(); // close the Statement to let the database know we're done with it
        con.close(); // close the Connection to let the database know we're done with it
    } catch (SQLException err) {
        System.out.println("ERROR: " + err);
    } catch (Exception err) {
        System.out.println("ERROR: " + err);
    }
}
}
}

```



The screenshot shows an IDE with two tabs: 'democlass.java' and 'MySQLdatabase.java'. The 'MySQLdatabase.java' tab is active, displaying the following code:

```

7  import java.sql.*;
8  public class MySQLdatabase {
9      public static void main(String[] args) {
10         try {
11             Class.forName("com.mysql.cj.jdbc.Driver");
12             Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/manisha", "root", "");
13             Statement s = con.createStatement();
14             s.execute("create table student ( stud_id integer,stud_name varchar(20),stud_address varchar(20))");
15             s.execute("insert into student values(001,'ARman','Delhi')"); // insert first row into the table
16             s.execute("insert into student values(002,'Robert','Canada')"); // insert second row into the table
17             s.execute("insert into student values(003,'Ahuja','Karnal')"); // insert third row into the table
18             ResultSet rs = s.executeQuery("select * from student");
19             if (rs != null) // if rs == null, then there is no record in ResultSet to show
20                 while (rs.next()) // By this line we will step through our data row-by-row
21                 {
22                     System.out.println("_____");
23                     System.out.println("Id of the student: " + rs.getString(1));
24                     System.out.println("Name of student: " + rs.getString(2));
25                     System.out.println("Address of student: " + rs.getString(3));
26                     System.out.println("_____");
27                 }
28             s.close(); // close the Statement to let the database know we're done with it
29             con.close(); // close the Connection to let the database know we're done with it
30         } catch (SQLException err) {
31             System.out.println("ERROR: " + err);

```

## Output - JavaApplication1 (run) ×



run:



---

Id of the student: 1

Name of student: ARman

Address of student: Delhi

---

---

Id of the student: 2

Name of student: Robert

Address of student: Canada

---

---

Id of the student: 3

Name of student: Ahuja

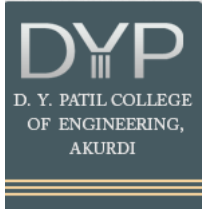
Address of student: Karnal

---

BUILD SUCCESSFUL (total time: 1 second)

**Conclusion :-**





# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Name: Shiva Shree

Date:-

Subject : Advanced Java programming

PRN: 72018544M

Roll No:-TEB237

---

### Experiment No. 5

#### Problem Statement: -

Develop an RMI application which accepts a string or a number and checks that string or number is palindrome or not.

#### Objectives: -

The objective of this assignment is to learn the concepts of Remote Method Invocation.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans, Tomcat software, MySQL.

#### Theory :-

RMI (Remote Method Invocation)

The **RMI** (Remote Method Invocation) is an API that provides a mechanism to create distributed application in java. The RMI allows an object to invoke methods on an object running in another JVM.

The RMI provides remote communication between the applications using two objects stub and skeleton.

Understanding stub and skeleton

RMI uses stub and skeleton object for communication with the remote object.

A **remote object** is an object whose method can be invoked from another JVM. Let's understand the stub and skeleton objects:

stub

The stub is an object, acts as a gateway for the client side. All the outgoing requests are routed through it. It resides at the client side and represents the remote object. When the caller invokes method on the stub object, it does the following tasks:

It initiates a connection with remote Virtual Machine (JVM),

It writes and transmits (marshals) the parameters to the remote Virtual Machine (JVM),

It waits for the result

It reads (unmarshals) the return value or exception, and

It finally, returns the value to the caller.

skeleton

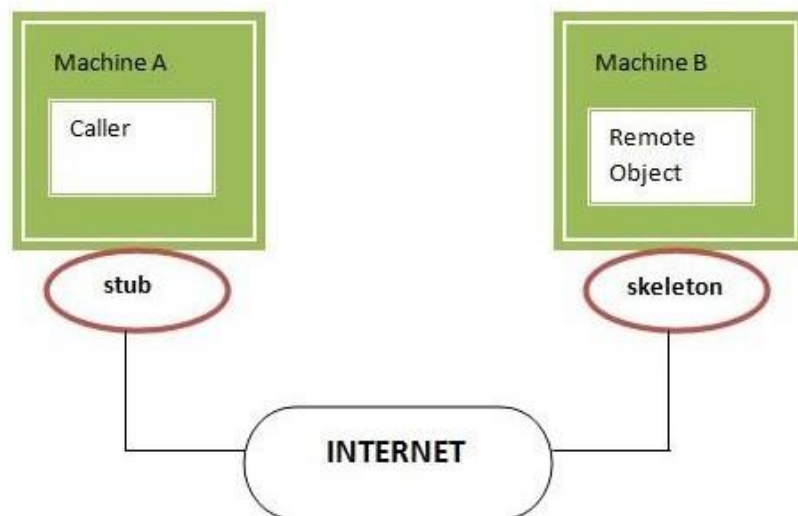
The skeleton is an object, acts as a gateway for the server side object. All the incoming requests are routed through it. When the skeleton receives the incoming request, it does the following tasks:

It reads the parameter for the remote method

It invokes the method on the actual remote object, and

It writes and transmits (marshals) the result to the caller.

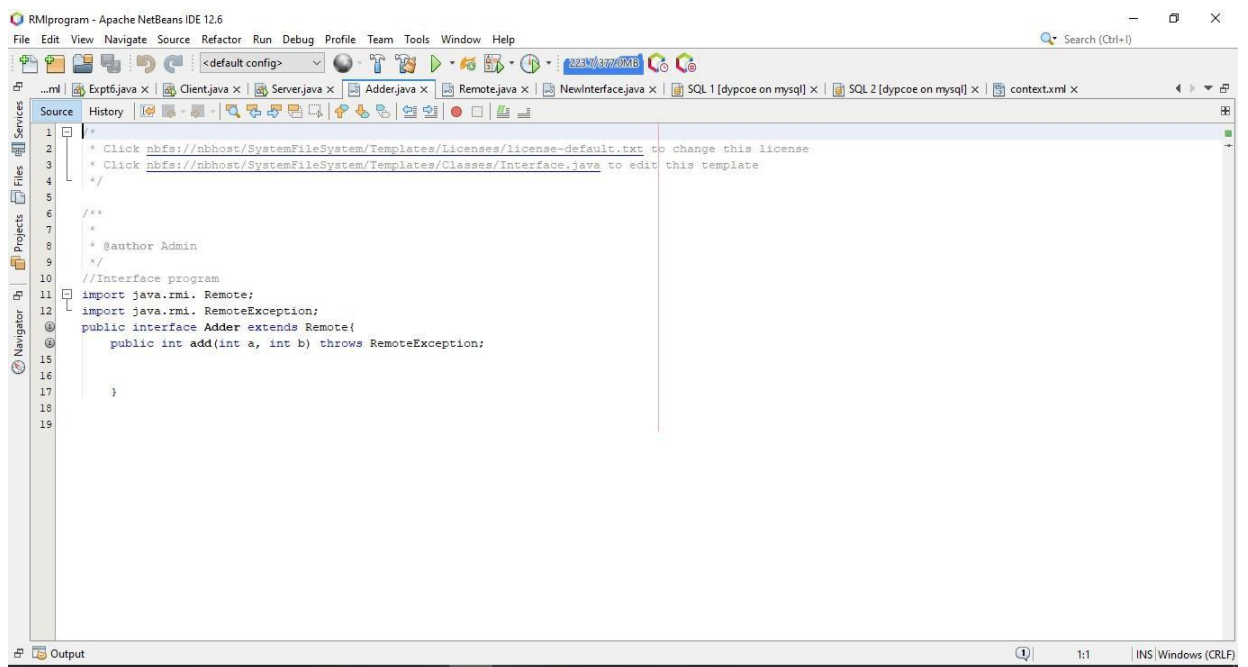
In the Java 2 SDK, an stub protocol was introduced that eliminates the need for skeletons.

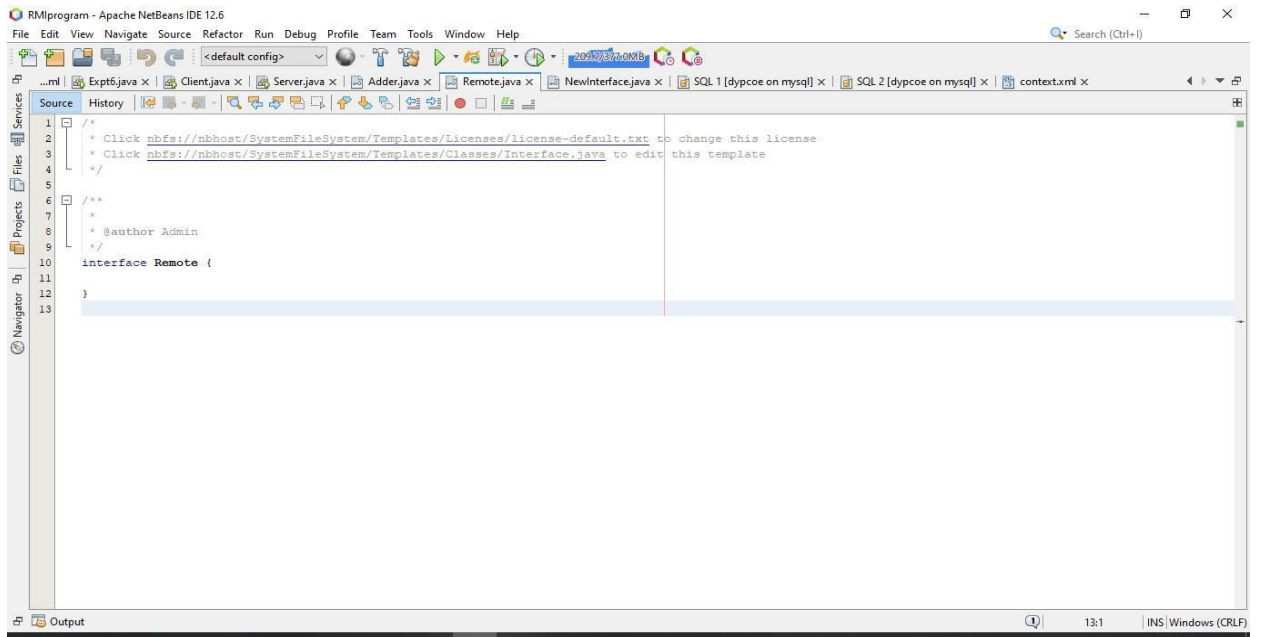


## Process of running RMI program:

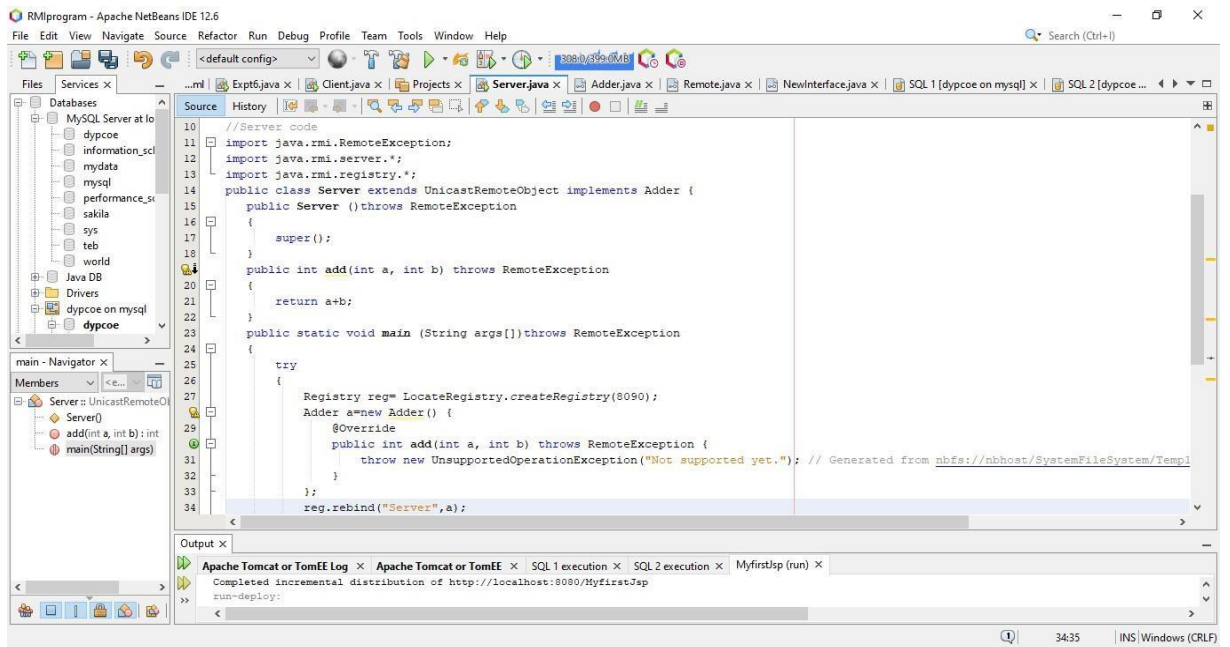
Following are the steps to write the RMI program.

1. Create the remote interface
2. Provide the implementation of the remote interface
3. Compile the implementation class and create the stub and skeleton objects using the rmic tool
4. Start the registry service by rmiregistry tool
5. Create and start the remote application
6. Create and start the client application

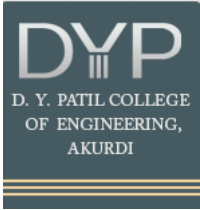




## Server code



## Conclusion:



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name: Shiva Shree

PRN: 72018544M

Date:-

Roll No:-TEB237

---

### Experiment No. 6

#### Problem Statement: -

Write a program to demonstrate the use of InetAddress class and its factory methods.

#### Objectives: -

The objective of this assignment is to learn the concepts of InetAddress.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans, Tomcat software

#### Theory :-

##### java.net.InetAddress Class in Java:

##### public class InetAddress extends Object implements Serializable:

The java.net.InetAddress class provides methods to get the IP address of any hostname. An IP address is represented by 32-bit or 128-bit unsigned number. InetAddress can handle both IPv4 and IPv6 addresses.

There are 2 types of addresses :

Unicast — An identifier for a single interface.

Multicast — An identifier for a set of interfaces.

InetAddress – Factory Methods :

The InetAddress class is used to encapsulate both, the numerical IP address and the domain name for that address. The InetAddress class has no visible constructors. The InetAddress class has the inability to create objects directly, hence factory methods are used for the purpose. Factory Methods are static methods in a class that return an object of that class.

There are 5 factory methods available in InetAddress class –

| Method   | Description   |
|--|---|
| static InetAddress getLocalHost() throws UnknownHostException                                    | This method returns the instance of InetAddress containing the local hostname and address.      |
| public static InetAddress getByName( String host ) throws UnknownHostException                   | This method returns the instance of InetAddress containing LocalHost IP and name.               |
| static InetAddress[] getAllByName( String hostName ) throws UnknownHostException                 | This method returns the array of the instance of InetAddress class which contains IP addresses. |
| static InetAddress getByAddress( byte IPAddress[] ) throws UnknownHostException                  | This method returns an InetAddress object created from the raw IP address.                      |
| static InetAddress getByAddress( String hostName, byte IPAddress[] ) throws UnknownHostException | This method creates and returns an InetAddress based on the provided hostname and IP address.   |

Java implementation of InetAddress class to demonstrate the use of factory methods –

```
import java.io.*;
import java.net.*;
import java.util.*;

class GFG {
    public static void main(String[] args)
        throws UnknownHostException
```

```

{
    // To get and print InetAddress of Local Host
    InetAddress address1 = InetAddress.getLocalHost();
    System.out.println("InetAddress of Local Host : "
        + address1);

    // To get and print InetAddress of Named Host
    InetAddress address2
        = InetAddress.getByName("45.22.30.39");
    System.out.println("InetAddress of Named Host : "
        + address2);

    // To get and print ALL InetAddresses of Named Host
    InetAddress address3[]
        = InetAddress.getAllByName("172.19.25.29");
    for (int i = 0; i < address3.length; i++) {
        System.out.println(
            "ALL InetAddresses of Named Host : "
            + address3[i]);
    }

    // To get and print InetAddresses of
    // Host with specified IP Address
    byte IPAddress[] = { 125, 0, 0, 1 };
    InetAddress address4
        = InetAddress.getByAddress(IPAddress);
    System.out.println(
        "InetAddresses of Host with specified IP Address : "
        + address4);

    // To get and print InetAddresses of Host
    // with specified IP Address and hostname

```

```

byte[] IPAddress2
    = { 105, 22, (byte)223, (byte)186 };
InetAddress address5 = InetAddress.getByAddress(
    "gfg.com", IPAddress2);
System.out.println(
    "InetAddresses of Host with specified IP Address and hostname : "
    + address5);
}
}

```

### Output

InetAddress of Local Host : localhost/127.0.0.1

InetAddress of Named Host : /45.22.30.39

ALL InetAddresses of Named Host : /172.19.25.29

InetAddresses of Host with specified IP Address : /125.0.0.1

InetAddresses of Host with specified IP Address and hostname : gfg.com/105.22.223.186

### InetAddress — Instance Methods :

InetAddress class has plenty of instance methods that can be called using the object. The instance methods are –

| Method                 | Description  |
|------------------------|--|
| equals(Object obj)     | This function compares this object against the specified object.             |
| getAddress()           | This method returns the raw IP address of this InetAddress object, in bytes. |
| getCanonicalHostName() | This method returns the fully qualified domain name for this IP address.     |



| Method                            | Description   |
|-----------------------------------|---|
| <code>getHostAddress()</code>     | This method gets the IP address in string form.   |
| <code>getHostName()</code>        | This method returns the host name for this IP address.  |
| <code>hashCode()</code>           | This method gets a hashcode for this IP address.  |
| <code>isAnyLocalAddress()</code>  | This method utility routine to check if the <code>InetAddress</code> is an unpredictable address. |
| <code>isLinkLocalAddress()</code> | This method utility routine to check if the address is not linked to local unicast address.       |
| <code>isLoopbackAddress()</code>  | This method used to check if the <code>InetAddress</code> represents a loopback address.          |
| <code>isMCGlobal()</code>         | This method utility routine check if this address has a multicast address of global scope.        |
| <code>isMCLinkLocal()</code>      | This method utility routine check if this address has a multicast address of link-local scope.    |
| <code>isMCNodeLocal()</code>      | This method utility routine check if the multicast address has node scope.                        |
| <code>isMCOrgLocal()</code>       | This method utility routine check if the multicast address has an organization scope.             |
| <code>isMCSiteLocal()</code>      | This method utility routine check if the multicast address has site scope.                        |

| Method  | Description   |
|---|---|
| isMulticastAddress()                                      | This method checks whether the site has multiple servers.                     |
| isReachable(int timeout)                                  | This method tests whether that address is reachable.                          |
| isReachable(NetworkInterface netif, int ttl, int timeout) | This method tests whether that address is reachable.                          |
| isSiteLocalAddress()                                      | This method utility routine check if the InetAddress is a site-local address. |
| toString()  | This method converts and returns an IP address in string form.                |

## Input

```

package expt6;

import java.io.*;
import java.net.*;
import java.util.*;

/**
 *
 * @author Admin
 */
public class Expt6 {

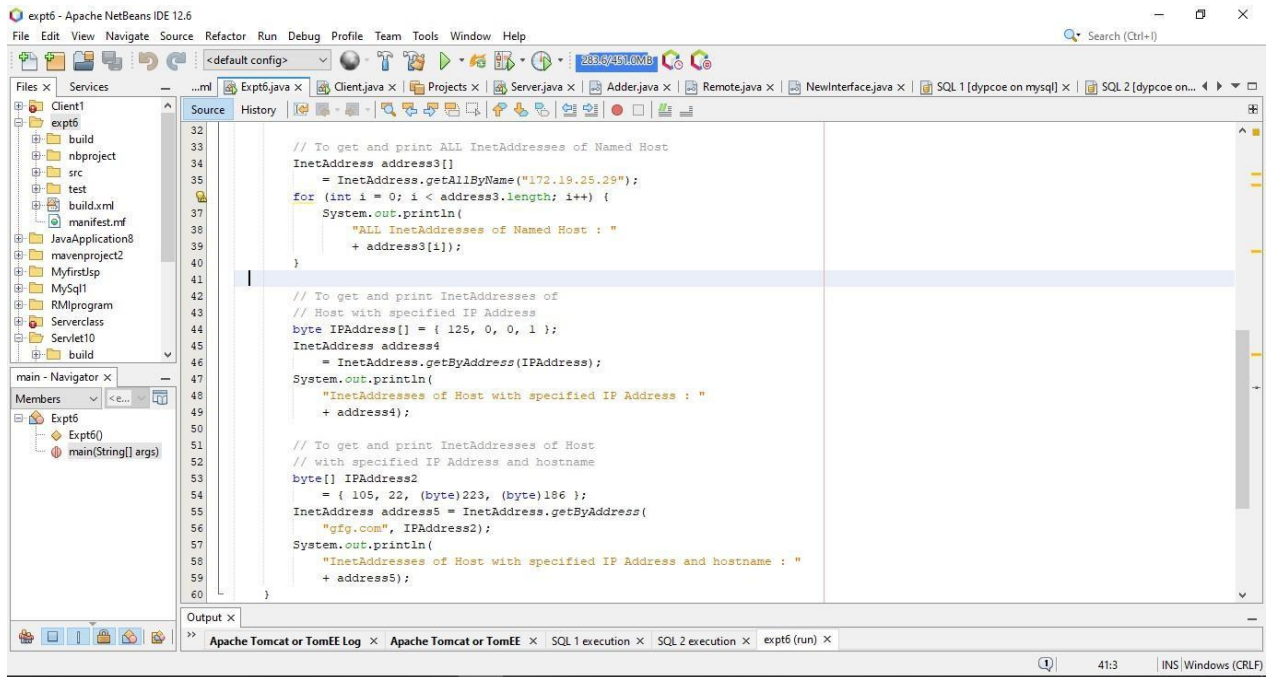
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws UnknownHostException {
        // TODO code application logic here

        // To get and print InetAddress of Local Host
        InetAddress address1 = InetAddress.getLocalHost();
        System.out.println("InetAddress of Local Host : "
            + address1);

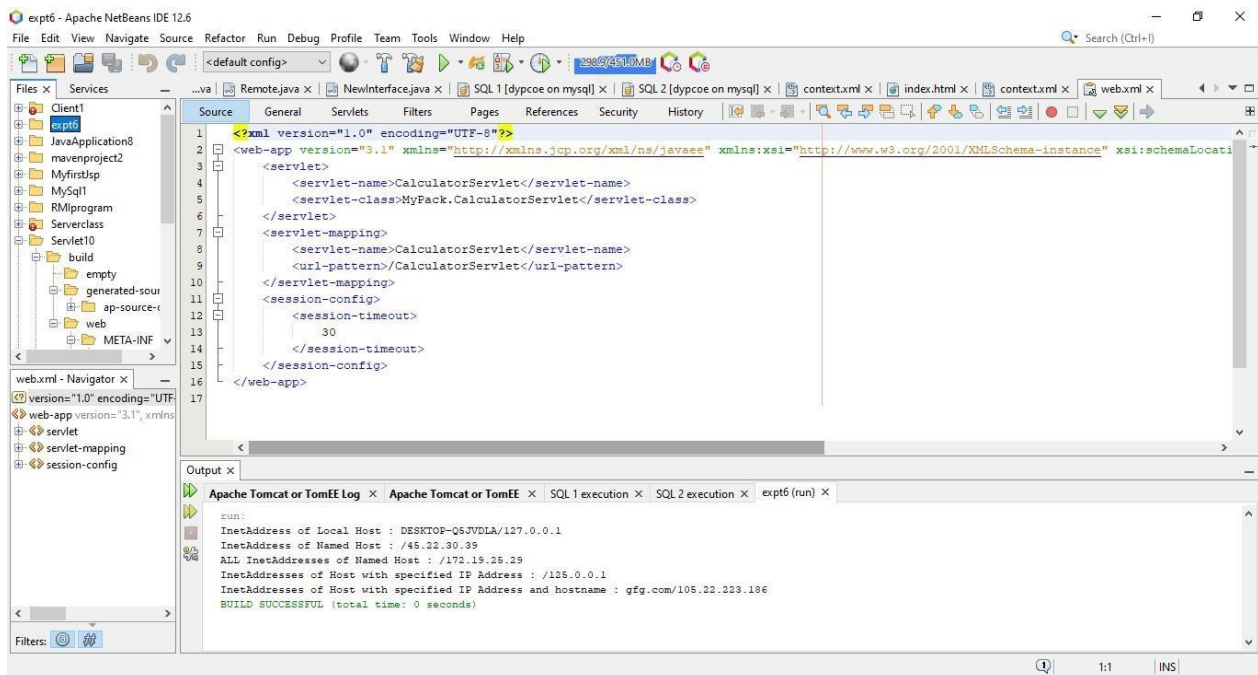
        // To get and print InetAddress of Named Host
        InetAddress address2
            = InetAddress.getByName("45.22.30.35");
        System.out.println("InetAddress of Named Host : "
            + address2);

        // To get and print ALL InetAddresses of Named Host
    }
}

```



## Output



## Conclusion:



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name: Shiva Shree

PRN: 72018544M

Date:-

Roll No:-TEB237

---

### Experiment No. 7

#### Problem Statement: -

- A. Write Servlet (procedure for client side) to display the username and password accepted from the client.
- B. Write Servlet (procedure for server side) to display the username and password accepted from the client.

#### Objectives: -

The objective of this assignment is to learn the concepts of Servlet.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans, Tomcat software

#### Theory :-

**Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).

**Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.

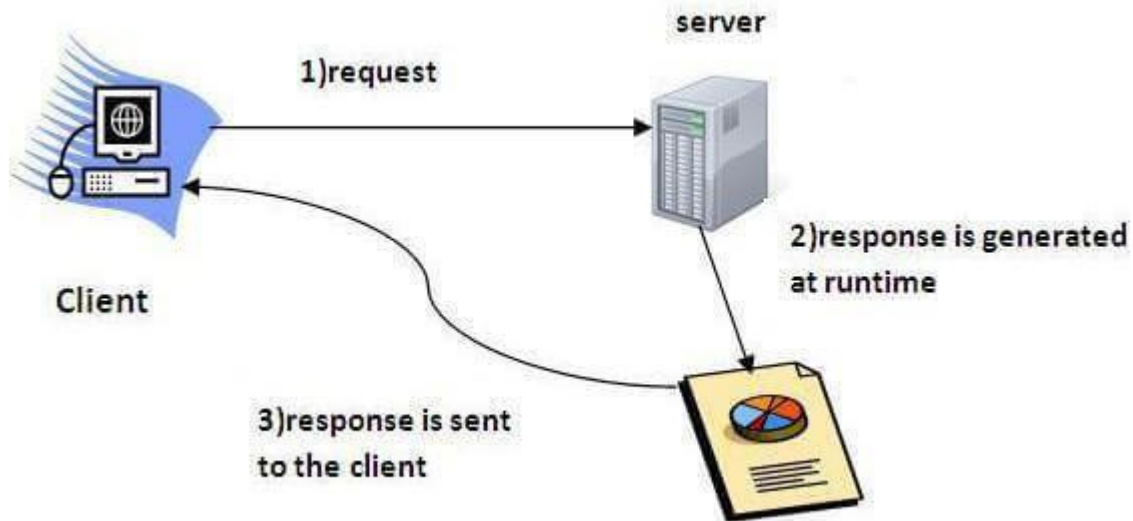
There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

#### What is a Servlet?

Servlet can be described in many ways, depending on the context.

- 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.

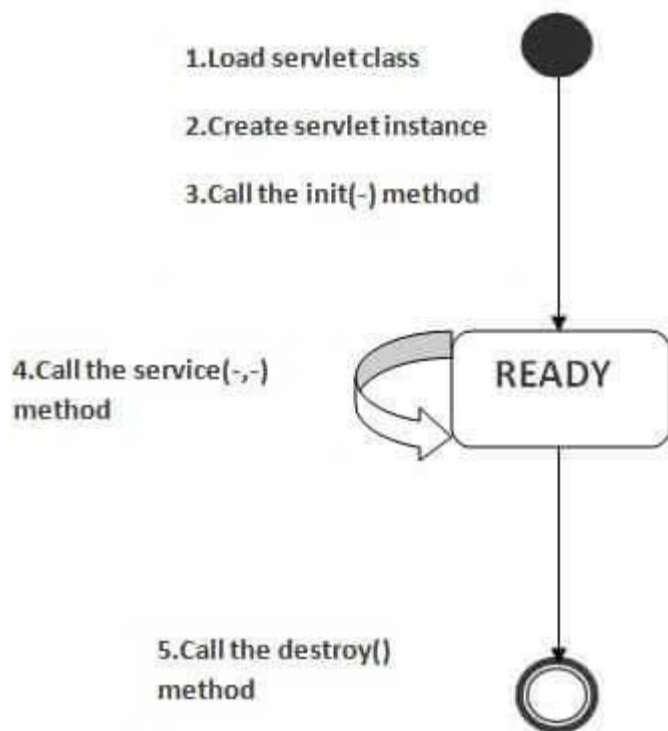


### **Life Cycle of a Servlet (Servlet Life Cycle)**

Servlet class is loaded

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

- Servlet class is loaded.
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.



As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the `init()` method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the `destroy()` method, it shifts to the end state.

### Code :

index.html

```
<form action="servlet1" method="post">
Name:<input type="text" name="username"/><br/><br/>
Password:<input type="password" name="userpass"/><br/><br/>
<input type="submit" value="login"/>
</form>
```

FirstServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
```

```

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class FirstServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        String n=request.getParameter("username");
        String p=request.getParameter("userpass");
        if(LoginDao.validate(n, p)){
            RequestDispatcher rd=request.getRequestDispatcher("servlet2");
            rd.forward(request,response);
        }
        else{
            out.print("Sorry username or password error");
            RequestDispatcher rd=request.getRequestDispatcher("index.html");
            rd.include(request,response);
        }
        out.close();
    } }

```

### **Login.java**

```

import java.sql.*;
public class LoginDao {
    public static boolean validate(String name,String pass){
        boolean status=false;
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection con=DriverManager.getConnection(
                "jdbc:oracle:thin:@localhost:1521:xe","system","oracle");

            PreparedStatement ps=con.prepareStatement(

```

```
"select * from userreg where name=? and pass=?");  
ps.setString(1,name);  
ps.setString(2,pass);  
ResultSet rs=ps.executeQuery();  
status=rs.next();  
} catch(Exception e){System.out.println(e);}  
return status;  
} }
```

### **WelcomeServlet.java**

```
import java.io.IOException;  
import java.io.PrintWriter;  
import javax.servlet.ServletException;  
import javax.servlet.http.HttpServlet;  
import javax.servlet.http.HttpServletRequest;  
import javax.servlet.http.HttpServletResponse;  
public class WelcomeServlet extends HttpServlet {  
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        response.setContentType("text/html");  
        PrintWriter out = response.getWriter();  
        String n=request.getParameter("username");  
        out.print("Welcome "+n);  
        out.close();  
    } }
```

Output:



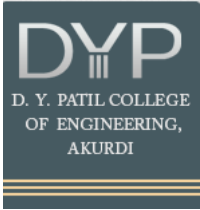
TODO supply a title

file:///C:/Users/manis/Documents/NetBeansProjects/JavaApplication1/src/newpackage/newhtml.html

Name:

Password:

**Conclusion:**



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Name: Shiva Shree

Date:-

Subject : Advanced Java programming

PRN: 72018544M

Roll No:-TEB237

---

### Experiment No. 8

#### Problem Statement: -

Write a database application that uses any JDBC driver.

#### Objectives: -

The objective of this assignment is to learn the concepts of database management with JDBC driver.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans, Tomcat software, MySQL

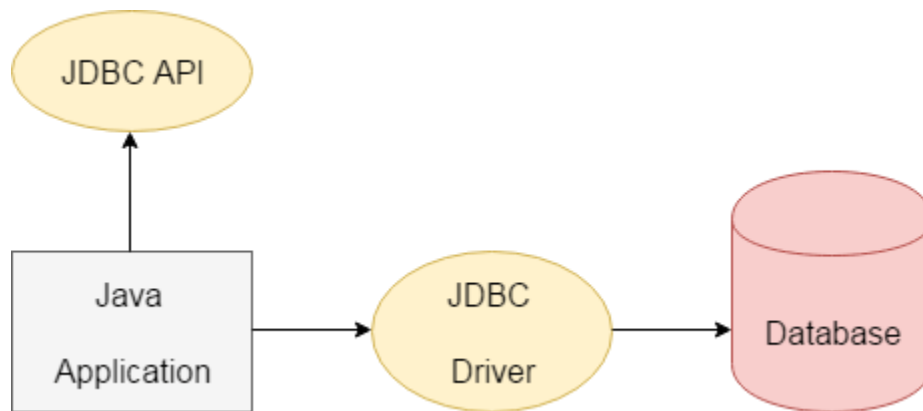
#### Theory :-

##### JDBC

JDBC stands for Java Database Connectivity. JDBC is a Java API to connect and execute the query with the database. It is a part of JavaSE (Java Standard Edition). JDBC API uses JDBC drivers to connect with the database. There are four types of JDBC drivers:

- JDBC-ODBC Bridge Driver,
- Native Driver,
- Network Protocol Driver, and
- Thin Driver

We can use JDBC API to access tabular data stored in any relational database. By the help of JDBC API, we can save, update, delete and fetch data from the database. It is like Open Database Connectivity (ODBC) provided by Microsoft.



The current version of JDBC is 4.3. It is the stable release since 21st September, 2017. It is based on the X/Open SQL Call Level Interface. The **java.sql** package contains classes and interfaces for JDBC API. A list of popular *interfaces* of JDBC API are given below:

- Driver interface
- Connection interface
- Statement interface
- PreparedStatement interface
- CallableStatement interface
- ResultSet interface
- ResultSetMetaData interface
- DatabaseMetaData interface
- RowSet interface

Before JDBC, ODBC API was the database API to connect and execute the query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

We can use JDBC API to handle database using Java program and can perform the following activities:

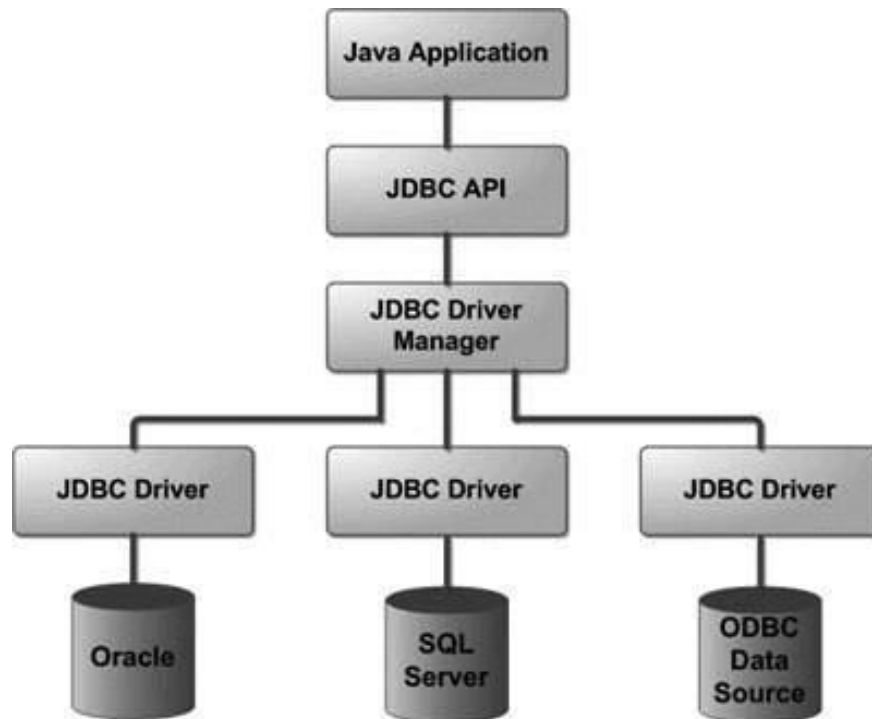
- Connect to the database
- Execute queries and update statements to the database

- Retrieve the result received from the database.

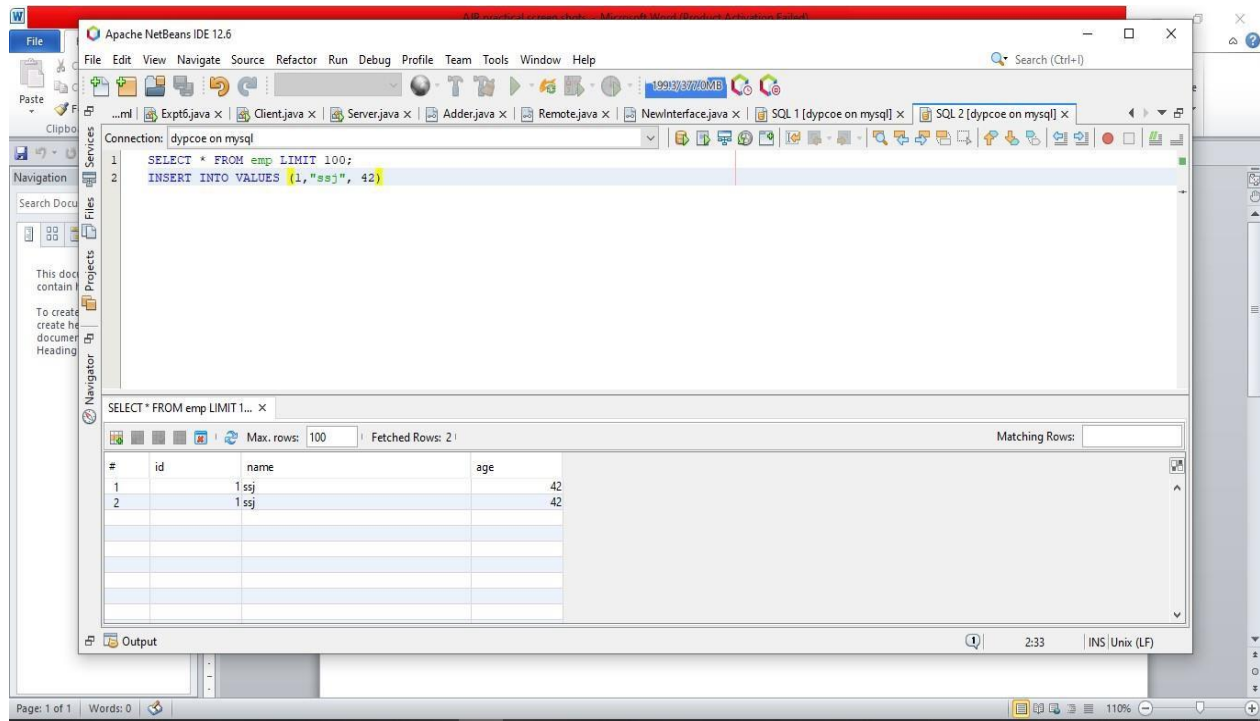
### JDBC Architecture:

A JDBC Driver is required to process the SQL requests and generate results. JDBC API provides classes and interfaces to handle database-specific calls from users.

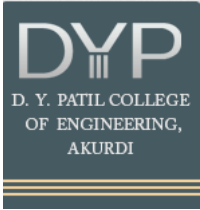
As shown in fig, the Java Application that needs to communicate with a database has to be programmed using JDBC API. The JDBC driver supporting data source, such as Oracle, SQL has to added in the Java Application for the JDBC Support, which can be done dynamically at run-time.



Program Screenshot



**Conclusion:**



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name: Shiva Shree

PRN: 72018544M

Date:-

Roll No:-TEB237

---

### Experiment No. 9

#### Problem Statement: -

Write a simple JSP page to display a simple message

#### Objectives: -

The objective of this assignment is to learn the concepts of JSP with simple example.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans

#### Theory:-

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

#### Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

##### 1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

##### 2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

##### 3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

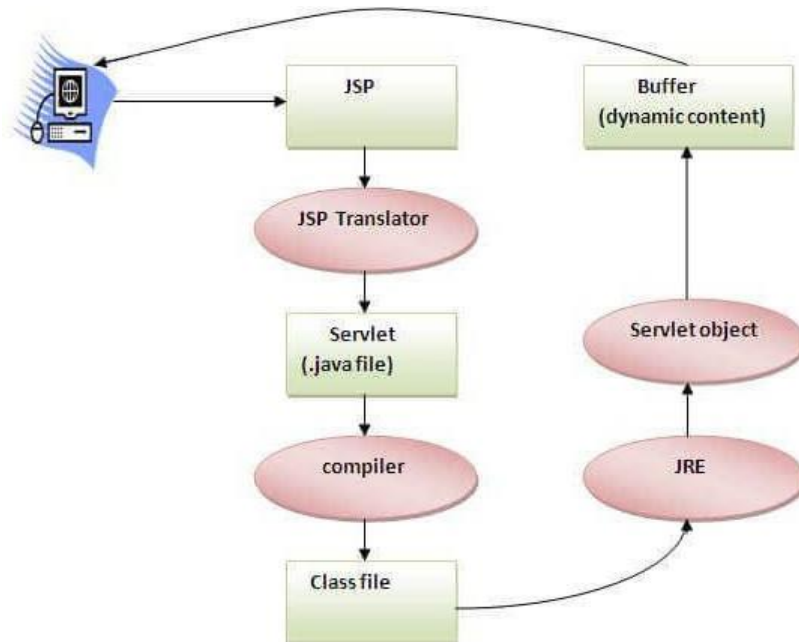
#### 4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

### The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization ( the container invokes `jspInit()` method).
- Request processing ( the container invokes `_jspService()` method).
- Destroy ( the container invokes `jspDestroy()` method).



As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

## Creating a simple JSP Page

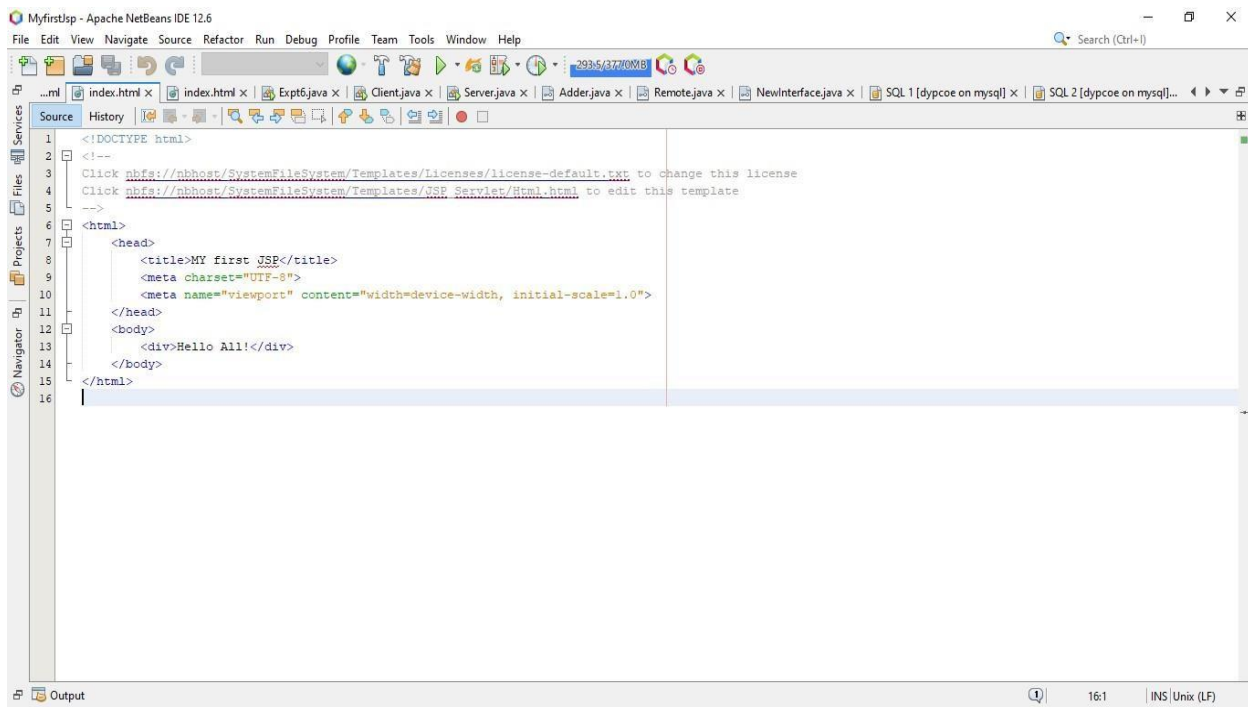
To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page.

## How to run a simple JSP Page?

Follow the following steps to execute this JSP page:

- Start the server
- Put the JSP file in a folder and deploy on the server
- Visit the browser by the URL `http://localhost:portno/contextRoot/jspfile`, for example, `http://localhost:8888/myapplication/index.jsp`

Input:



```
1 <!DOCTYPE html>
2 <!--
3 Click nbfs://pbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
4 Click nbfs://pbhost/SystemFileSystem/Templates/JSP_Servlet/Html.html to edit this template
5 -->
6 <html>
7 <head>
8 <title>MY first JSP</title>
9 <meta charset="UTF-8">
10 <meta name="viewport" content="width=device-width, initial-scale=1.0">
11 </head>
12 <body>
13 <div>Hello All!</div>
14 </body>
15 </html>
16
```

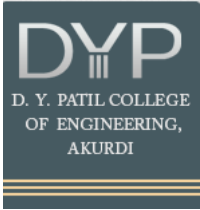


## Output:



---

## Conclusion:-



# D. Y. Patil College of Engineering, Akurdi, Pune

## Department of Electronics & Telecommunication Engineering

Class: TE

Subject : Advanced Java programming

Name: Shiva Shree

PRN: 72018544M

Date:-

Roll No:-TEB237

---

### Experiment No. 10

#### Problem Statement: -

Create a simple calculator application using servlet.

#### Objectives: -

The objective of this assignment is to learn the concepts of servlets and apply them to create a calculator.

#### Software used:-

64 bit Linux/Windows Operating System, JDK 1.4 or later, Eclipse/ Netbeans

#### Theory:-

**Servlet** technology is used to create a web application (resides at server side and generates a dynamic web page).

**Servlet** technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language. However, there were many disadvantages to this technology. We have discussed these disadvantages below.

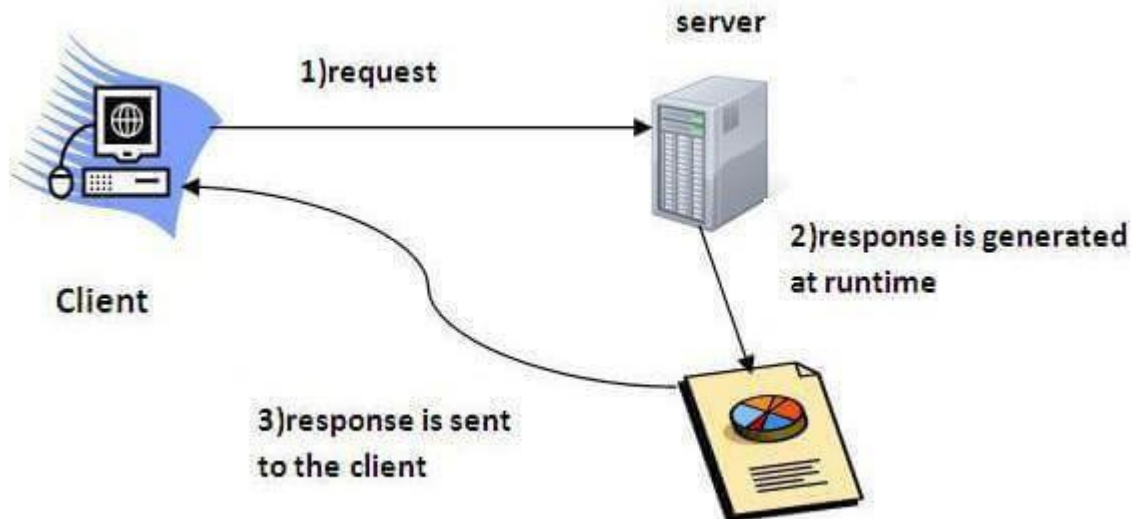
There are many interfaces and classes in the Servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse, etc.

#### What is a Servlet?

Servlet can be described in many ways, depending on the context.

- 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.

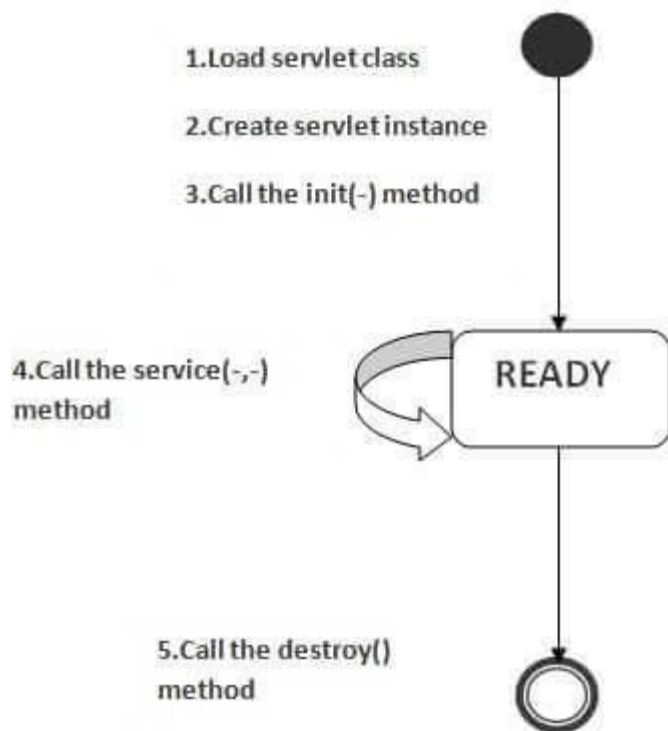


### **Life Cycle of a Servlet (Servlet Life Cycle)**

Servlet class is loaded

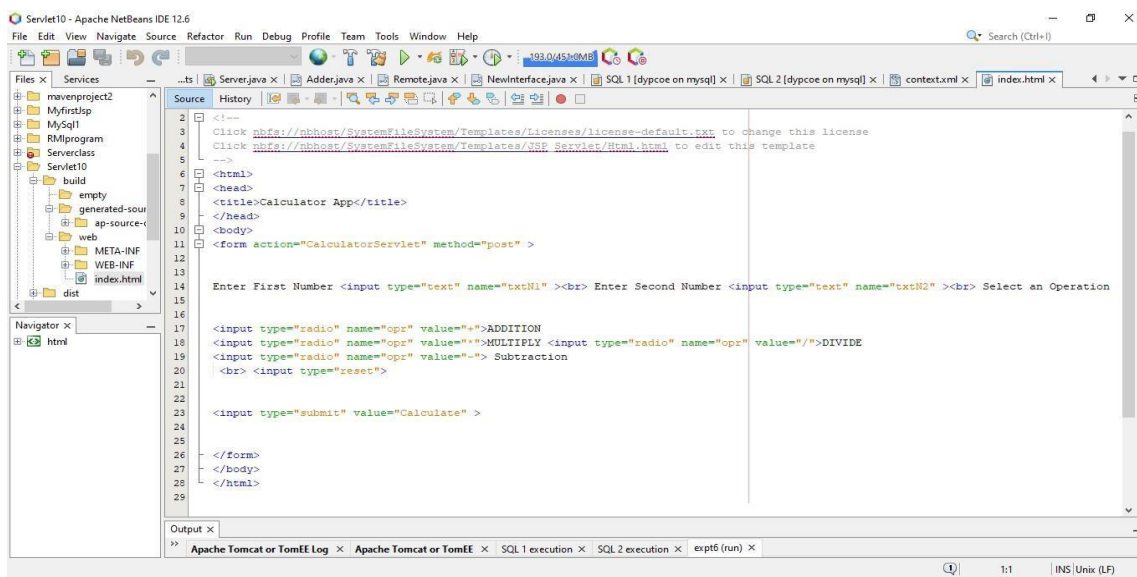
The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

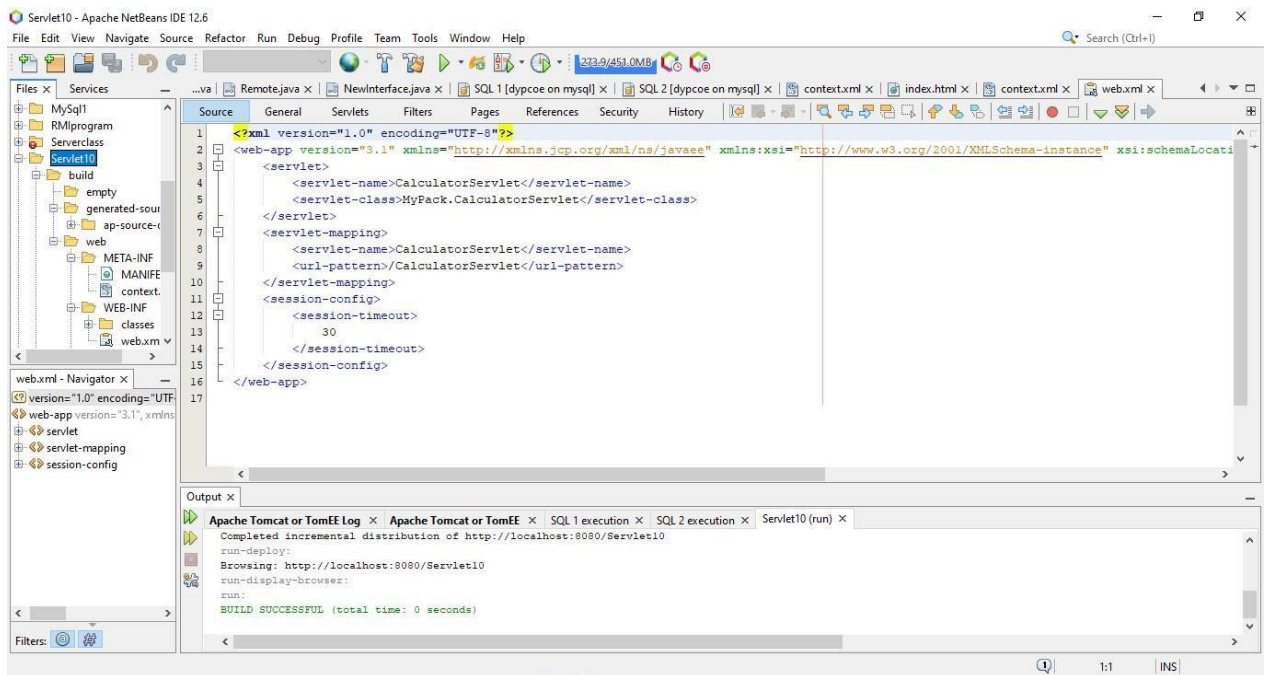
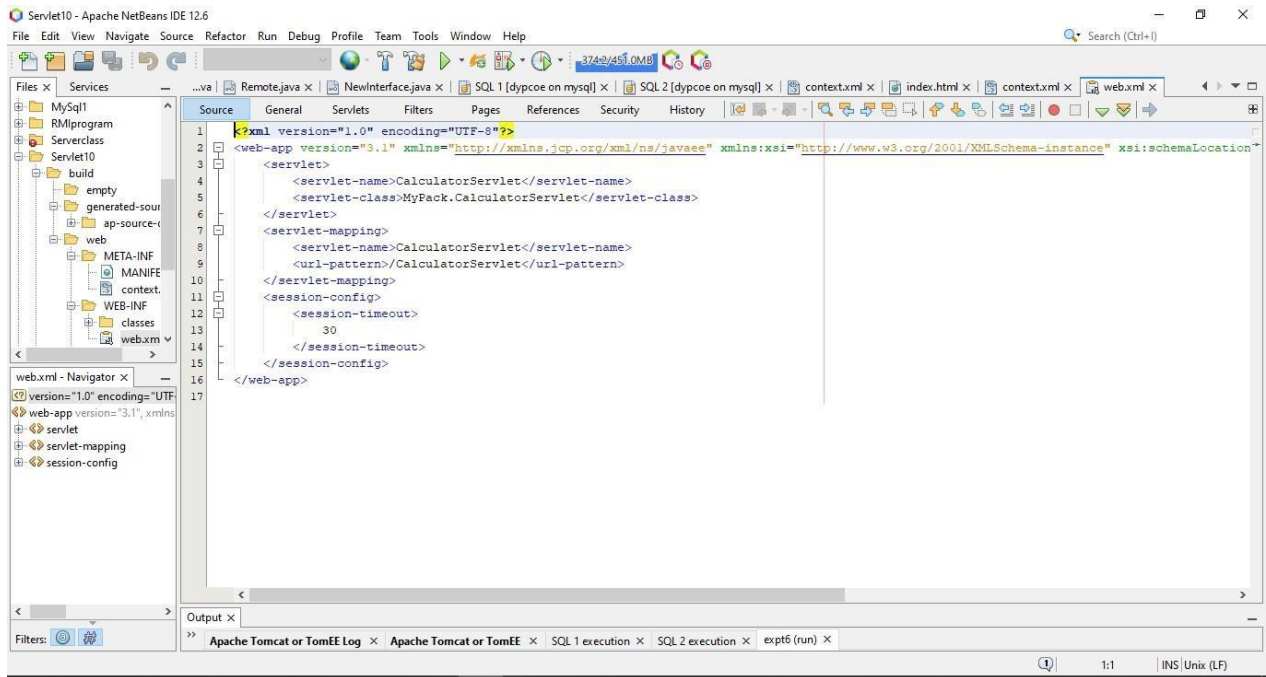
- Servlet class is loaded.
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.



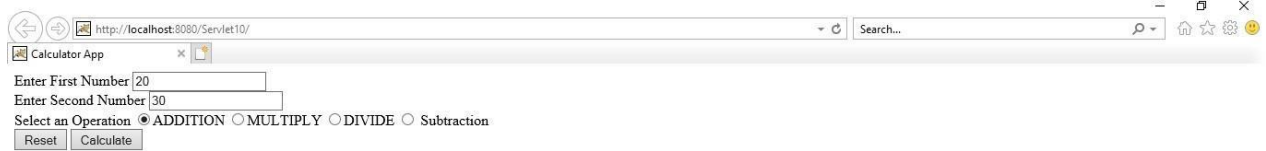
As displayed in the above diagram, there are three states of a servlet: new, ready and end. The servlet is in new state if servlet instance is created. After invoking the init() method, Servlet comes in the ready state. In the ready state, servlet performs all the tasks. When the web container invokes the destroy() method, it shifts to the end state.

Input:

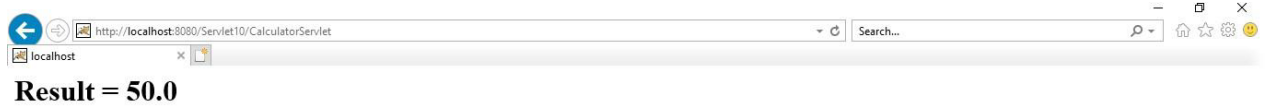




Output:



A screenshot of a web browser window. The address bar shows 'http://localhost:8080/Servlet10/'. The page title is 'Calculator App'. The form contains two input fields: 'Enter First Number' with the value '20' and 'Enter Second Number' with the value '30'. Below these is a radio button group for 'Select an Operation' with options: 'ADDITION' (selected), 'MULTIPLY', 'DIVIDE', and 'Subtraction'. At the bottom are two buttons: 'Reset' and 'Calculate'.



A screenshot of a web browser window. The address bar shows 'http://localhost:8080/Servlet10/CalculatorServlet'. The page title is 'localhost'. The main content of the page is 'Result = 50.0'.

**Conclusion:**