Experiment No.1

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

Objective:

To Handle Events of AWT and Swing Components.
To Develop programs to handle events in Java Programming.

Resources: Eclipse IDE 2018, JDK 1.8.0 is required

Course Outcome Addressed: CO2

Theory:

Event handling is fundamental to Java programming because it is used to create event driven programs eg • Applets • GUI based windows application • Web Application.

Event handling mechanism have been changed significantly between the original version of Java (1.0) and all subsequent versions of Java, beginning with version 1.1.

The modern approach to handling events is based on the delegation event model,

What is an Event?
Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event:
The events can be broadly classified into two categories:
Foreground Events - Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.

Background Events - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model. The Delegation Event Model has the following key participants namely: Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provides classes for source object. Listener - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns. Advantages of

Advantages of event Handling:

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

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:
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:

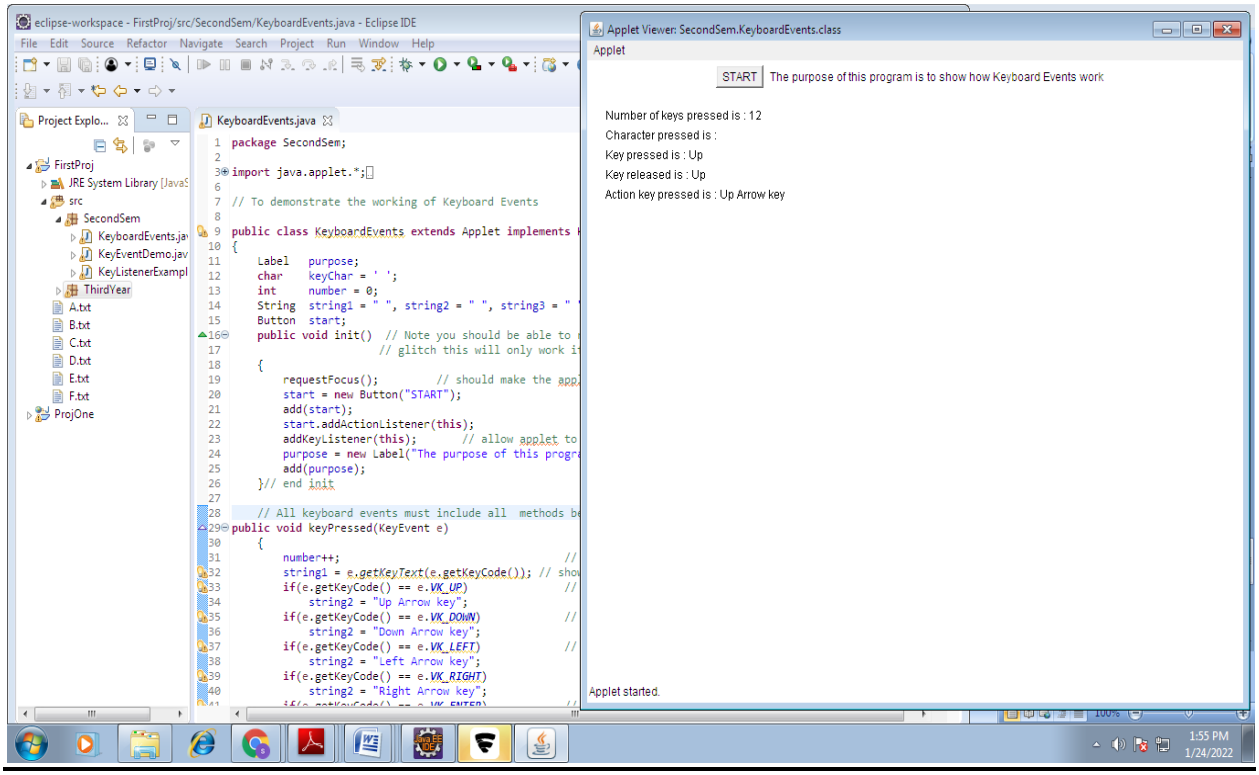
This interface inherits methods from the following interface:
java.awt.EventListener

SOURCE CODE:

```
package SecondSem;

import java.applet.*;
import java.awt.*;
import java.awt.event.*;

// To demonstrate the working of Keyboard Events

public class KeyboardEvents extends Applet implements KeyListener , ActionListener
{
        Label    purpose;
        char     keyChar = ' ';
        int              number = 0;
        String   string1 = " ", string2 = " ", string3 = " ", string4 = " ";
        Button   start;
        public void init()  // Note you should be able to requestFocus() in init (see below) but due to a
                            // glitch this will only work if a start  button press actually activates
         requestFocus()
        {
        requestFocus();          // should make the applet the active component for Key events
        start = new Button("START");
        add(start);
        start.addActionListener(this);
        addKeyListener(this);            // allow applet to process Key events
        purpose = new Label("The purpose of this program is to show how Keyboard Events work");
        add(purpose);
        }               // end init

        // All keyboard events must include all  methods below:
public void keyPressed(KeyEvent e)
        {
        number++;                                   // add 1 to count of key presses
        string1 = e.getKeyText(e.getKeyCode()); // show which key is pressed
        if(e.getKeyCode() == e.VK_UP)                   // see if UP arrow key is pressed
        string2 = "Up Arrow key";
        if(e.getKeyCode() == e.VK_DOWN)                 // see if DOWN arrow key is pressed
        string2 = "Down Arrow key";
        if(e.getKeyCode() == e.VK_LEFT)                 // see if LEFT arrow key is pressed
        string2 = "Left Arrow key";
        if(e.getKeyCode() == e.VK_RIGHT)                // see if Right arrow key is pressed
        string2 = "Right Arrow key";
        if(e.getKeyCode() == e.VK_ENTER)              // see if ENTER key is pressed
        string2 = "Enter key";
```

```
        repaint();

        }                                       // end keyPressed

        public void keyReleased(KeyEvent e)
        {
        string4 = e.getKeyText(e.getKeyCode());
        repaint();

        }                                         // end keyReleased


        public void keyTyped(KeyEvent e)
        {
        keyChar = e.getKeyChar();               // to get the actual Unicode character typed
        if(keyChar == 'x')
        string3 = "The key lower case x was pressed";
        repaint();

        }                                   // end keyTyped

        public void paint(Graphics g)
        {
        g.drawString("Number of keys pressed is : " + number, 20, 60);
        g.drawString("Character pressed is : " + keyChar, 20,80);
        g.drawString("Key pressed is : " + string1, 20, 100);
        g.drawString("Key released is : " + string4, 20, 120);
        g.drawString("Action key pressed is : " + string2, 20, 140);
        g.drawString(string3, 20, 160);

        }                                                   // end paint

        public void actionPerformed(ActionEvent e)
        {
                requestFocus();

        }                                                   // end actionPerformed

}                                                           // end KeyboardEvents
```

OUTPUT:



Conclusion:-

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1.Write algorithm of a program to demonstrate status of key on an Applet window such as KeyPressed, KeyReleased, KeyUp, KeyDown.
2. Name any four Event Listener interfaces.
3. State the situation when all three events of KeyListener interface are generated?
4.Elaborate the terms Event, Source and Listener.
5. List various methods of ActionListener interface.

Experiment No.2

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

Objective:

To Develop programs to handle events in Java Programming.

Resources:

Eclips IDE, JDK 1.8.0 is required

Course Outcome Addressed: CO2

Theory:

Event handling is fundamental to Java programming because it is used to create event driven programs eg • Applets • GUI based windows application • Web Application.

Event handling mechanism have been changed significantly between the original version of Java (1.0) and all subsequent versions of Java, beginning with version 1.1.

The modern approach to handling events is based on the delegation event model,

What is an Event?

Change in the state of an object is known as event i.e. event describes the change in state of source. Events are generated as result of user interaction with the graphical user interface components. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page are the activities that causes an event to happen.

Types of Event:

The events can be broadly classified into two categories:

Foreground Events - Those events which require the direct interaction of user. They are generated as consequences of a person interacting with the graphical components in Graphical User Interface. For example, clicking on a button, moving the mouse, entering a character through keyboard, selecting an item from list, scrolling the page etc.

Background Events - Those events that require the interaction of end user are known as background events. Operating system interrupts, hardware or software failure, timer expires, an operation completion are the example of background events.

What is Event Handling?

Event Handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when an event occurs. Java Uses the Delegation Event Model to handle the events. This model defines the standard mechanism to generate and handle the events. Let's have a brief introduction to this model. The Delegation Event Model has the following key participants namely: Source - The source is an object on which event occurs. Source is responsible for providing information of the occurred event to it's handler. Java provides classes for source object. Listener - It is also known as event handler. Listener is responsible for generating response to an event. From java implementation point of view the listener is also an object. Listener waits until it receives an event. Once the event is received , the listener process the event an then returns. Advantages of

Advantages of event Handling:

The benefit of this approach is that the user interface logic is completely separated from the logic that generates the event. The user interface element is able to delegate the processing of an event to the separate piece of code. In this model ,Listener needs to be registered with the source object so that the listener can receive the event notification. This is an efficient way of handling the event because the event notifications are sent only to those listener that want to receive them.

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:

1. public abstract void mouseClicked(MouseEvent e);
2. public abstract void mouseEntered(MouseEvent e);
3. public abstract void mouseExited(MouseEvent e);
4. public abstract void mousePressed(MouseEvent e);
5. public abstract void mouseReleased(MouseEvent e);

Program Logic:

1. Start the program

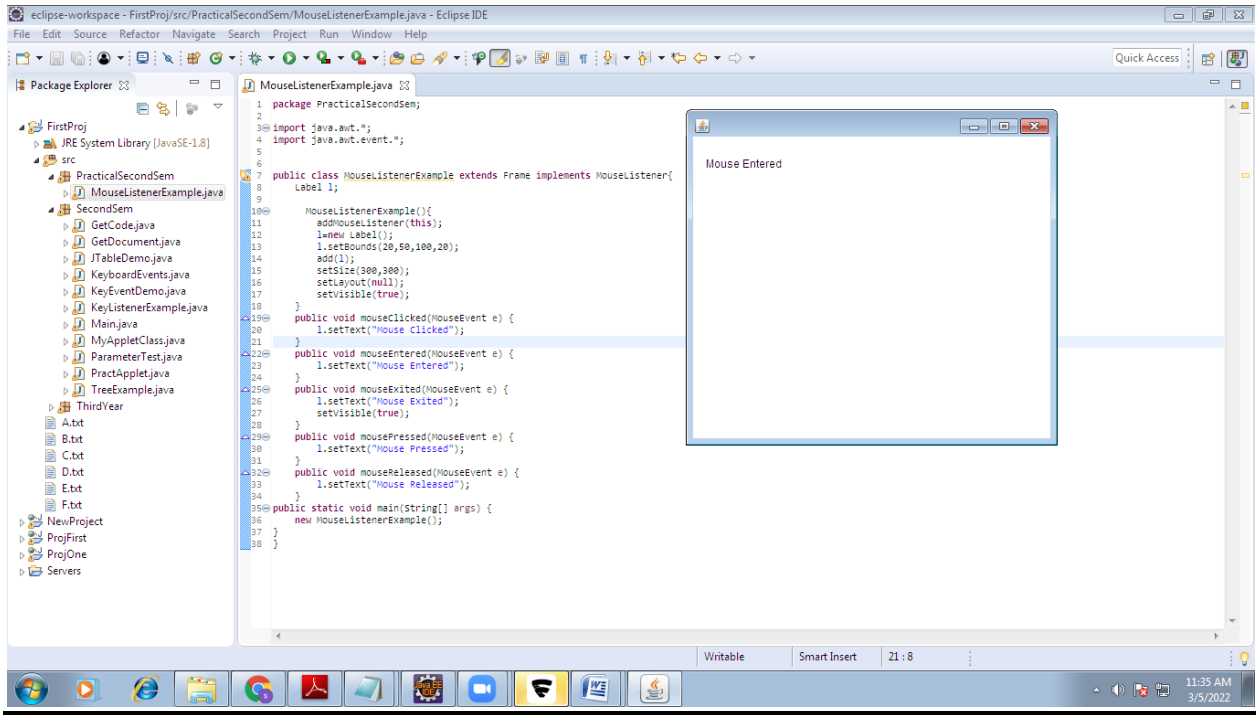
2. Create a class with the name: MouseListenerExample extends Frame implements MouseListener

3.

SOURCE CODE:

```
package PracticalSecondSem;
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");
      setVisible(false);
   }
   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:-

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Practical Related Questions

1. Write algorithm of a program to create a frame using AWT. Implement mouseClicked, mouseEntered() and mouseExited() events. Frame should become visible when the mouse enters it.
2. List various methods of MouseListener and MouseMotionListener
3. Do all components generate the MouseEvent
4. Write the steps to obtain the coordinates of MouseClick
5. Write the steps to register for MosueEvents.

Experiment No. 3

Aim of the Experiment: 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.

Objective:

To Develop program using GUI framework (AWT and Swing)

Resources: Eclipse IDE 2018, JDK 1.8.0 is required

.Course Outcome Addressed: CO3

Theory:

GUI: GUI stands for **Graphical User Interface**. This is a type of user interface where user interacts with the computer **using graphics**. Graphics include icons, navigation bars, images etc. Mouse can be used while using this interface to interact with the graphics. It is a very **user-friendly** interface and requires no expertise. Eg: Windows has GUI.

Swing in Java is a Graphical User Interface (GUI) toolkit that includes the GUI components. Swing provides a rich set of widgets and packages to make sophisticated GUI components for Java applications. Swing is a part of Java Foundation Classes(JFC), which is an API for Java GUI programing that provide GUI.

The Java Swing library is built on top of the Java Abstract Widget Toolkit (**AWT**), an older, platform dependent GUI toolkit. You can use the Java simple GUI programming components like button, textbox, etc., from the library and do not have to create the components from scratch.

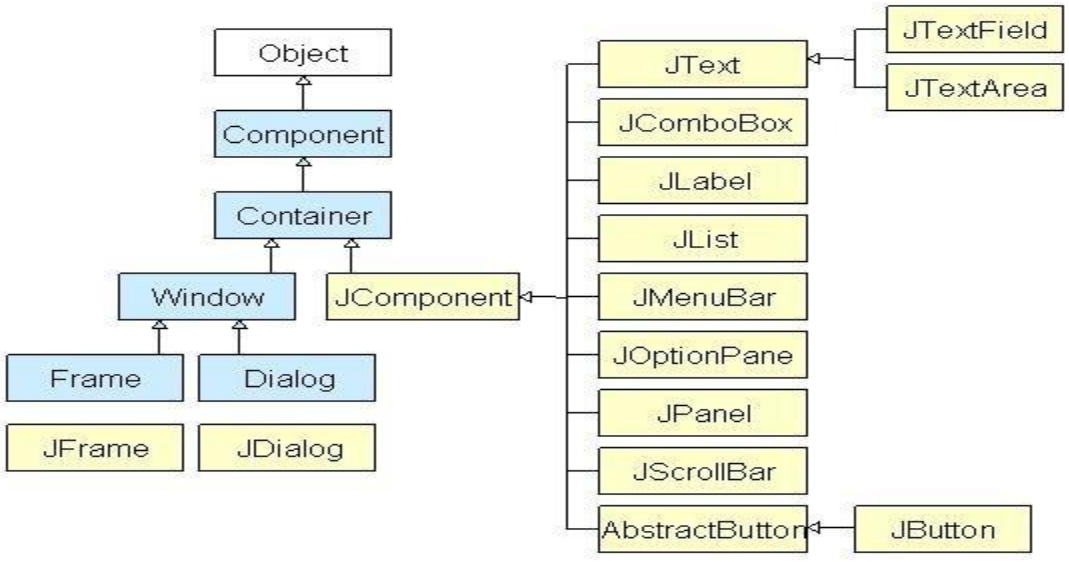


Fig. Java Swing Class Hierarchy Diagram

Container Class:

Container classes are classes that can have other components on it. So for creating a Java Swing GUI, we need at least one container object. There are 3 types of Java Swing containers.

1. Panel: It is a pure container and is not a window in itself. The sole purpose of a Panel is to organize the components on to a window.
2. Frame: It is a fully functioning window with its title and icons.
3. Dialog: It can be thought of like a pop-up window that pops out when a message has to be displayed. It is not a fully functioning window like the Frame.

Following is the list of commonly used controls while designing GUI using SWING.

S.No.	Class & Description
1	JLabel A JLabel object is a component for placing text in a container.
2	JButton This class creates a labeled button.
3	JColorChooser A JColorChooser provides a pane of controls designed to allow a user to manipulate and select a color.
4	JCheck Box A JCheckBox is a graphical component that can be in either an on (true) or off (false) state.
5	JRadioButton The JRadioButton class is a graphical component that can be in either an on (true) or off (false) state. in a group.
6	JList A JList component presents the user with a scrolling list of text items.
7	JComboBox A JComboBox component presents the user with a to show up menu of choices.

8	JTextField A JTextField object is a text component that allows for the editing of a single line of text.
9	JPasswordField A JPasswordField object is a text component specialized for password entry.
10	JTextArea A JTextArea object is a text component that allows editing of a multiple lines of text.
11	ImageIcon A ImageIcon control is an implementation of the Icon interface that paints Icons from Images
12	JScrollbar A Scrollbar control represents a scroll bar component in order to enable the user to select from range of values.
13	JOptionPane JOptionPane provides set of standard dialog boxes that prompt users for a value or informs them of something.
14	JFileChooser A JFileChooser control represents a dialog window from which the user can select a file.
15	JProgressBar As the task progresses towards completion, the progress bar displays the task's percentage of completion.
16	JSlider A JSlider lets the user graphically select a value by sliding a knob within a bounded interval.
17	JSpinner A JSpinner is a single line input field that lets the user select a number or an object value from an ordered sequence.

SOURCE CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;
public class StudentResult {
  // Function to write a student information in JFrame and storing it in a file
  public static void StudentInfo()
  {// Creating a new frame using JFrame
     JFrame f = new JFrame("Student Result Form");


// Creating the labels
     JLabel l1, l2, l3, l4, l5,l6,l7,l8,l9;


// Creating three text fields for student name, college mail ID  and for  Mobile No
     JTextField t1, t2, t3,t4,t5,t6,t7;


 // Creating two JComboboxes for Branch and for Section
     JComboBox j1, j2;


// Creating  two buttons
     JButton b1, b2;


// Naming the labels and setting
     // the bounds for the labels
     l1 = new JLabel("Student Name:");
     l1.setBounds(50, 50, 100, 30);
     l2 = new JLabel("Branch:");
     l2.setBounds(50, 100, 120, 30);
     l3 = new JLabel("Div :");
     l3.setBounds(400, 50, 50, 30);
     l4 = new JLabel("Roll no:");
     l4.setBounds(400, 100, 70, 30);
     l5 = new JLabel(" MC:");
     l5.setBounds(400, 150, 70, 30);
     l6 = new JLabel("EMF:");
     l6.setBounds(50, 150, 70, 30);
     l7 = new JLabel("FJP:");
     l7.setBounds(400, 200, 70, 30);
```

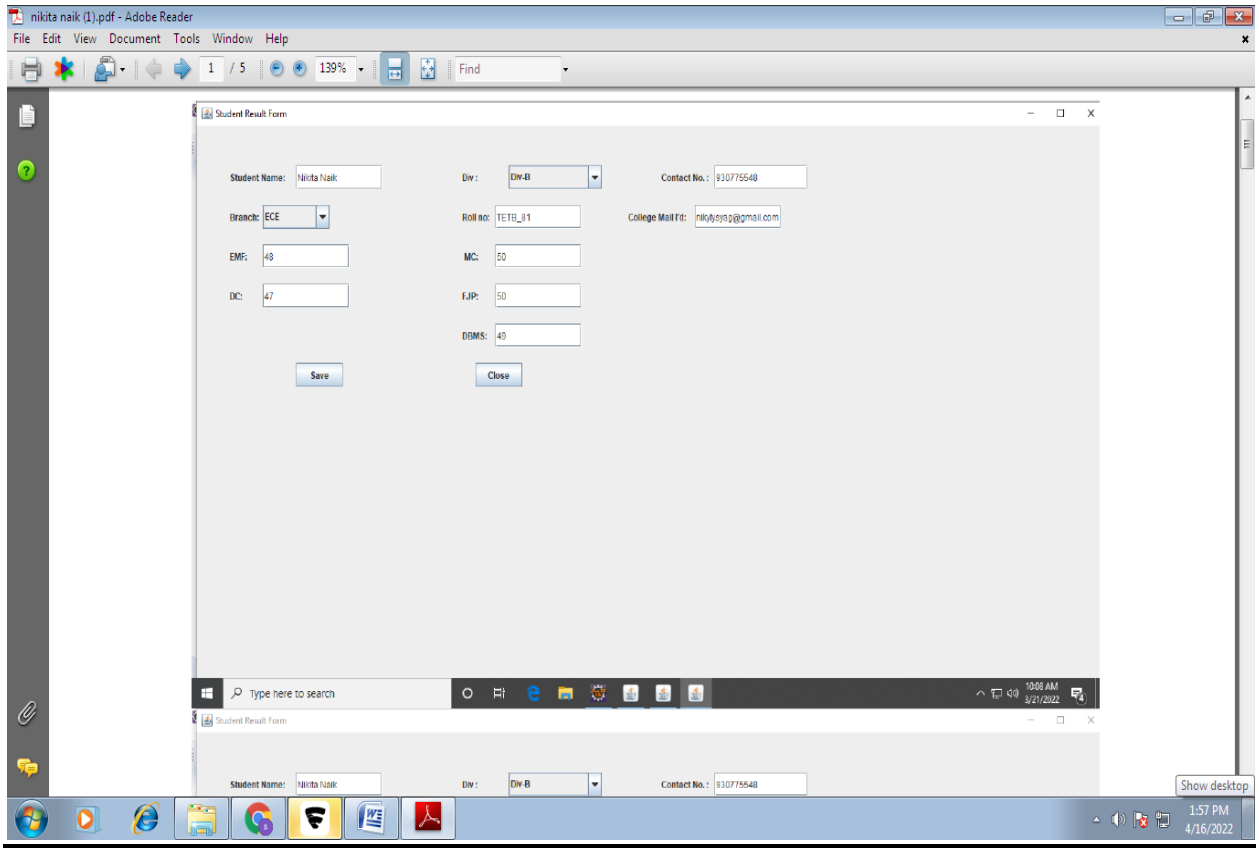
```
l8 = new JLabel("DC:");
l8.setBounds(50, 200, 70, 30);
l9 = new JLabel("DBMS:");
l9.setBounds(400, 250, 70, 30);



// Creating the textfields and
// setting the bounds for textfields
t1 = new JTextField();
t1.setBounds(150, 50, 130, 30);
t2 = new JTextField();
t2.setBounds(450, 100, 130, 30);
t3 = new JTextField();
t3.setBounds(450, 150, 130, 30);
t4 = new JTextField();
t4.setBounds(100, 150, 130, 30);
t5 = new JTextField();
t5.setBounds(450, 200, 130, 30);
t6 = new JTextField();
t6.setBounds(100, 200, 130, 30);
t7 = new JTextField();
t7.setBounds(450, 250, 130, 30);

// Creating two string arrays one for
// braches and other for sections
String s1[ ]
   = { "  ", "CSE", "ECE", "EEE",
      "CIVIL", "MECH", "Others" };
String s2[ ]
   = { "  ", "Div-A", "Div-B", "Div-C"};

// Creating two JComboBoxes for selecting branch and other for selecting the section
// and setting the bounds
j1 = new JComboBox(s1);
j1.setBounds(100, 100, 100, 30);
j2 = new JComboBox(s2);
j2.setBounds(470, 50, 140, 30);

// Creating one button for Saving and other button to close
// and setting the bounds
b1 = new JButton("Save");
b1.setBounds(150, 300, 70, 30);
b2 = new JButton("Close");
```

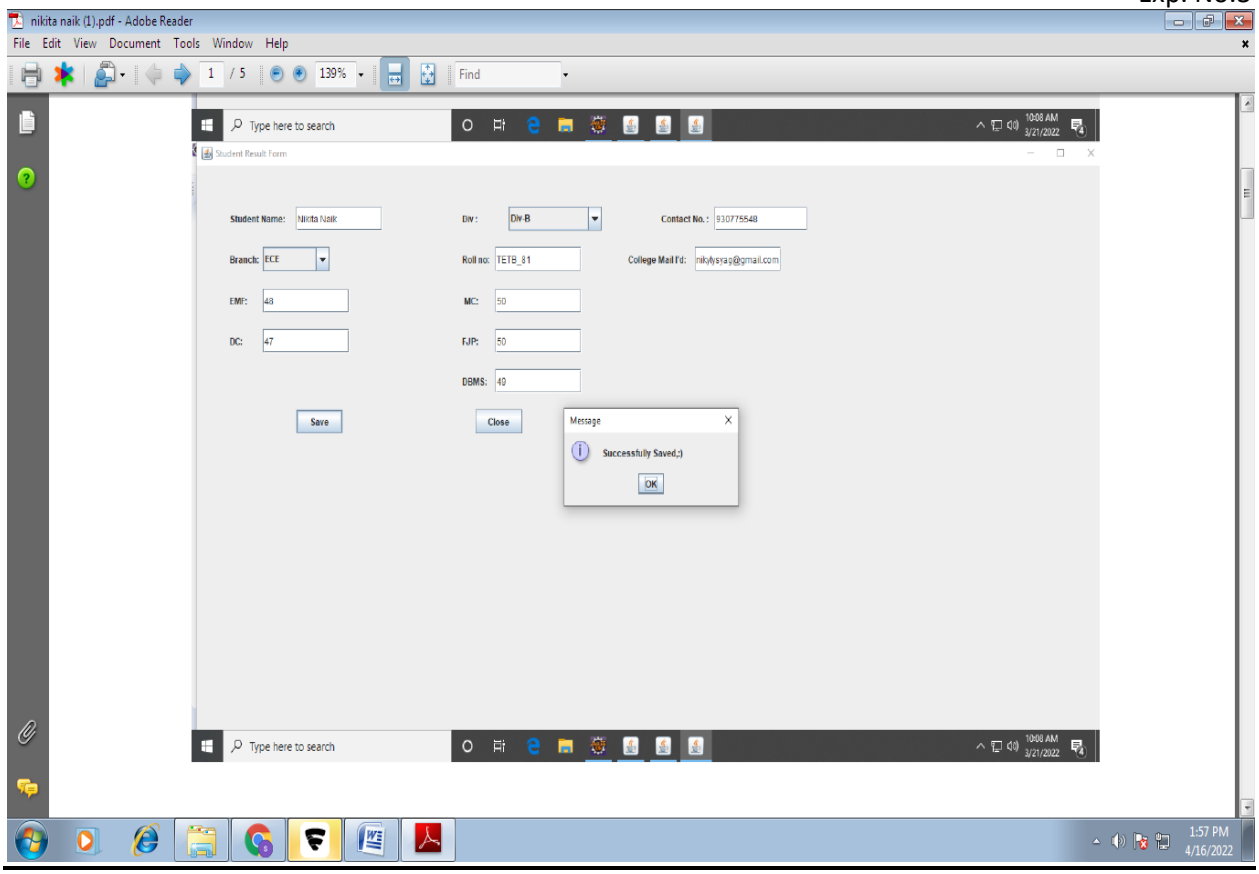
```
b2.setBounds(420, 300, 70, 30);

// Adding action listener
b1.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e)
   {

      // Getting the text from text fields
      // and JComboboxes
      // and copying it to a strings

      String s1 = t1.getText();
      String s2 = t2.getText();
      String s3 = j1.getSelectedItem() + "";
      String s4 = j2.getSelectedItem() + "";
      String s5 = t3.getText();
      String s6 = t4.getText();
      String s7 = t5.getText();
      String s8 = t6.getText();
      String s9 = t7.getText();

      String res="";
      if (e.getSource() == b1) {

            res=res+"Name is: "+ s1 + "  |  ";
            res=res+ "Roll no is:"+s2 + "  |  ";
            res=res+"Branch is:"+ s3 + "  |  ";
            res=res+"Division is:"+ s4 + "  |  ";
            res=res+"Marks of MC is:"+s5 + "  |  ";
            res=res+"Marks of EMF is:"+s6 + "  |  ";
            res=res+"Marks of DC is:"+s7 + "  |  ";
            res=res+"Marks of FJP is:"+s8 + "  |   ";
            res=res+"Marks of DBMS is:"+s9 + "  |  ";


      // Shows a Pop up Message when save button is clicked
      JOptionPane.showMessageDialog(f,"Successfully Saved,:)");
      f.dispose();
      new ResultForm(res);
   }
 }
});

// Action listener to close the form
```

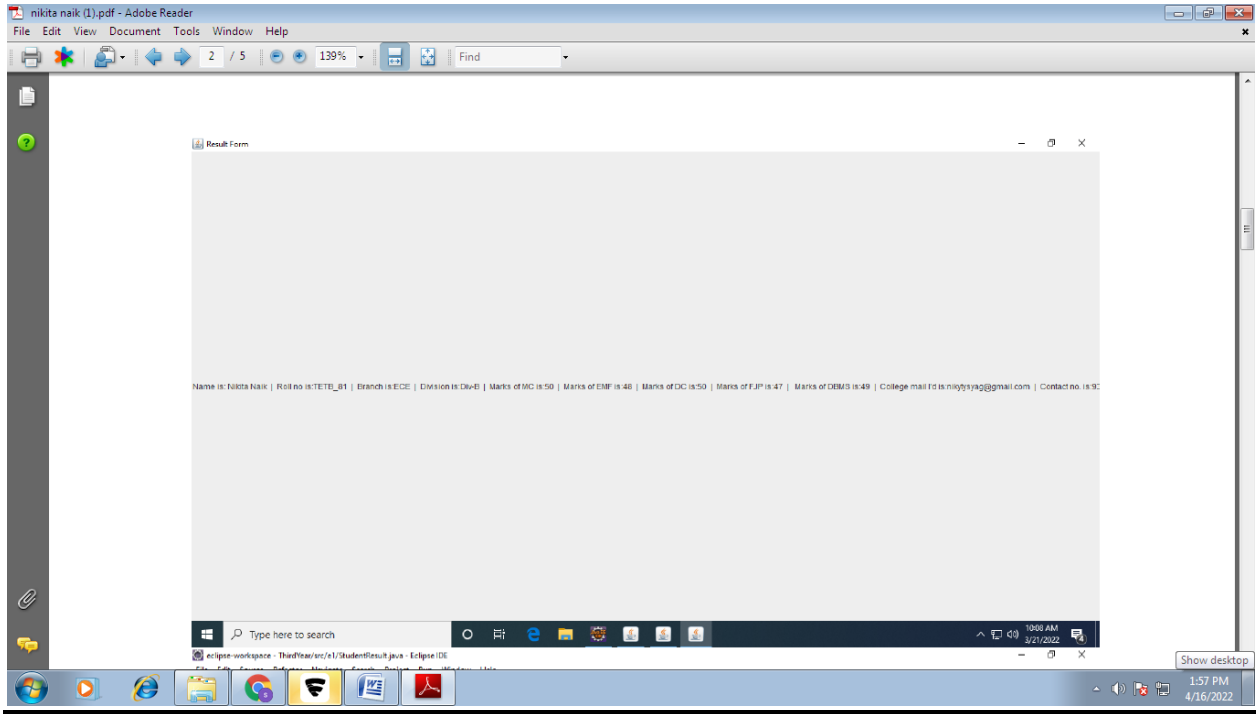
```
    b2.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e)
       {
          f.dispose();
       }
    });

    // Default method for closing the frame
    f.addWindowListener(new WindowAdapter() {
       public void windowClosing(WindowEvent e)
       {
          System.exit(0);
       }
    });

    // Adding the created objects
    // to the frame
    f.add(l1);
    f.add(t1);
    f.add(l2);
    f.add(t2);
    f.add(l3);
    f.add(j1);
    f.add(l4);
    f.add(j2);
    f.add(l5);
    f.add(t3);
    f.add(l6);
    f.add(t4);
    f.add(l7);
    f.add(t5);
    f.add(l8);
    f.add(t6);
    f.add(l9);
    f.add(t7);
    f.add(b1);
    f.add(b2);

    f.setLayout(null);
    f.setSize(3000, 3000);
    f.setVisible(true);
 }
 // Driver code
 public static void main(String args[])
```

```
  {
     StudentInfo();
  }
}

class ResultForm extends JFrame {

public ResultForm(String res){
super("Result Form");
Label l1=new Label(res);
add(l1);
setSize(3000,3000);
setVisible(true);

}

}
```

OUTPUT:




nikita naik (1).pdf - Adobe Reader
File Edit View Document Tools Window Help
Student Result Form
Student Name: Nikita Naik
Div : Div-B
Contact No. : 9307775548
Branch: ECE
Roll no: TETB_81
College Mail Id: nikytysyag@gmail.com
EMF: 48
MC: 50
DC: 47
FJP: 50
DBMS: 49
Save
Close
Message
Successfully Saved.:)
OK


nikita naik (1).pdf - Adobe Reader
File Edit View Document Tools Window Help
Result Form
Name is: Nikita Naik | Roll no is:TETB_81 | Branch is:ECE | Division is:Div-B | Marks of MC is:50 | Marks of EMF is:48 | Marks of DC is:50 | Marks of FJP is:47 | Marks of DBMS is:49 | College mail Id is:nikytysyag@gmail.com | Contact no. is:9

Conclusion:

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. What is Java Swing?
2. What is AWT?
3. What is JFC?
4. What are the Differences Between Swing And AWT?
5. Why Swing components are called lightweight components?

Experiment No. 4

Aim of the Experiment: Write a program to insert and retrieve the data from the database using JDBC.

Objective:

To create database using MySQL to insert and retrieve the data from the database using JDBC.

Resources: MySQL, MySQL connector, Eclipse IDE 2018, JDK 1.8.0 is required

Course Outcome Addressed: CO4

Theory:

JDBC stands for Java Database Connectivity, which is a standard Java API for database-independent connectivity between the Java programming language and a wide range of databases.

The JDBC library includes APIs for each of the tasks mentioned below that are commonly associated with database usage.

- Making a connection to a database.
- Creating SQL or MySQL statements.
- Executing SQL or MySQL queries in the database.
- Viewing & Modifying the resulting records.

Fundamentally, JDBC is a specification that provides a complete set of interfaces that allows for portable access to an underlying database. Java can be used to write different types of executables, such as −

- Java Applications
- Java Applets
- Java Servlets
- Java ServerPages (JSPs)
- Enterprise JavaBeans (EJBs).

All of these different executables are able to use a JDBC driver to access a database, and take advantage of the stored data.

JDBC provides the same capabilities as ODBC, allowing Java programs to contain database-independent code.

Structured Query Language (SQL) is a standardized language that allows you to perform operations on a database, such as creating entries, reading content, updating content, and deleting entries.

SQL is supported by almost any database you will likely use, and it allows you to write database code independently of the underlying database

MySQL is the most popular Open Source Relational SQL database management system.

To establish a connection with MySQL in JDBC, first we have to install MySQL on our system and we have to add the **MySQL connector** (it is a .jar file containing the classes that are the implementation of interfaces provided by Sun Microsystems) to our class path variable.

Download mysql-connector.jar

https://static.javatpoint.com/src/jdbc/mysql-connector.jar

- To add this to eclipse java project,
- Right click on Project and go to properties
- Click on Java Build Path -> Libraries -> Classpath -> Add External JARs
- Provide the path of mysql-connector.jar file and click on Open
- Click Apply and Close

To connect Java application with the MySQL database, we need to follow 5 following steps. We are using MySql as the database; so we need to know following information's for the Mysql database:

- Driver class: The driver class for the mysql database is com.mysql.jdbc.Driver
- Connection URL: The connection URL for the mysql database is jdbc:mysql://localhost:3306/sppu where jdbc is the API, mysql is the database, localhost is the server name on which mysql is running, we may also use IP address, 3306 is the port number and sppu is the database name. We may use any database, in such case, we need to replace the sppu with our database name.
- Username: The default username for the mysql database is root.
- Password: It is the password given by the user at the time of installing the mysql database. In this example, we are going to use root as the password.

VIMP Classes Use in JDBC Practical:

- DriverManager Class: DriverManager class acts as an interface between users and drivers. It keeps track of the drivers that are available and handles establishing a connection between a database and the appropriate driver.
- Connection Interface: A Connection is a session between a Java application and a database. It helps to establish a connection with the database. The Connection interface is a factory of Statement, PreparedStatement.
- 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.
- ResultSet Interface: The object of ResultSet maintains a cursor pointing to a row of a table. Initially, cursor points to before the first row.
- PreparedStatement Interface: The PreparedStatement interface is a subinterface of Statement. It is used to execute parameterized query.
-

SOURCE CODE:

4a. Write a program to insert the data from the database using JDBC

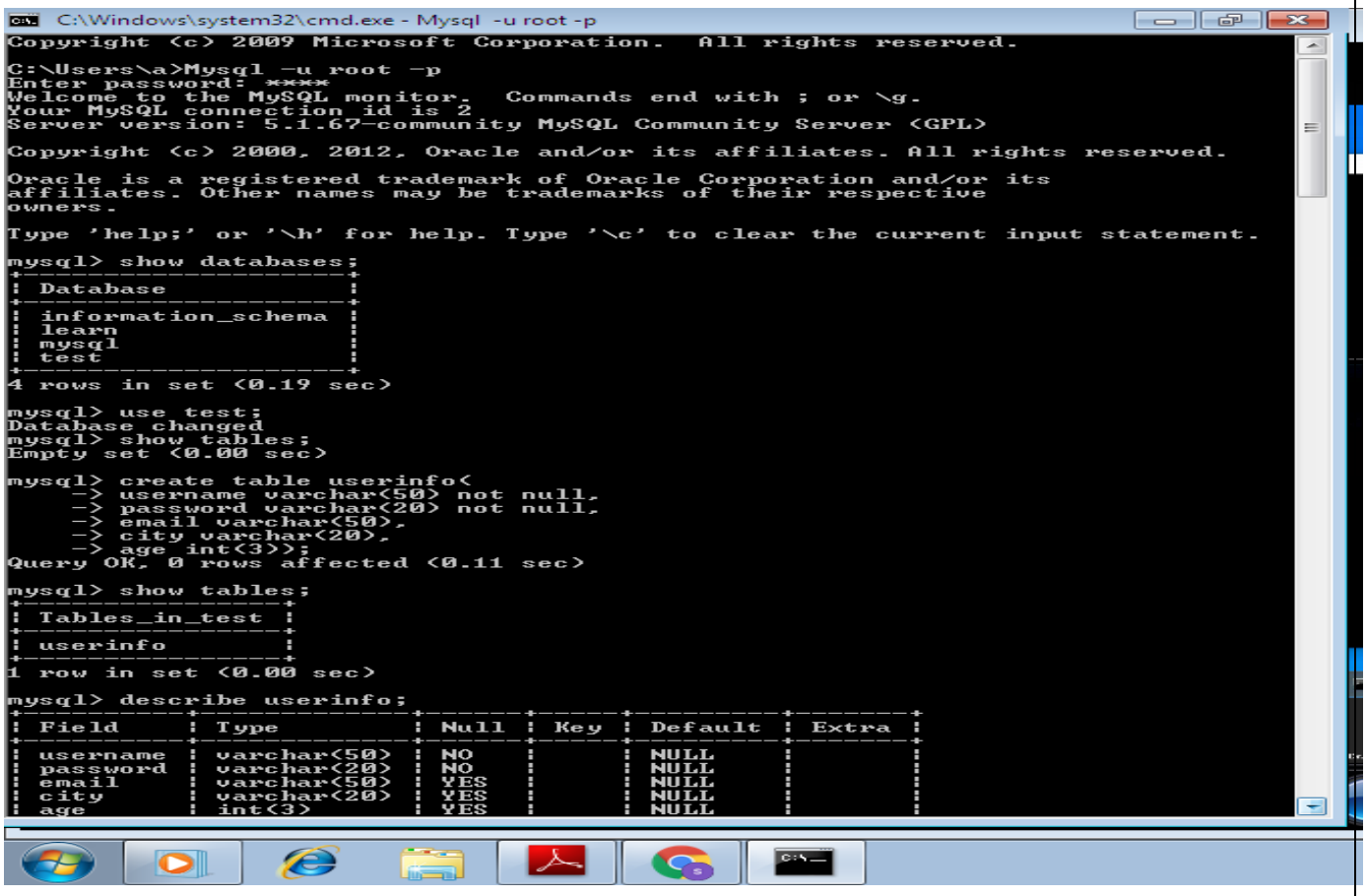
Program:

```
package Mysql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;


public class SQLPreparedStatementInsert {
public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","ro
ot");
```

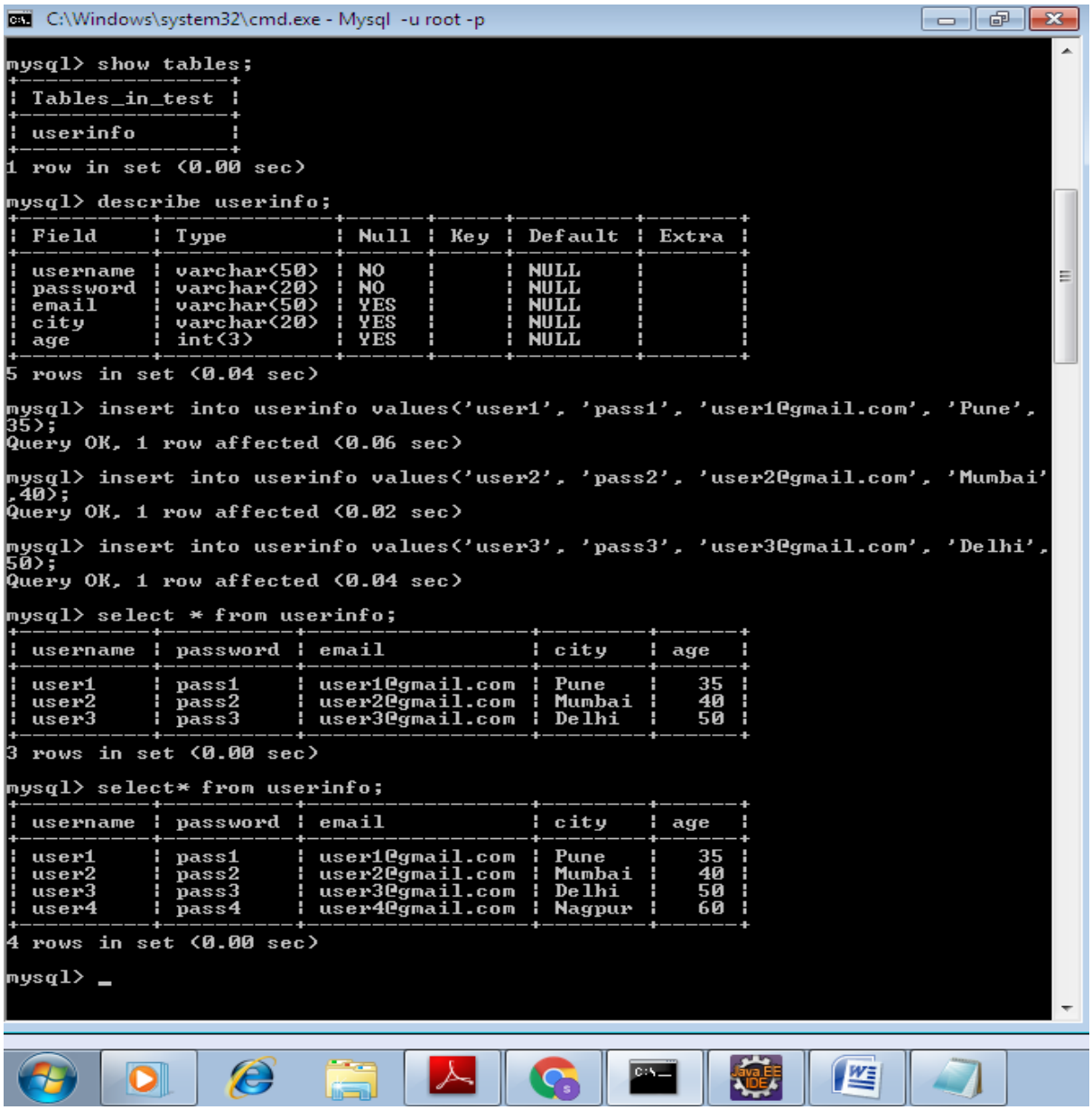
```
PreparedStatement stmt = con.prepareStatement("insert into userinfo
values (?,?,?,?,?)");
stmt.setString(1,"user4");
stmt.setString(2,"pass4");
stmt.setString(3,"user4@gmail.com");
stmt.setString(4, "Nagpur");
stmt.setString(5, "60");

int i = stmt.executeUpdate();
System.out.println(i + "Records inserted..");
con.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```

Output:

Before Insert program-

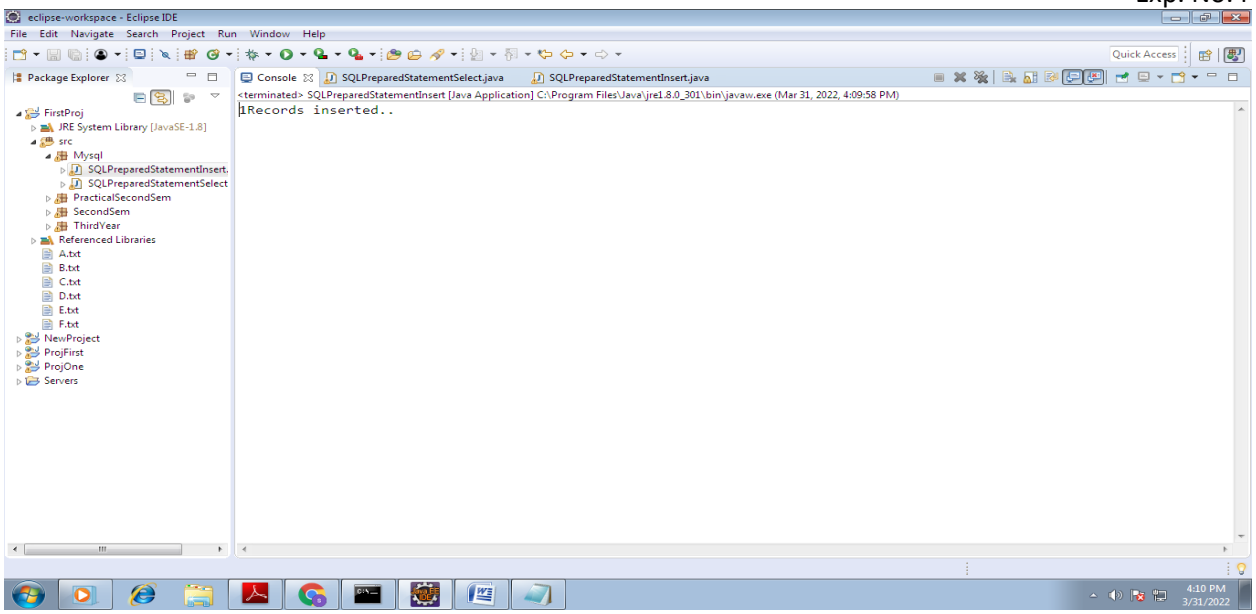


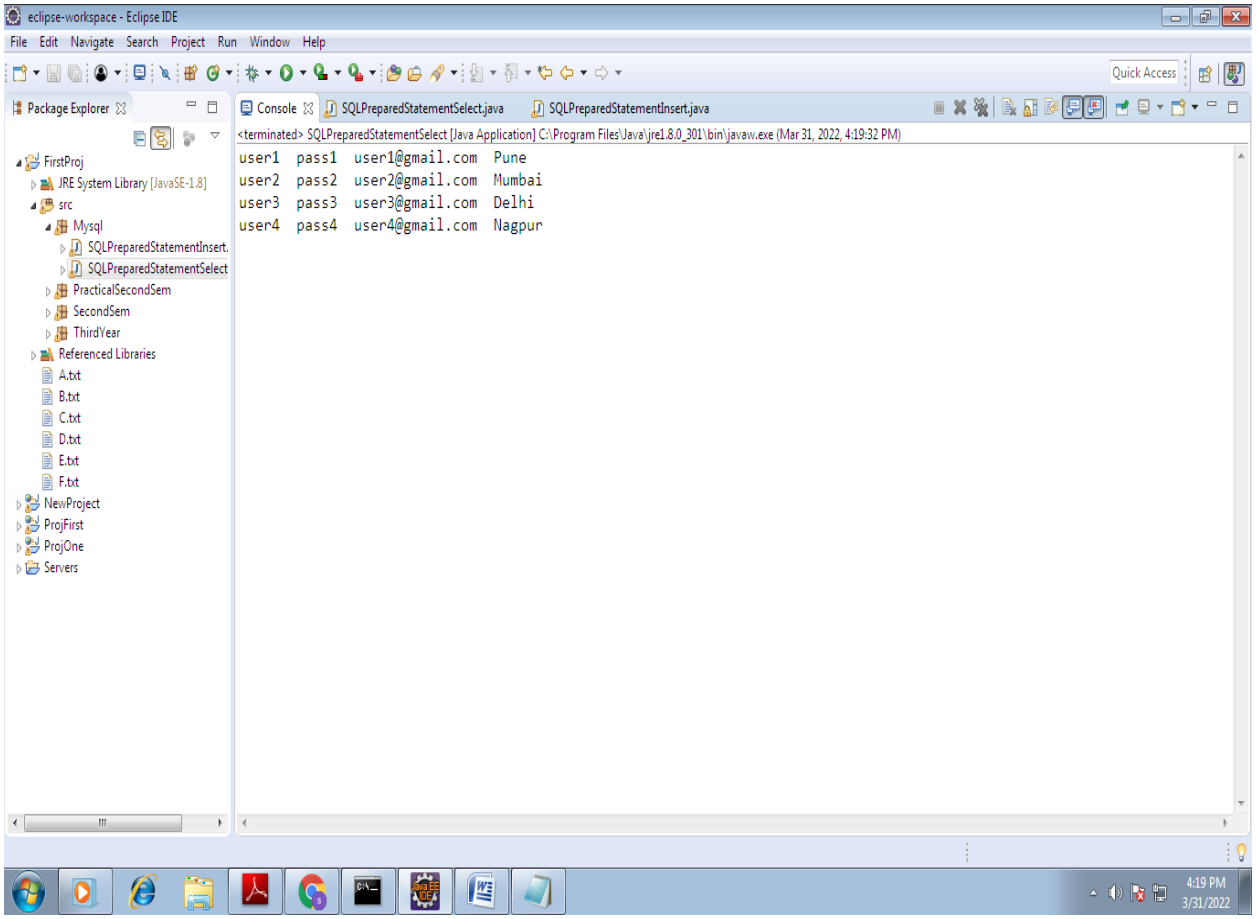
After Insert program-



Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)

Exp. No.4



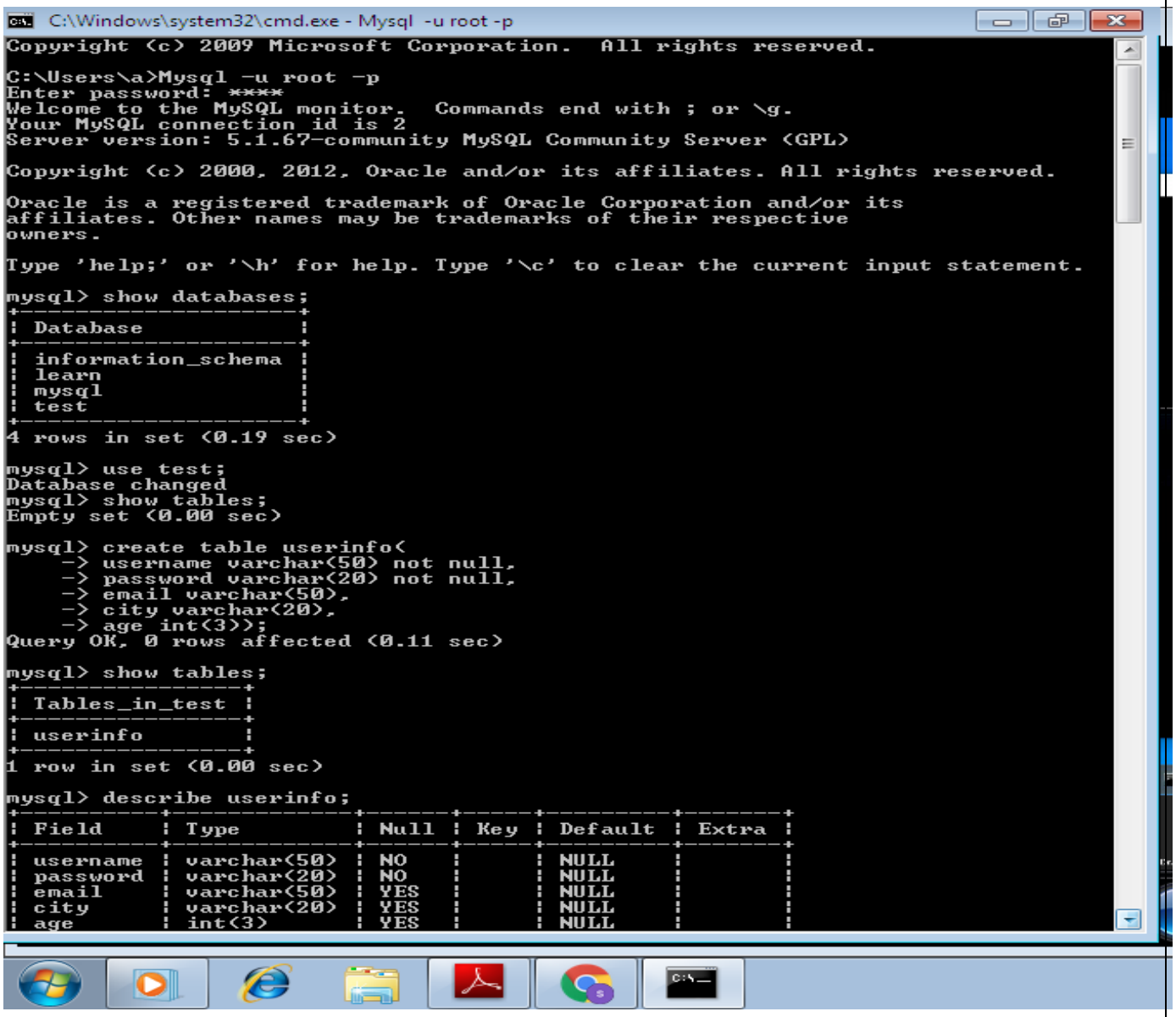


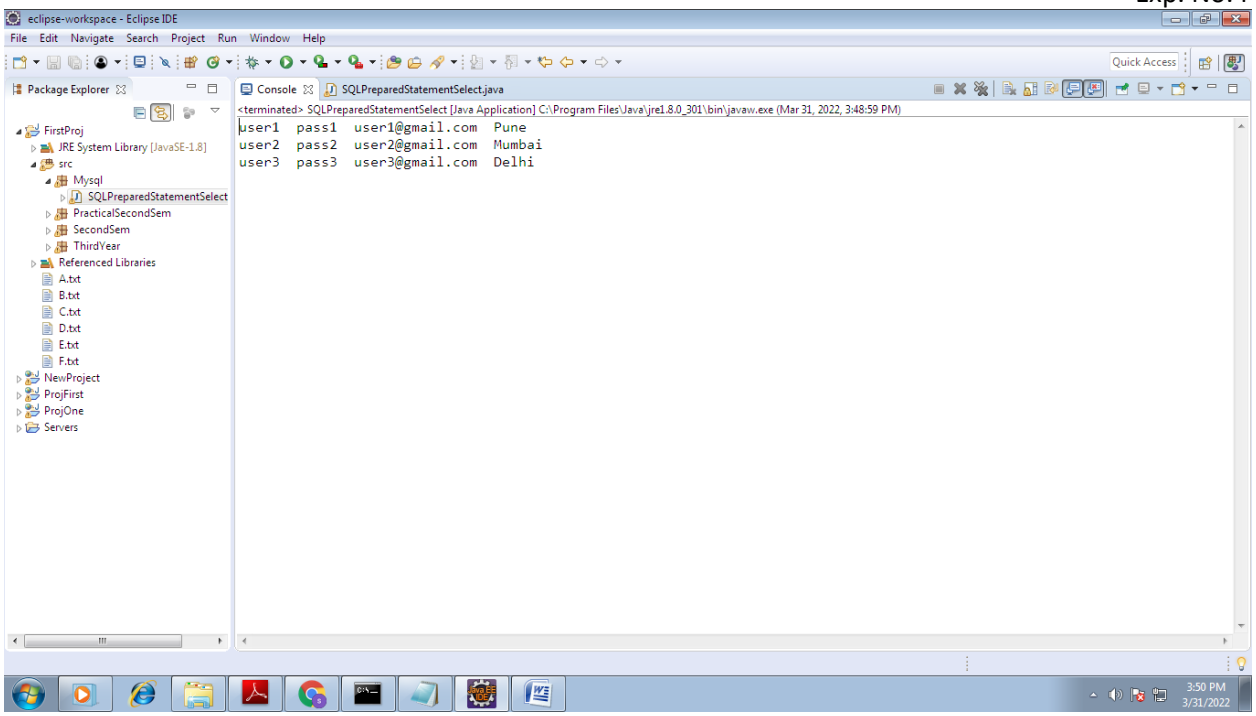
AJP Lab 4b Write a program to retrieve the data from the database using JDBC.

Program:

```
package Mysql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
public class SQLPreparedStatementSelect {

public static void main(String[] args) {
try{

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

Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","ro
ot");


PreparedStatement stmt = con.prepareStatement("select * from userinfo");
ResultSet rs = stmt.executeQuery();
while(rs.next())
{

System.out.println(rs.getString(1) + "   " +
rs.getString(2) + "   " + rs.getString(3) + "   " +
rs.getString(4) + "   ");
}

con.close();
}

catch(Exception e){
System.out.println(e);
}
}
}
```

Output:


C:\Windows\system32\cmd.exe - Mysql -u root -p
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\a>Mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.67-community MySQL Community Server (GPL)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
+--------------------+
| Database |
+--------------------+
| information_schema |
| learn |
| mysql |
| test |
+--------------------+
4 rows in set (0.19 sec)
mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)
mysql> create table userinfo(
-> username varchar(50) not null,
-> password varchar(20) not null,
-> email varchar(50),
-> city varchar(20),
-> age int(3));
Query OK, 0 rows affected (0.11 sec)
mysql> show tables;
+----------------+
| Tables_in_test |
+----------------+
| userinfo |
+----------------+
1 row in set (0.00 sec)
mysql> describe userinfo;
+----------+-------------+------+-----+---------+-------+
| Field | Type | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
username	varchar(50)	NO		NULL	
password	varchar(20)	NO		NULL	
email	varchar(50)	YES		NULL	
city	varchar(20)	YES		NULL	
age	int(3)	YES		NULL	



Conclusion:

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. What is JDBC?
2. What is JDBC Driver?
3. What are the steps to connect to the database in java?
4. What are the JDBC API components?
5. What are the differences between execute, executeQuery, and executeUpdate?

Experiment No. 5

Aim of the Experiment: Develop an RMI application which accepts a string or a number and checks that string or number is palindrome or not.

Objective:

To Develop program using RMI application

Course Outcome Addressed: CO5

Theory:

Palindrome number:

A **palindrome number** is a number that is same after reverse. For example 545, 151, 34543, 343, 171, 48984 are the palindrome numbers. It can also be a string like LOL, MADAM etc.

Remote method invocation(RMI):

Remote method invocation(RMI) allow a java object to invoke method on an object running on another machine. RMI provide remote communication between java program. RMI is used for building distributed application.

Concept of RMI application:

A RMI application can be divided into two part,Client program and Server program. A Server program creates some remote object, make their references available for the client to invoke method on it. A Client program make request for remote objects on server and invoke method on them. Stub and Skeleton are two important object used for communication with remote object.

Stub:

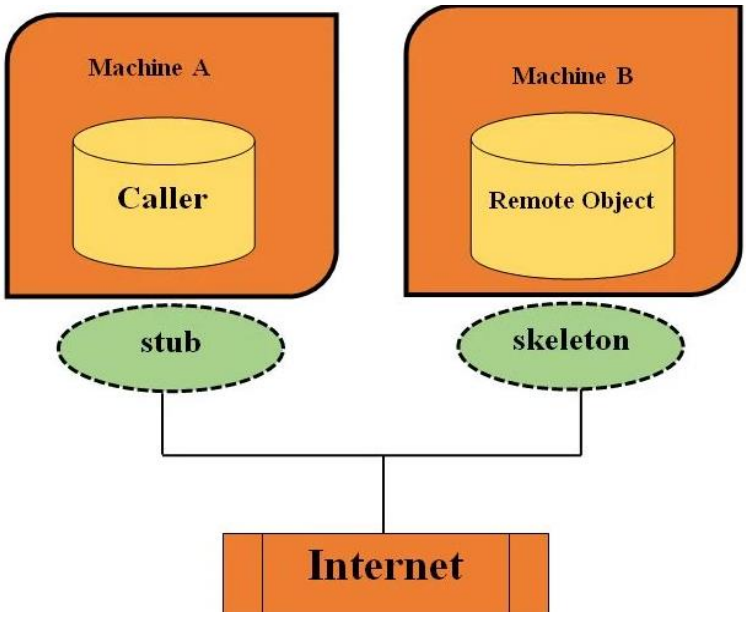
In RMI, a stub is an object that is used as a Gateway for the client-side. All the outgoing request are sent through it. When a client invokes the method on the stub object following things are performed internally:

1. A connection is established using Remote Virtual Machine.
2. It then transmits the parameters to the Remote Virtual Machine. This is also known as Marshals
3. After the 2nd step, it then waits for the output.
4. Now it reads the value or exception which is come as an output.
5. At last, it returns the value to the client.

Skeleton

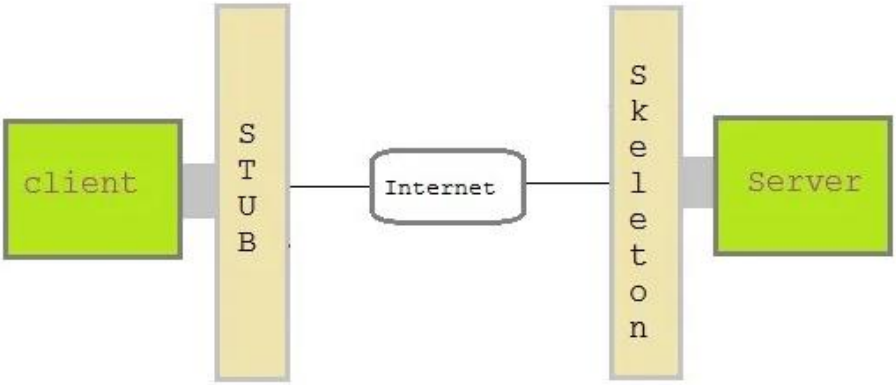
In RMI, a skeleton is an object that is used as a Gateway for the server-side.All the incoming request are sent through it. When a Server invokes the method on the skeleton object following things are performed internally:

1. All the Parameters are read for the remote method.
2. The method is invoked on the remote object.
3. It then writes and transmits the parameters for the result. This is also known as Marshals.



Stub and Skeleton

Stub act as a gateway for Client program. It resides on Client side and communicate with Skeleton object. It establish the connection between remote object and transmit request to it. Skeleton object resides on server program. It is responsible for passing request from Stub to remote object.



Creating a Simple RMI application involves following steps:-

- Define a remote interface.
- Implementing remote interface.
- create and start remote application
- create and start client application

Algorithm:

SERVER SIDE:

Step 1: Start
Step 2: Define the class rmiserver
Step 3: Create the object twox in try
Step 4: Register the object twox
Step 5: Display the exception in catch
Step 6: Stop

CLIENT SIDE:

Step 1: Start
Step 2: Define the class rmiclient
Step 3: Initialize the string s1 in try
Step 4: Create and Initialize the object onex
Step 5: Assign the value to m by calling the method palin
Step 6: Check whether the string is palindrome or not
Step 7: Display whether the string is palindrome or not
Step 8: Display the exception in catch
Step 9: Stop

SOURCE CODE:

one.java

```
import java.rmi.*;
interface one extends Remote
{

   public int palin(String a) throws RemoteException;
}
```

two.java

```
import java.rmi.*;
import java.lang.*;
import java.rmi.server.*;
public class two extends UnicastRemoteObject implements one
{
   public two() throws RemoteException { }
   public int palin(String a) throws RemoteException
   {
      System.out.println("Hello");
      StringBuffer str = new StringBuffer(a);
      String str1 = str.toString();
      System.out.println("Print : " + str1.toString());
      StringBuffer str2 = str.reverse();
      System.out.println("Print : " + str2.toString());
```

```
        int b = str1.compareTo(str2.toString());
        System.out.println("Print : " + b);
        if (b == 0)
            return 1;
        else
            return 0;
    }
}
```

rmiserver.java

```
import java.io.*;
import java.rmi.*;
import java.net.*;
public class rmiserver
{
    public static void main(String args[]) throws Exception
    {
        try
        {
            two twox = new two();
            Naming.bind("palin", twox);
            System.out.println("Object registered");
        }
        catch(Exception e)
        {
            System.out.println("Exception" + e);
        }
    }
}
```

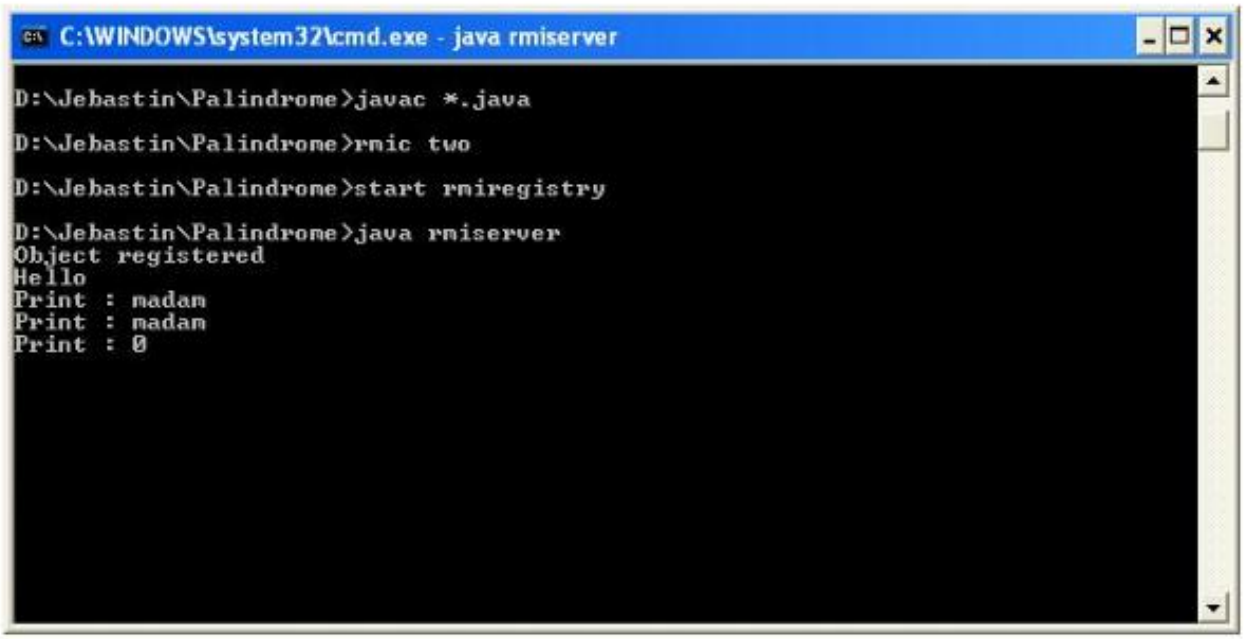
rmiclient.java

```
import java.io.*;
import java.rmi.*;
import java.net.*;
public class rmiclient
{
    public static void main(String args[]) throws Exception
    {
        try
        {
            String s1 = "rmi://localhost/palin";
            one onex = (one)Naming.lookup(s1);
            int m = onex.palin("madam");
            System.out.println("Print : " + m);
            if (m == 1)
            {
                System.out.println("The given string is a Palindrome");
```

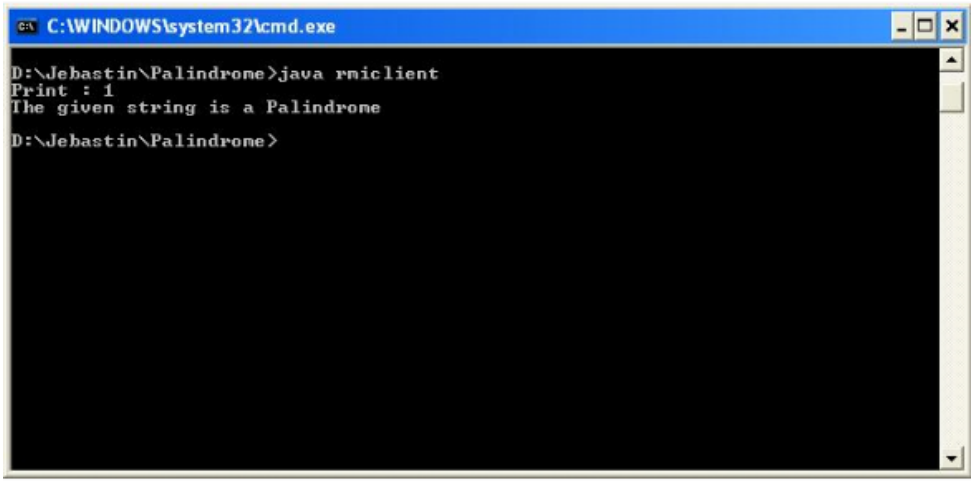
```
        }
        else
        {
           System.out.println("The given string is not a Palindrome");
        }
      }
      catch (Exception e)
      {
        System.out.println("Exception" + e);
      }
   }
}
```

OUTPUT:

SERVER SIDE:



CLIENT SIDE:



Conclusion:

Thus the Java program for the implementation of Remote Method Invocation for palindrome was performed and the output was verified.

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. What is Java Remote Method Invocation (RMI)?
2. What is RMI remote object?
3. How does RMI communicate with the remote object?
4. Explain Different layers of RMI architecture.
5. What Are The Different Types Of Classes That Are Used In RMI?

Experiment No. 6

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

Objective:

To understand use of InetAddress class and its factory methods.

Resources: Eclipse IDE 2018, JDK 1.8.0 is required

Course Outcome Addressed: CO6

Theory:

Java InetAddress class represents an IP address. The java.net.InetAddress class provides methods to get the IP of any host name *for example* www.google.com, www.facebook.com, etc.
An IP address is represented by 32-bit or 128-bit unsigned number. An instance of InetAddress represents the IP address with its corresponding host name.
There are two types of addresses: Unicast and Multicast.
The Unicast is an identifier for a single interface whereas Multicast is an identifier for a set of interfaces.
Moreover, InetAddress has a cache mechanism to store successful and unsuccessful host name resolutions.
The **InetAddress** class is used to encapsulate both the numerical IP address and the domain name for that address. You interact with this class by using the name of an IP host, which is more convenient and understandable than its IP address. The InetAddress class hides the number inside. InetAddress can handle both **IPv4** and **IPv6** addresses.

IP Address

- An IP address helps to identify a specific resource on the network using a numerical representation.
- Most networks combine IP with TCP (Transmission Control Protocol). It builds a virtual bridge among the destination and the source.

There are two versions of IP address:

1. IPv4

IPv4 is the primary Internet protocol. It is the first version of IP deployed for production in the ARAPNET in 1983. It is a widely used IP version to differentiate devices on network using an addressing scheme. A 32-bit addressing scheme is used to store 2^{32} addresses that is more than 4 million addresses.

Features of IPv4:

- It is a connectionless protocol.
- It utilizes less memory and the addresses can be remembered easily with the class based addressing scheme.
- It also offers video conferencing and libraries.

2. IPv6

IPv6 is the latest version of Internet protocol. It aims at fulfilling the need of more internet addresses. It provides solutions for the problems present in IPv4. It provides 128-bit address space that can be used to form a network of 340 undecillion unique IP addresses. IPv6 is also identified with a name IPng (Internet Protocol next generation).

Features of IPv6:

- It has a stateful and stateless both configurations.
- It provides support for quality of service (QoS).
- It has a hierarchical addressing and routing infrastructure.

TCP/IP Protocol

- TCP/IP is a communication protocol model used connect devices over a network via internet.
- TCP/IP helps in the process of addressing, transmitting, routing and receiving the data packets over the internet.
- The two main protocols used in this communication model are:
    1. TCP i.e. Transmission Control Protocol. TCP provides the way to create a communication channel across the network. It also helps in transmission of packets at sender end as well as receiver end.
    2. IP i.e. Internet Protocol. IP provides the address to the nodes connected on the internet. It uses a gateway computer to check whether the IP address is correct and the message is forwarded correctly or not.

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. The InetAddress class has no visible constructors. To create an InetAddress object, you have to use one of the available factory methods. Factory methods are merely a convention whereby static methods in a class return an instance of that class.

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.

Method	Description
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.

Methods of InetAddress class:

Method	Description
public byte[] getAddress()	It returns the raw IP address of this InetAddress object as an array.
public String getHostAddress()	It returns IP address in textual form.
public boolean isAnyLocalAddress()	It returns true if this address represents a local address.
public boolean isLinkLocalAddress()	It returns true if this address is a link local address.
public boolean isLoopbackAddress()	It returns true if this address is a loopback address.
public boolean isMCGlobal()	It used to check whether the multicast address has a global scope or not
public boolean isMCLinkLocal()	It is used to check utility routine if the multicast address has a link scope or not.
public boolean isMCNodeLocal()	It is used to check the utility routine if the multicast address has node scope or not.
public boolean isMCOrgLocal()	It is used to check utility routine if the multicast address has

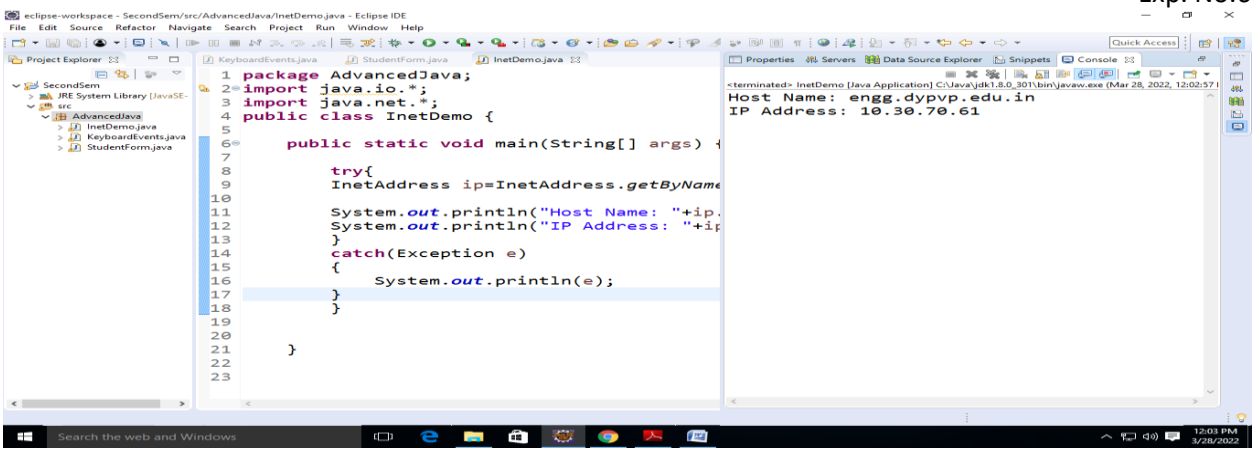
	organization scope or not.
public boolean isMCSiteLocal()	It returns true if this multicast address has site scope.
public boolean isMulticastAddress()	It returns true if this address is an IP multicast address.
public boolean isSiteLocalAddress()	It returns true if this address is a site local address.
public int hashCode()	It returns the hashcode associated with this address object.
public boolean equals(Object obj)	It returns true if this IP address is the same as that of the object specified.

SOURCE CODE:

Programme 1:

```
package AdvancedJava;
import java.io.*;
import java.net.*;
public class InetDemo {

public static void main(String[] args) {

try{
InetAddress ip=InetAddress.getByName("engg.dypvp.edu.in");

System.out.println("Host Name: "+ip.getHostName());
System.out.println("IP Address:"+ip.getHostAddress());
}
catch(Exception e)
            {
                  System.out.println(e);
            }
}
}
```

OUTPUT:

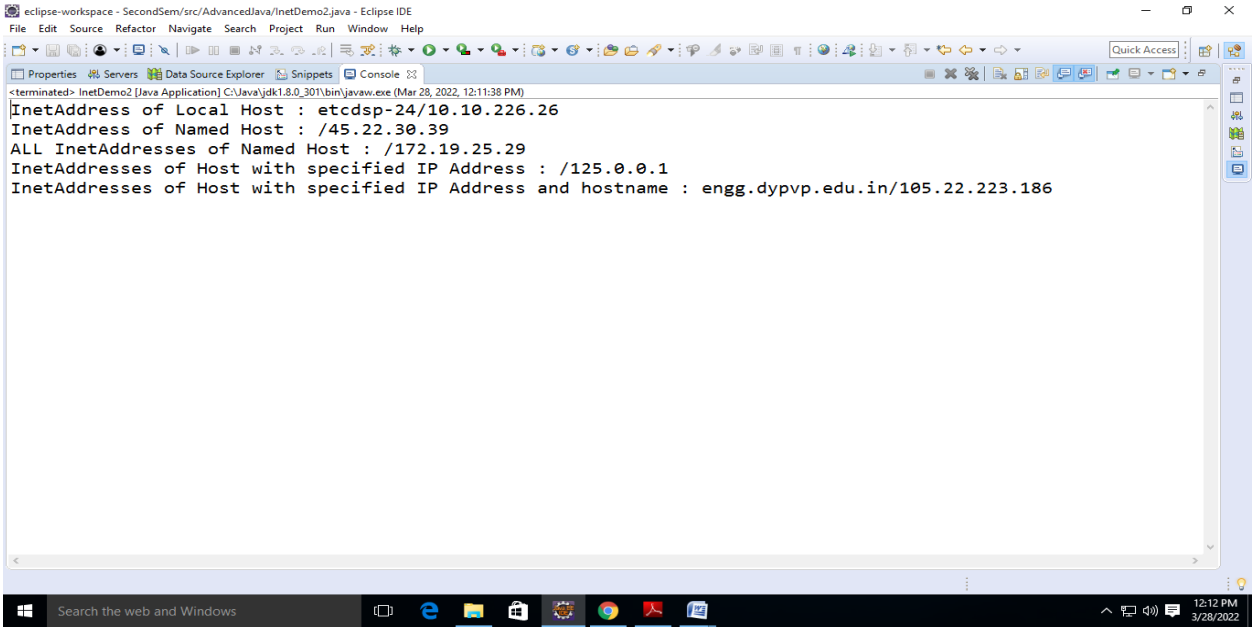


Programme 2:

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

public class InetDemo2 {

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( "engg.dypvp.edu.in", IPAddress2);
System.out.println("InetAddresses of Host with specified IP Address and hostname : "  + address5);
}
}
```

OUTPUT:



Conclusion:

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. What are the purposes of using InetAddress class?
2. Which are the factory method of InetAddress class?
3. How are instances of InetAddress class created?
4. What happens if IP address of host Cannot be determined?
5. Which is the proper method to retrieve the host name of local machine?

Experiment No. 7

Aim of the Experiment:

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

Objective:

To understand client and server side servlet.

Course Outcome Addressed: CO6

Theory:

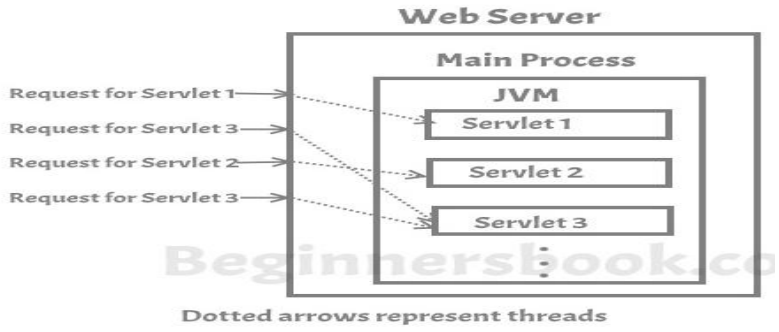
Servlet is a java program that runs inside JVM on the web server. It is used for developing dynamic web applications. The main **difference between static and dynamic web page** is that static page as name suggests remains same for all users however a dynamic web page changes based on the request from client (user's browser). To create servlets we need Servlet API. There are two packages that you must remember while using API, the javax.servlet package that contains the classes to support generic servlet (protocol-independent servlet) and the javax.servlet.http package that contains classes to support http servlet.

hierarchy of packages:

```
java.lang.Object
        |_extended byjavax.servlet.GenericServlet
        |_extended byjavax.servlet.http.HttpServlet
```

Every Servlet must implement the java.servlet.Servlet interface, you can do it by extending one of the following two classes: javax.servlet.GenericServlet or javax.servlet.http.HttpServlet. The first one is for protocol independent Servlet and the second one for http Servlet.

How servlet works?

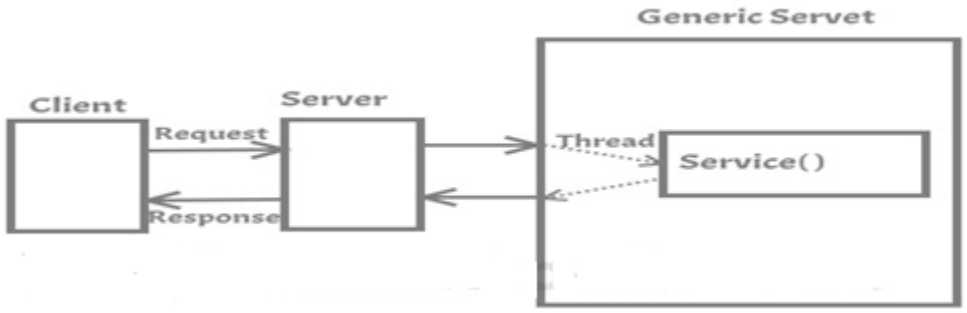


Generic Servlet

To create a Generic Servlet we must extend `javax.servlet.GenericServlet` class. GenericServlet class has an abstract service() method. Which means the subclass of GenericServlet should always override the service() method.
Signature of service() method:

```
public abstract void service(ServletRequest request, ServletResponse response)
        throws ServletException, java.io.IOException
```

The service() method accepts two arguments ServletRequest object and ServletResponse object. The request object tells the servlet about the request made by client while the response object is used to return a response back to the client.



HTTP Servlet

To create Http Servlet we must extend javax.servlet.http.HttpServlet class, which is an abstract class. Unlike Generic Servlet, the HTTP Servlet doesn't override the service() method. Instead it overrides one or more of the following methods. It must override at least one method from the list below:

- **doGet()** – This method is called by servlet service method to handle the HTTP GET request from client. The Get method is used for getting information from the server
- **doPost()** – Used for posting information to the Server
- **doPut()** – This method is similar to doPost method but unlike doPost method where we send information to the server, this method sends file to the server, this is similar to the FTP operation from client to server
- **doDelete()** – allows a client to delete a document, webpage or information from the server
- **init() and destroy()** – Used for managing resources that are held for the life of the servlet
- **getServletInfo()** – Returns information about the servlet, such as author, version, and copyright.

In Http Servlet there is no need to override the service() method as this method dispatches the Http Requests to the correct method handler, for example if it receives HTTP GET Request it dispatches the request to the doGet() method.

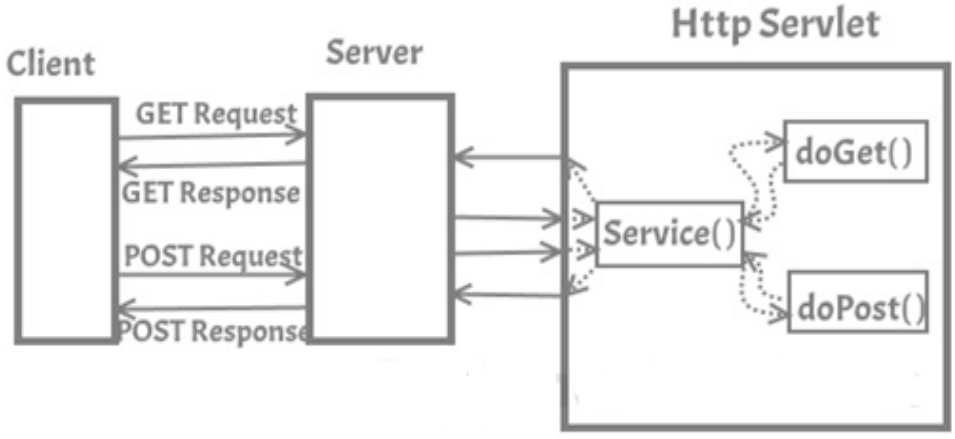
How Servlet Works?

1) When the web server (e.g. Apache Tomcat) starts up, the servlet container deploy and loads all the servlets. During this step Servlet container creates ServletContext object. **ServletContext is an interface that defines the set of methods that a servlet can use to communicate with the servlet container**.

Note: **There is only one ServletContext per webapp** which is common to all the servlets. ServletContext has several useful methods such as addListener(), addFilter() etc. For now I am not explaining them as I will cover them in a separate text about ServletContext.

2) Once the servlet is loaded, the servlet container creates the instance of servlet class. For each instantiated servlet, its init() method is invoked.

3) Client (user browser) sends an Http request to web server on a certain port. Each time the web server receives a request, the servlet container creates HttpServletRequest and HttpServletResponse objects. The HttpServletRequest object provides the access to the request information and the HttpServletResponse object allows us to format and change the http response before sending it to the client.
The servlet container spawns a new thread that calls service() method for each client request. **The service() method dispatches the request to the correct handler method based on the type of request**.
For example if server receives 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.



4) When servlet container shuts down, it unloads all the servlets and calls destroy() method for each initialized servlets.

Program:

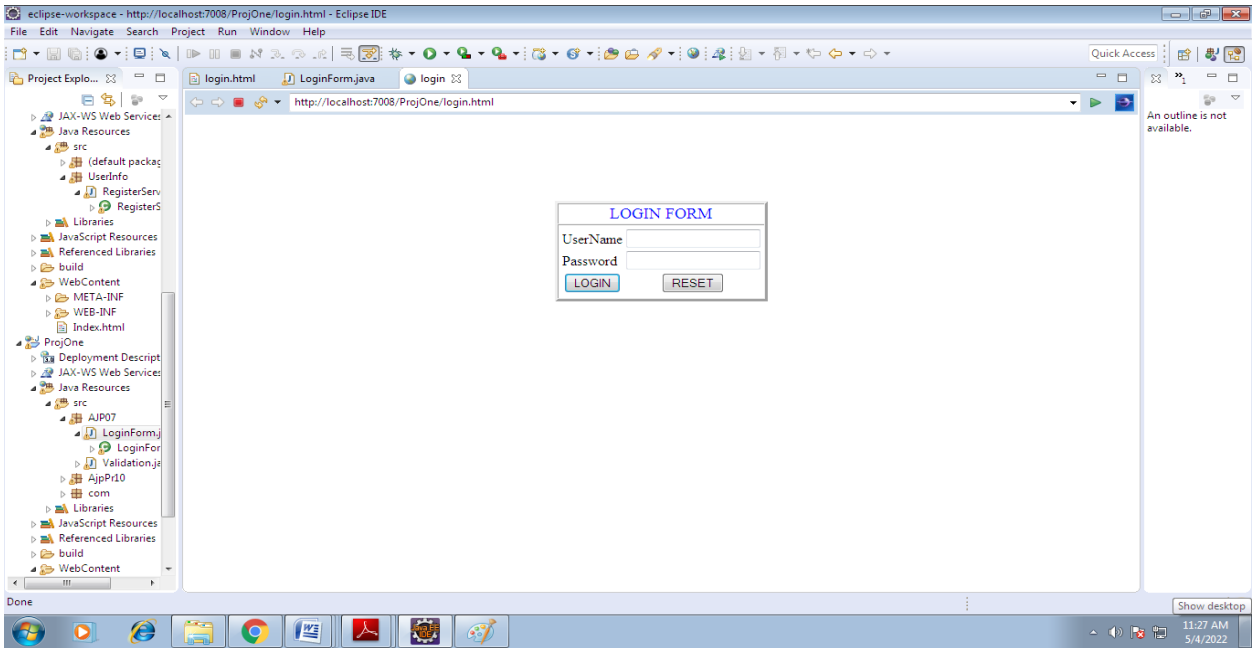
Login.html:

```
<html>
<head><title>login</title></head>
<body>
<form name="login form" method="post"
action="http://localhost:8080/examples/servlet/Validation">
<br/><br/><br/><br/><br/>
<table align="center" border="3" border color="blue" cellspacing="0"height="120">
<tr><td align="center"><font color="blue" size="4">LOGIN FORM</font></td></tr>
<tr><td><table><tr><td>UserName</td><td><input type="text" name="user"/></td></tr>
<tr><td>Password</td><td><input type="password" name="pwd"/></td></tr>
<tr><td align="center"><input type="submit" value="LOGIN"/></td><td
align="center"><input type="Reset" value="RESET"/></td></tr>
</table></td></tr></table></form></body>
</html>
```

Validation.java:

```
import java.io.*;
import java.util.*;
import javax.servlet.*;
public class Validation extends GenericServlet
{
public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
{
PrintWriter pw=res.getWriter();
String x=req.getParameter("user");
String y=req.getParameter("pwd");
if(x.equals("admin")&&y.equals("admin"))
pw.println("<font color='green' size='5'>Welcome to this webpage</font>");
else
pw.println("<font color='red' size='5'>Invalid username or password</font>");
pw.close();
}
}
```

Output:



Conclusion:

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. Explain the Servlet API.
2. How can a servlet get the name of the server and the port number for a particular request?
3. How can a servlet get information about the client machine?
4. What are the three methods of inter-servlet communication?

Experiment No. 8

Aim of the Experiment: Write a database application that uses any JDBC driver.

Objective:

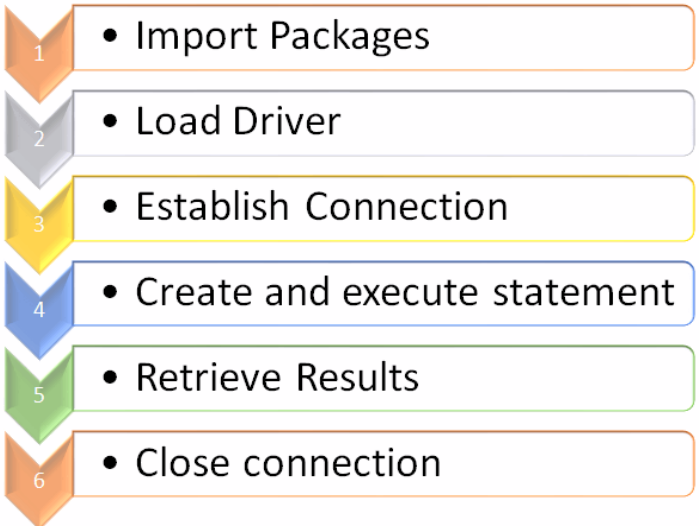
Create a database application using JDBC driver.

Course Outcome Addressed: CO4

Theory:

JDBC Connection Steps

There are 6 basic steps to connect with JDBC. They are enlisted in the below image:



1) Import Packages

First, we need to import the existing packages to use it in our Java program. Import will make sure that JDBC API classes are available for the program. We can then use the classes and subclasses of the packages.

Irrespective of the JDBC Driver, add the following import statement in the Java program.

import java.sql.*;

Import the other classes based on the functionality which you will use in the program. Download the appropriate Jar files for the database which you will use in the program.

.

JDBC API 4.0 mainly provides 2 important packages:

- java.sql
- javax.sql

(i) java.sql package

This package provides classes and interfaces to perform most of the JDBC functions like creating and executing SQL queries.

Classes/ Interfaces	Description
BLOB	It represents SQL Blob value in Java program
CallableStatement	It is used to execute SQL stored procedures
CLOB	It represents SQL Clob value in Java program
Connection	It creates a connection (session) with a specific Database
Date	It provides support for Date SQL type
Driver	It creates an instance of a Driver with Driver Manager
DriverManager	It provides basic service to manage a set of JDBC Drivers
ParameterMetaData	It is an object which can be used to get the information about the types and properties of each parameter in a PreparedStatement Object
PreparedStatement	It is used to create and execute a parameterized query in the Java program
ResultSet	It is used to access the result row-by-row
ResultSetMetaData	It is used to get the information about the types and properties of the columns in a ResultSet object
RowId	It represents the SQL ROWID value
Savepoint	It represents savepoint in transaction
SQLData	It is used to map the SQL User Defined Type (UDT) to a class in Java program
SQLXML	It represents SQL XML type
Statement	It is used to execute a static SQL statement
DriverPropertyInfo	It provides Driver properties to make a connection
SQLException	It provides information on database errors
SQLTimeoutException	It is a subclass of SQLException thrown when the timeout specified by the statement has expired

Classes/ Interfaces	Description
SQLWarning	It is an exception that provides information on database access warnings
Struct	It is a standard mapping in Java program for SQL structured type

(ii) javax.sql package
It is a JDBC extension API and provides server-side data access and processing in Java Program.

Classes/ Interfaces	**Description**
CommonDataSource	It is an interface that defines the methods which are common between DataSource, XADataSource and ConnectionPoolDataSource
ConnectionPoolDataSource	It is a factory for PooledConnection objects
DataSource	It is a factory for connections to the physical DataSource that the object represents
PooledConnection	It is used to manage Connection Pool
RowSet	It provides support to the JDBC API for Java beans Component Model
RowSetMetadata	It has the information about the columns in a RowSet object
ConnectionEvent	It provides information about the occurrence of connection-related events
ConnectionEventListener	It is used to register PooledConnection object events
RowSetEvent	It generates when an event occurs to a Rowset object
StatementEvent	It is sent to all StatementEventListeners which were registered with a PooledConnection generated

2) Load Driver

First, we should load/register the driver in the program before connecting to the Database. You need to register it only once per database in the program.

We can load the driver in the following 2 ways:

1. **Class.forName()**
2. **DriverManager.registerDriver()**

(i) Class.forName()
In this way, the driver's class file loads into the memory at runtime. It implicitly loads the driver. While loading, the driver will register with JDBC automatically.

DB Name	JDBC Driver Name
MySQL	com.mysql.jdbc.Driver
Oracle	oracle.jdbc.driver.OracleDriver
Microsoft SQL Server	com.microsoft.sqlserver.jdbc.SQLServerDriver
MS Access	net.ucanaccess.jdbc.UcanaccessDriver
PostgreSQL	org.postgresql.Driver
IBM DB2	com.ibm.db2.jdbc.net.DB2Driver
Sybase	com.sybase.jdbcSybDriver
TeraData	com.teradata.jdbc.TeraDriver

Note: forName() method is valid only for JDK Compliant Virtual Machines.

(ii) DriverManager.registerDriver()
DriverManager is an inbuilt class that is available in the java.sql package. It acts as a mediator between Java application and database which you want to connect. Before you connect with the database, you need to register the driver with DriverManager. The main function of DriverManager is to load the driver class of the Database and create a connection with DB.

Public static void registerDriver(driver) – This method will register the driver with the Driver Manager. If the driver is already registered, then it won't take any action.

- It will throw **SQLException** if the database error occurs.
- It will throw **NullPointerException** if the driver is null.

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver())

DriverManager.registerDriver(new com.microsoft.sqlserver.jdbc.SQLServerDriver())

Like this, you can register the driver for your Database by passing it as a parameter.

3) Establish Connection

After loading the driver, the next step is to create and establish the connection. Once required, packages are imported and drivers are loaded and registered, then we can go for establishing a Database connection.

DriverManager class has the getConnection method, we will use this method to get the connection with Database. To call getConnection() method, we need to pass 3 parameters. The 3 parameters are string data type URL, a username, and a password to access the database.

The getConnection() method is an overloaded method. The 2 methods are:

- **getConnection(URL,username,password);** – It has 3 parameters URL, username, password.
- **getConnection(URL);** – It has only one parameter. URL has a username and password also.

The following table lists the JDBC connection strings for the different databases:

Database	Connection String/DB URL
MySQL	jdbc:mysql://HOST_NAME:PORT/DATABASE_NAME
Oracle	jdbc:oracle:thin:@HOST_NAME:PORT:SERVICE_NAME
Microsoft SQL Server	jdbc:sqlserver://HOST_NAME:PORT;DatabaseName=< DATABASE_NAME>
MS Access	jdbc:ucanaccess://DATABASE_PATH
PostgreSQL	jdbc:postgresql://HOST_NAME:PORT/DATABASE_NAME
IBM DB2	jdbc:db2://HOSTNAME:PORT/DATABASE_NAME
Sybase	jdbc:Sybase:Tds:HOSTNAME:PORT/DATABASE_NAME
TeraData	jdbc:teradata://HOSTNAME/database=< DATABASE_NAME>,tmode=ANSI,charset=UTF8

Example:

Connection con = DriverManager.getConnection(jdbc:oracle:thin:@localhost:1521:xe,System,Pass123@)

Here in this example,

- **thin** refers to the Driver type.
- **localhost** is where the Oracle database is running.
- **1521** is the port number to connect to DB.
- **xe** – SID
- **System** – User name to connect to the Oracle Database.
- **Pass123@** – Password

4) Create And Execute Statement

Once the connection has established, we can interact with the connected Database. First, we need to create the statement to perform the SQL query and then execute the statement.

(i) Create Statement

Now we will create the statement object that runs the query with the connected database. We use the createStatement method of the *Connection* class to create the query.

There are 3 statement interfaces are available in the java.sql package. These are explained below:

a) Statement

This interface is used to implement simple SQL statements with no parameter. It returns the ResultSet object.

```
Statement statemnt1 = conn.createStatement();
```

b) PreparedStatement

This PreparedStatement interface extends the Statement interface. So, it has more features than the Statement interface. It is used to implement parameterized and precompiled SQL statements. The performance of the application increases because it compiles the query only once.

It is easy to reuse this interface with a new parameter. It supports the IN parameter. Even we can use this statement without any parameter.

```
String select_query = “Select * from states where state_id = 1”;

PreparedStatement prpstmt = conn.prepareStatement(select_query);
```

c) CallableStatement

CallableStatement interface extends the PreparedStatement interface. So, it has more features than the PreparedStatement interface. It is used to implement a parameterized SQL statement that invokes procedure or function in the database. A stored procedure works like a method or function in a class. It supports the IN and OUT parameters.

The CallableStatement instance is created by calling the prepareCall method of the Connection object.

```
CallableStatementcallStmt = con.prepareCall("{call procedures(?,?)}");
```

(ii) Execute The Query

There are 4 important methods to execute the query in Statement interface. These are explained below:

- ResultSet executeQuery(String sql)
- int executeUpdate(String sql)
- boolean execute(String sql)
- int []executeBatch()

a) ResultSet executeQuery(String sql)

The executeQuery() method in Statement interface is used to execute the SQL query and retrieve the values from DB. It returns the ResultSet object. Normally, we will use this method for the SELECT query.

b) executeUpdate(String sql)
The executeUpdate() method is used to execute value specified queries like INSERT, UPDATE, DELETE (DML statements), or DDL statements that return nothing. Mostly, we will use this method for inserting and updating.

c) execute(String sql)
The execute() method is used to execute the SQL query. It returns **true** if it executes the SELECT query. And, it returns **false** if it executes INSERT or UPDATE query.
d) executeBatch()
This method is used to execute a batch of SQL queries to the Database and if all the queries get executed successfully, it returns an array of update counts. We will use this method to insert/update the bulk of records.

5) Retrieve Results

When we execute the queries using the executeQuery() method, the result will be stored in the ResultSet object. The returned ResultSet object will never be null even if there is no matching record in the table. ResultSet object is used to access the data retrieved from the Database.

ResultSet rs 1= statemnt1.executeQuery(QUERY));

We can use the executeQuery() method for the SELECT query. When someone tries to execute the insert/update query, it will throw SQLExecption with the message "**executeQuery method can not be used for update**".
A ResultSet object points to the current row in the Resultset. To iterate the data in the ResultSet object, call the next() method in a while loop. If there is no more record to read, it will return FALSE.

ResultSet can also be used to update data in DB. We can get the data from ResultSet using getter methods such as getInt(), getString(), getDate(). We need to pass the column index or column name as the parameter to get the values using Getter methods.

We will get to know more about the ResultSet in the next tutorial.

6) Close Connection

Finally, we are done with manipulating data in DB. Now we can close the JDBC connection. We need to make sure that we have closed the resource after we have used it. If we don't close them properly we may end up out of connections.

When we close the connection object, Statement and ResultSet objects will be closed automatically.

conn.close();

From Java 7 onwards, we can close the JDBC connections automatically using a try-catch block. JDBC connection should be opened in the parenthesis of the try block. Inside the try block, you can do the database connections normally as we do.

Once the execution exits the try block, it will automatically close the connection. In this case, we don't need to close the connection by calling conn.close method in the Java program.

```
try(Connection conn = DriverManager.getConnection(url, user, password))

{

   //database connection and operation

}
```

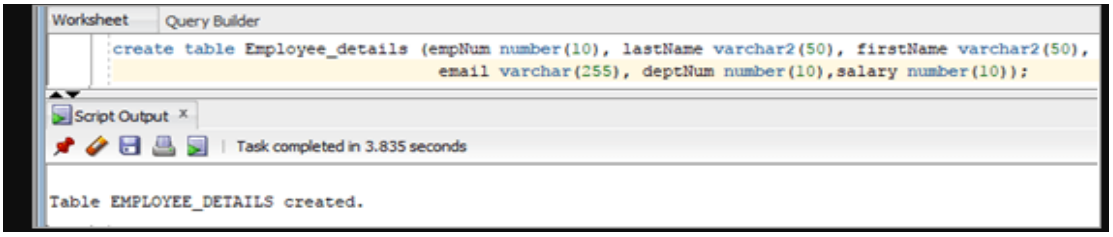
Java JDBC Connection Example
In this example, you will see how to implement the 6 basic steps to connect with database using JDBC in Java program.

Create Table
Before that, first, create one table and add some entries into it.

Below is the SQL query to create a table.

```
create table employee_details (empNum number(10), lastName varchar(50),

 firstName varchar(50), email varchar(255) , deptNum number(10), salary number(10));
```

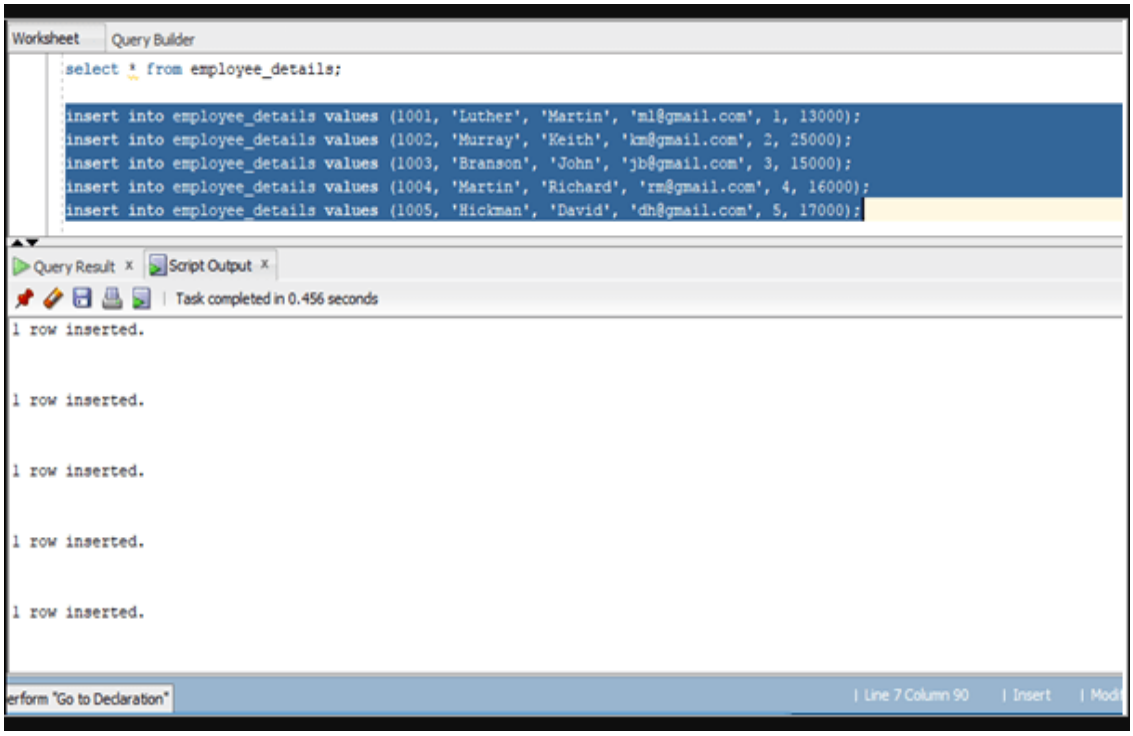
Created the "employee_details" table in Oracle DB.



Insert Data Into Table
Using the following queries, insert the data into the "employee_details" table.

```
insert into employee_details values (1001, 'Luther', 'Martin', 'ml@gmail.com', 1, 13000);

insert into employee_details values (1002, 'Murray', 'Keith', 'km@gmail.com', 2, 25000);

insert into employee_details values (1003, 'Branson', 'John', 'jb@gmail.com', 3, 15000);

insert into employee_details values (1004, 'Martin', 'Richard', 'rm@gmail.com', 4, 16000);
```

insert into employee_details values (1005, 'Hickman', 'David', 'dh@gmail.com', 5, 17000);



Java Program

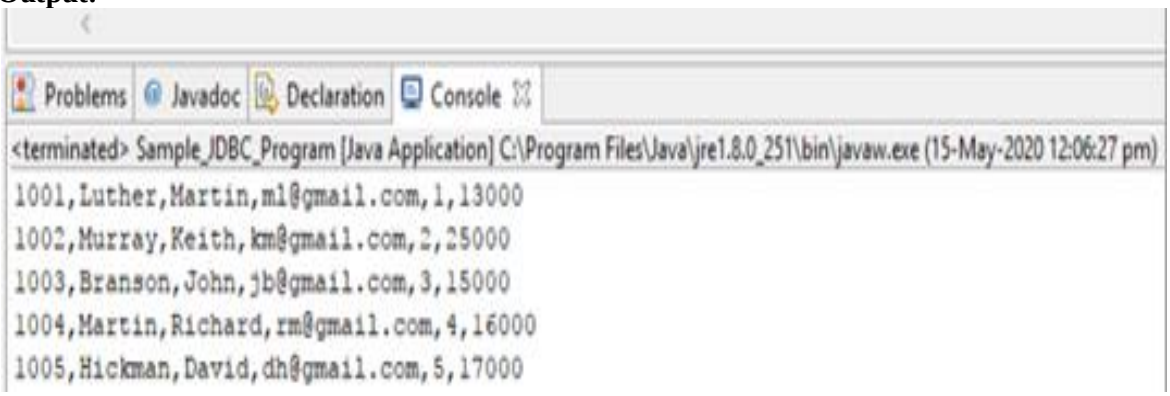
Download the JDBC jar file and import it into the Java project.

```
package com.STH.JDBC;

   // import sql package to use it in our program

   import java.sql.*;


   public class Sample_JDBC_Program {


   public static void main(String[] args) throws ClassNotFoundException, SQLException {

      // store the SQL statement in a string

      String QUERY = "select * from employee_details";

      //register the oracle driver with DriverManager

      Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
//Here we have used Java 8 so opening the connection in try statement
try(Connection conn = DriverManager.getConnection("jdbc:oracle:thin:system/pass123@localhost:1521:XE"))
{
    Statement statemnt1 = conn.createStatement();
    //Created statement and execute it
    ResultSet rs1 = statemnt1.executeQuery(QUERY);
    {
      //Get the values of the record using while loop
      while(rs1.next())
      {
        int empNum = rs1.getInt("empNum");
        String lastName = rs1.getString("lastName");
        String firstName = rs1.getString("firstName");
        String email = rs1.getString("email");
        String deptNum = rs1.getString("deptNum");
        String salary = rs1.getString("salary");
        //store the values which are retrieved using ResultSet and print it
     System.out.println(empNum + "," +lastName+ "," +firstName+ "," +email +","+deptNum +"," +salary);
      }
    }
}
catch (SQLException e) {
    //If exception occurs catch it and exit the program
    e.printStackTrace();
```

```
    }
   }
  }
```

Output:



Key points to be noted:

- First, we need to import the packages which we will be using in our Java program for the JDBC connection. So we can use the classes, subclasses, and interfaces in the packages.
- We need to register or load the driver with DriverManager before establishing a connection.
- After registering the driver, we can establish the connection and perform the operations.
- Using a statement interface we can create and execute the SQL query. For a simple SQL query, we can use the Statement interface. For insert/update/delete, we can use the PreparedStatement interface.
- After the statement execution, the results will be stored in the ResultSet object. We get the results from the ResultSet object using the next() method for more than 1 record.
- Once we are done with the database operation, we need to close the connection. So that the resource will be available for others to use.

What Are Database Applications?

"Database application" can mean two things:

One: It can refer to software running a database system. MongoDB Server or SQL Server are both software that provide the following:

1. Efficiently store and retrieve data from a file system to a network client.
2. Offer rich capabilities for querying and manipulating data from a variety of drivers.
3. Secure and authorize the access to the stored data
4. Scale
5. Provide fault tolerance and recovery (including backups) for our data

Two: It can refer to applications that are heavily coupled to a specific database and built to provide elements of that database to the end user. Some examples of such applications include:

- Online encyclopedias (Wikipedia)
- Social media websites (Facebook)
- CRM systems (Salesforce)
- Email systems (Gmail)
- E-commerce websites (Amazon)

The Purpose of Database Applications

The main purpose of database applications is to provide a way for data to be consumed either by end users (via UI) or other higher-level applications (via APIs). A database application can be used for storing or retrieving data, processing transactions, or various machine learning calculations.

For example, Facebook has a user database with which it authenticates users when they log into their Facebook account. However, Facebook also provides the ability to consume their user database by another application. This is done via a secure API Facebook exposes, and you could probably see this in many of today's platforms' authentication methods.

Another example is MongoDB Atlas, a Data-as-a-Service platform. Atlas clusters provide a variety of ways to consume the data — for example, via a driver, a Realm serverless function, or even via MongoDB Charts to provide dashboards based on data stored in Atlas.

Database Application Types (and their Pros and Cons)

Organizations and database administrators have to understand the pros and cons of the different database applications and database software out there. Databases can be categorized by the way they structure and consume data. Some use a normalized model and relations (Relational) while others use nested objects (Documents and some NoSQL flavors).

Database Application Type	Pros	Cons
Database Software - Document Store (eg., MongoDB)	• Flexible schema • Rich query language • Built-in resilience and scalability • Rich indexing strategies • Growing support communities and open-source projects • Transaction processing	• Learning curve for SQL-oriented developers • Relational schemas will need a redesign to work optimally
Database Software - Other NoSQL	• Distributed systems • More modern data stores	• Schemas are not flexible • Small support communities • Not general purpose - good for narrow use cases • No transaction processing
Database Software - Relational Databases (SQL)	• SQL-oriented • Large communities • Owned by large companies	• Expensive to start • Usually requires strong hardware to start • Not designed for the cloud era
Database Application Providers - (Amazon, Facebook)	• Offer robust services • Cloud-oriented	• Not flexible in the API • Limited ability to work with raw data • Not a pure database software

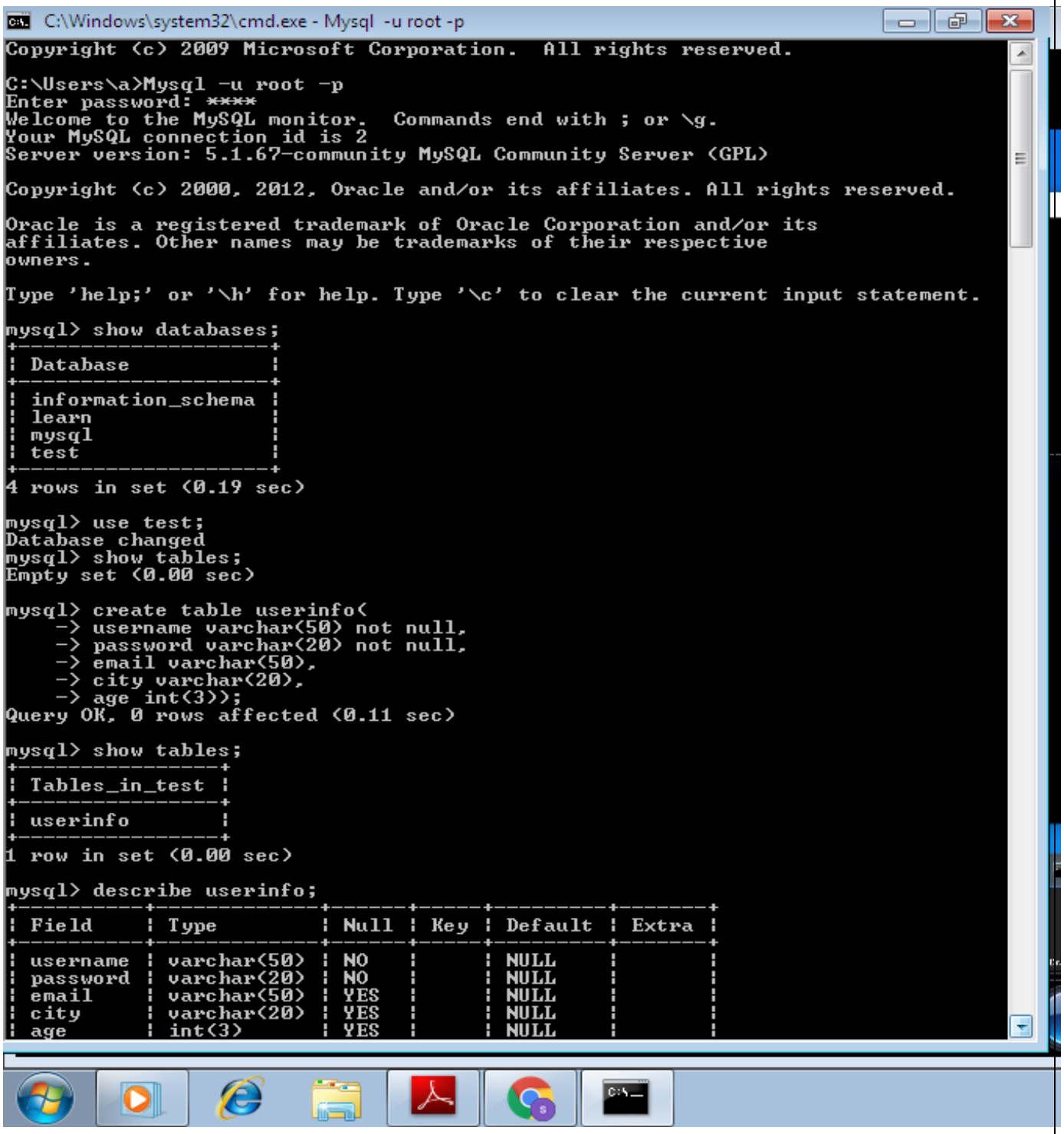
AJP Lab 8a. Insert the data from the database using JDBC

Program:

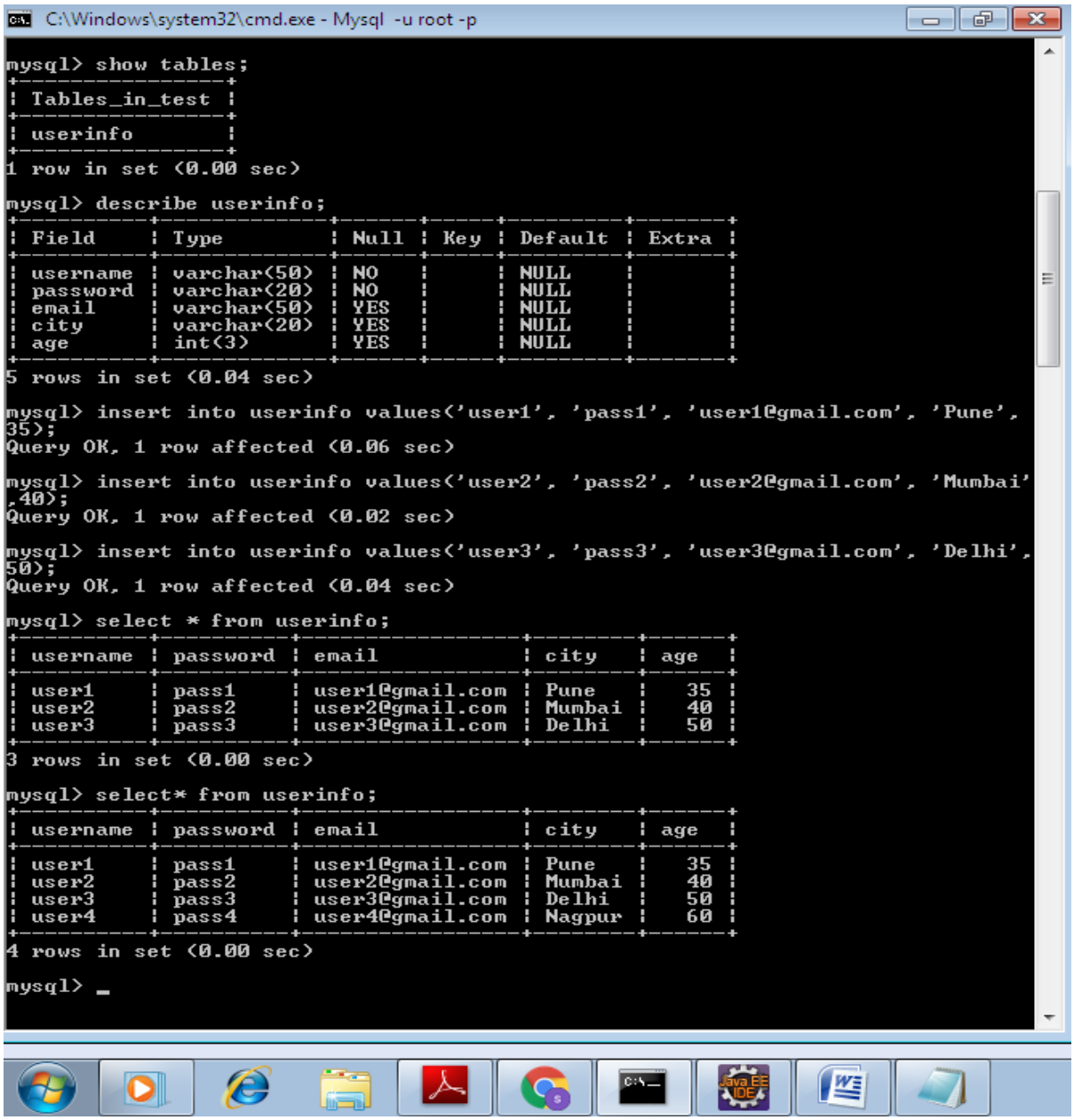
```
package Mysql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;


public class SQLPreparedStatementInsert {
public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","ro
ot");


PreparedStatement stmt = con.prepareStatement("insert into userinfo
values (?,?,?,?,?)");


stmt.setString(1,"user4");
stmt.setString(2,"pass4");
stmt.setString(3,"user4@gmail.com");
stmt.setString(4, "Nagpur");
stmt.setString(5, "60");

int i = stmt.executeUpdate();
System.out.println(i + "Records inserted..");
con.close();
}

catch(Exception e){
System.out.println(e);
}
}
}
```

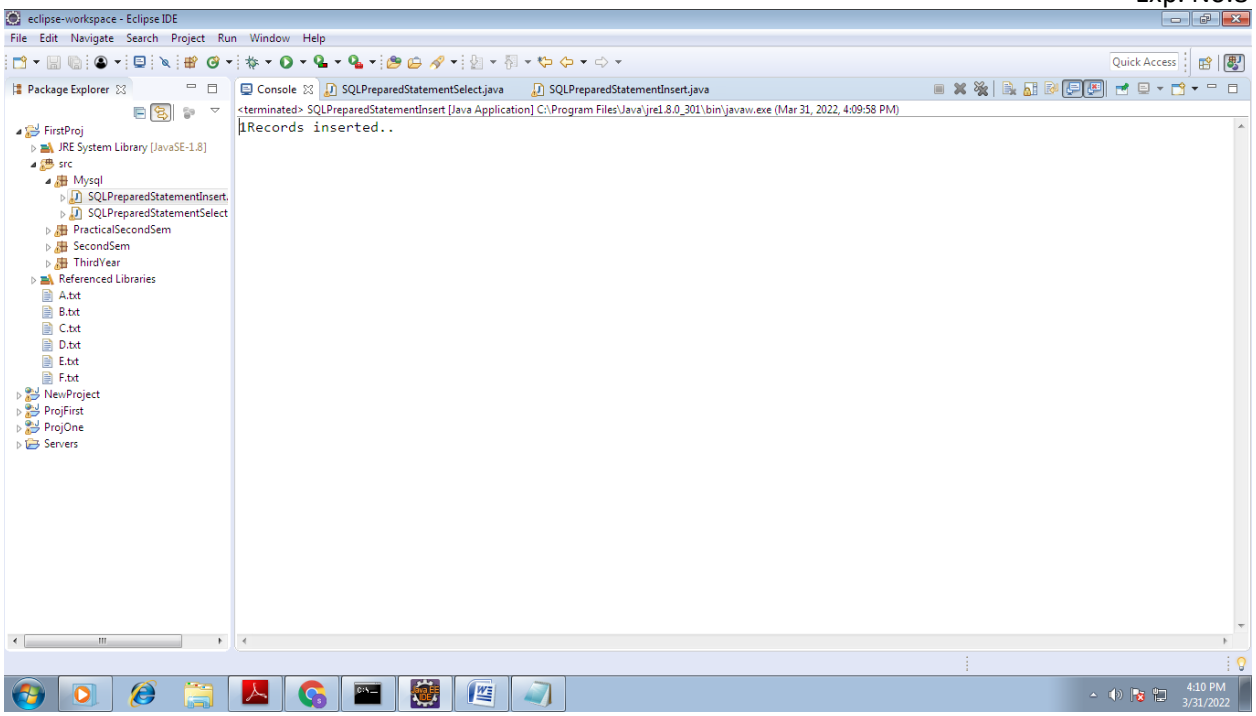
Output:

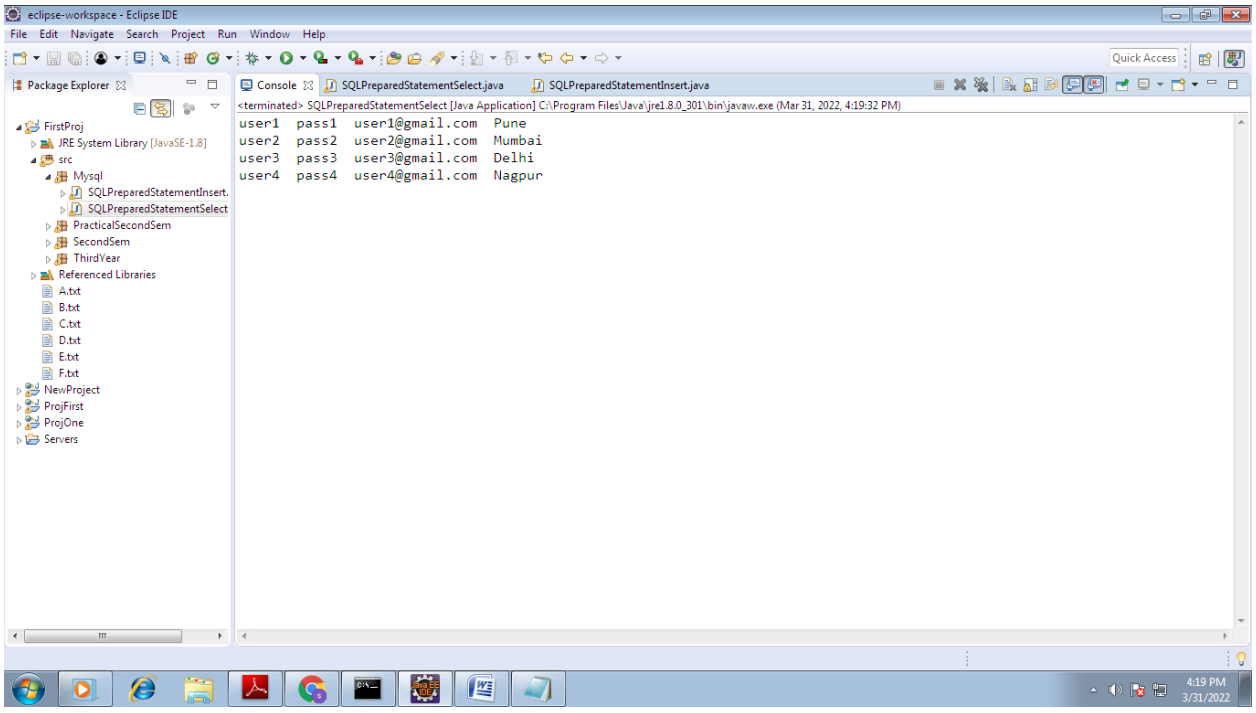
Before Insert program-



After Insert program-




eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Package Explorer
Console
SQLPreparedStatementSelect.java
SQLPreparedStatementInsert.java
<terminated> SQLPreparedStatementInsert [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Mar 31, 2022, 4:09:58 PM)
1Records inserted..
FirstProj
JRE System Library [JavaSE-1.8]
src
Mysql
SQLPreparedStatementInsert.
SQLPreparedStatementSelect
PracticalSecondSem
SecondSem
ThirdYear
Referenced Libraries
A.txt
B.txt
C.txt
D.txt
E.txt
F.txt
NewProject
ProjFirst
ProjOne
Servers
4:10 PM
3/31/2022

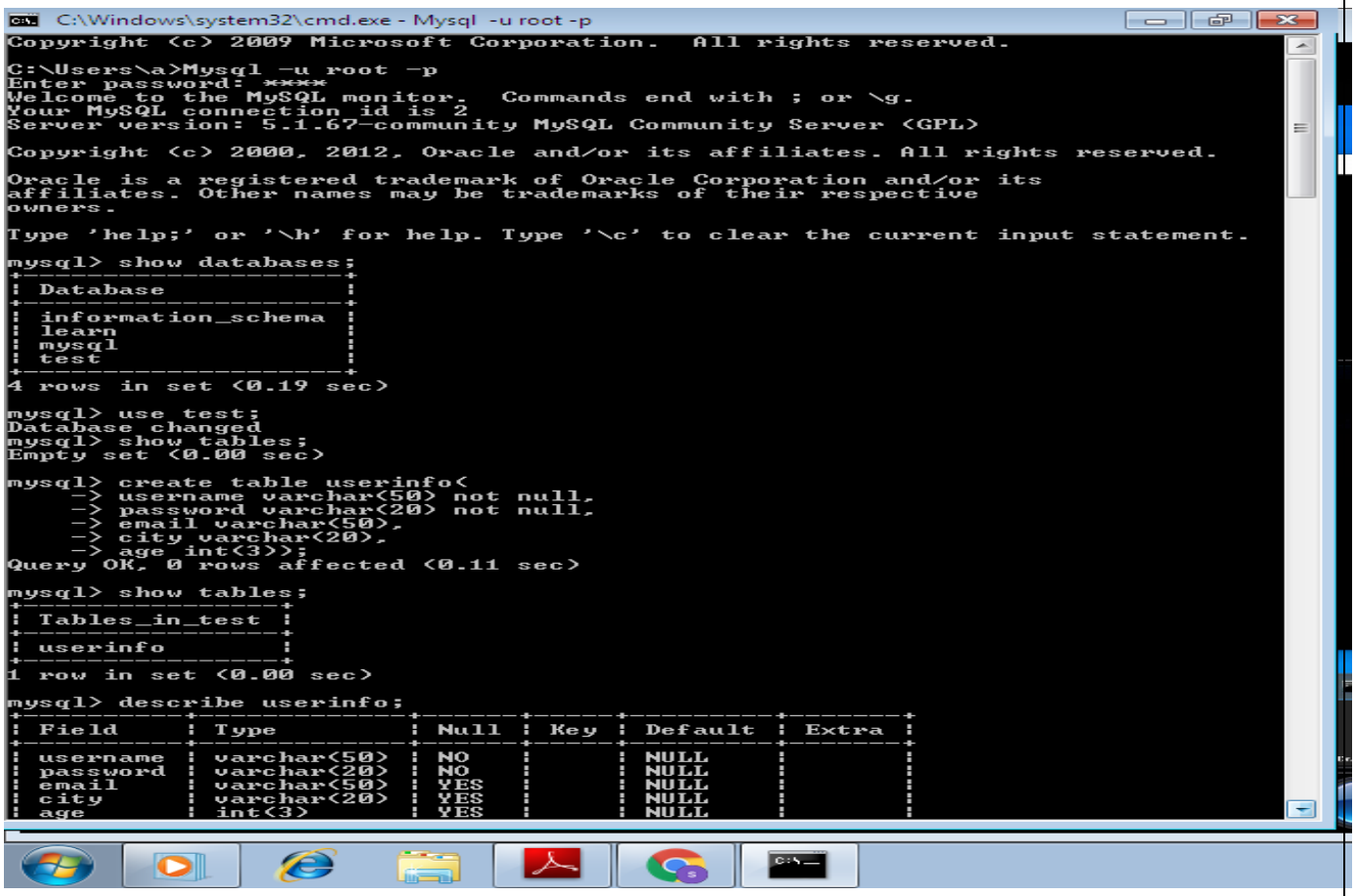

eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Quick Access
Package Explorer
Console
SQLPreparedStatementSelect.java
SQLPreparedStatementInsert.java
<terminated> SQLPreparedStatementSelect [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Mar 31, 2022, 4:19:32 PM)
user1 pass1 user1@gmail.com Pune
user2 pass2 user2@gmail.com Mumbai
user3 pass3 user3@gmail.com Delhi
user4 pass4 user4@gmail.com Nagpur
FirstProj
JRE System Library [JavaSE-1.8]
src
Mysql
SQLPreparedStatementInsert.
SQLPreparedStatementSelect
PracticalSecondSem
SecondSem
ThirdYear
Referenced Libraries
A.txt
B.txt
C.txt
D.txt
E.txt
F.txt
NewProject
ProjFirst
ProjOne
Servers
4:19 PM
3/31/2022

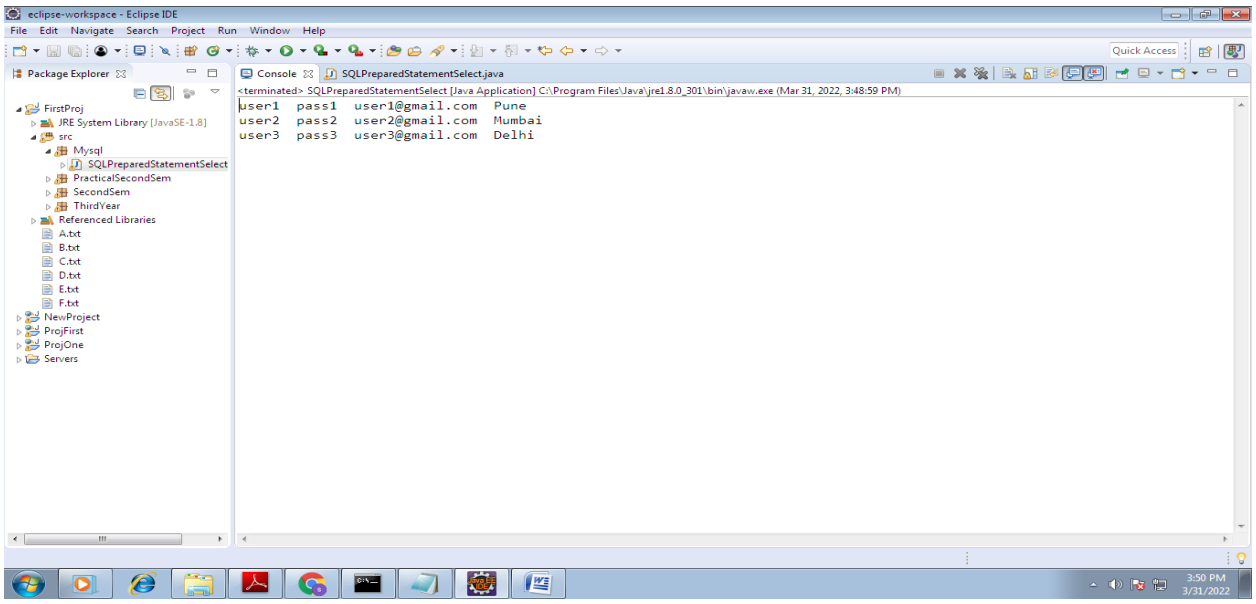
AJP Lab 8b. Rretrieve the data from the database using JDBC.

Program:

```
package Mysql;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
public class SQLPreparedStatementSelect {

public static void main(String[] args) {
try{

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

Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","ro
ot");


PreparedStatement stmt = con.prepareStatement("select * from userinfo");
ResultSet rs = stmt.executeQuery();
while(rs.next())
{

System.out.println(rs.getString(1) + "   " +
rs.getString(2) + "   " + rs.getString(3) + "   " +
rs.getString(4) + "   ");
}

con.close();
}

catch(Exception e){
System.out.println(e);
}
}
}
```

Output:


C:\Windows\system32\cmd.exe - Mysql -u root -p
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\a>Mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 2
Server version: 5.1.67-community MySQL Community Server (GPL)
Copyright (c) 2000, 2012, Oracle and/or its affiliates. All rights reserved.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
mysql> show databases;
| Database |
| information_schema |
| learn |
| mysql |
| test |
4 rows in set (0.19 sec)
mysql> use test;
Database changed
mysql> show tables;
Empty set (0.00 sec)
mysql> create table userinfo(
-> username varchar(50) not null,
-> password varchar(20) not null,
-> email varchar(50),
-> city varchar(20),
-> age int(3));
Query OK, 0 rows affected (0.11 sec)
mysql> show tables;
| Tables_in_test |
| userinfo |
1 row in set (0.00 sec)
mysql> describe userinfo;
Field	Type	Null	Key	Default	Extra
username	varchar(50)	NO		NULL	
password	varchar(20)	NO		NULL	
email	varchar(50)	YES		NULL	
city	varchar(20)	YES		NULL	
age	int(3)	YES		NULL	


eclipse-workspace - Eclipse IDE
File Edit Navigate Search Project Run Window Help
Package Explorer
FirstProj
JRE System Library [JavaSE-1.8]
src
Mysql
SQLPreparedStatementSelect
PracticalSecondSem
SecondSem
ThirdYear
Referenced Libraries
A.txt
B.txt
C.txt
D.txt
E.txt
F.txt
NewProject
ProjFirst
ProjOne
Servers
Console
SQLPreparedStatementSelect.java
<terminated> SQLPreparedStatementSelect [Java Application] C:\Program Files\Java\jre1.8.0_301\bin\javaw.exe (Mar 31, 2022, 3:48:59 PM)
user1 pass1 user1@gmail.com Pune
user2 pass2 user2@gmail.com Mumbai
user3 pass3 user3@gmail.com Delhi

AJP Lab 8C. Update the data from the database using JDBC.

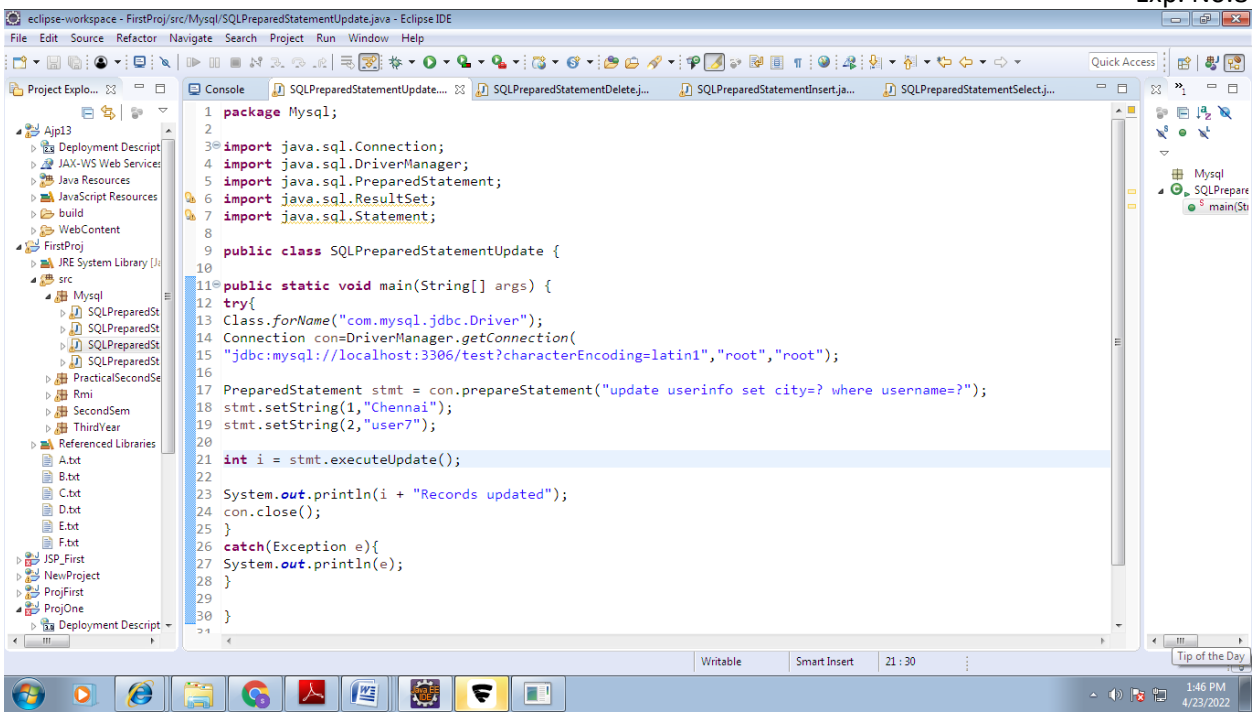
Program:

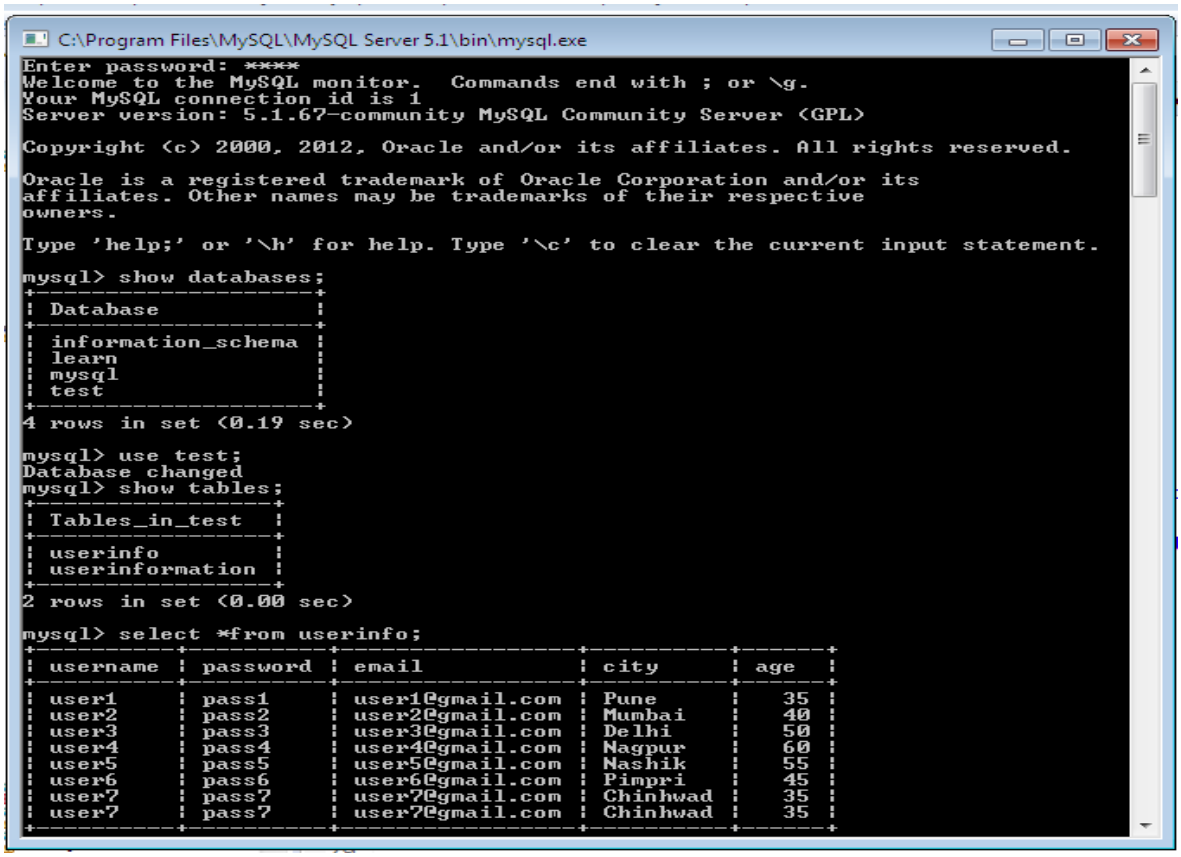
```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLPreparedStatementUpdate {

public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/test?characterEncoding=latin1","root","root");

PreparedStatement stmt = con.prepareStatement("update userinfo set  city=? where username=?");
stmt.setString(1,"Chennai");
stmt.setString(2,"user7");
int i = stmt.executeUpdate();

System.out.println(i + "Records updated");
con.close();
}
catch(Exception e){
System.out.println(e);
}
}

}
```

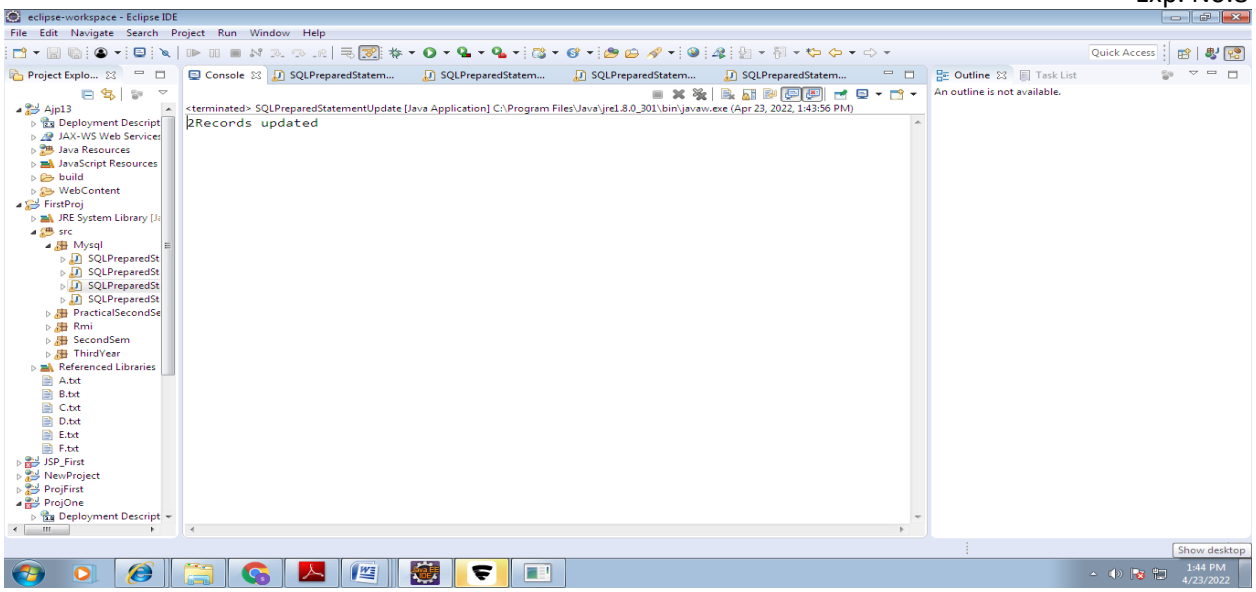
Output:

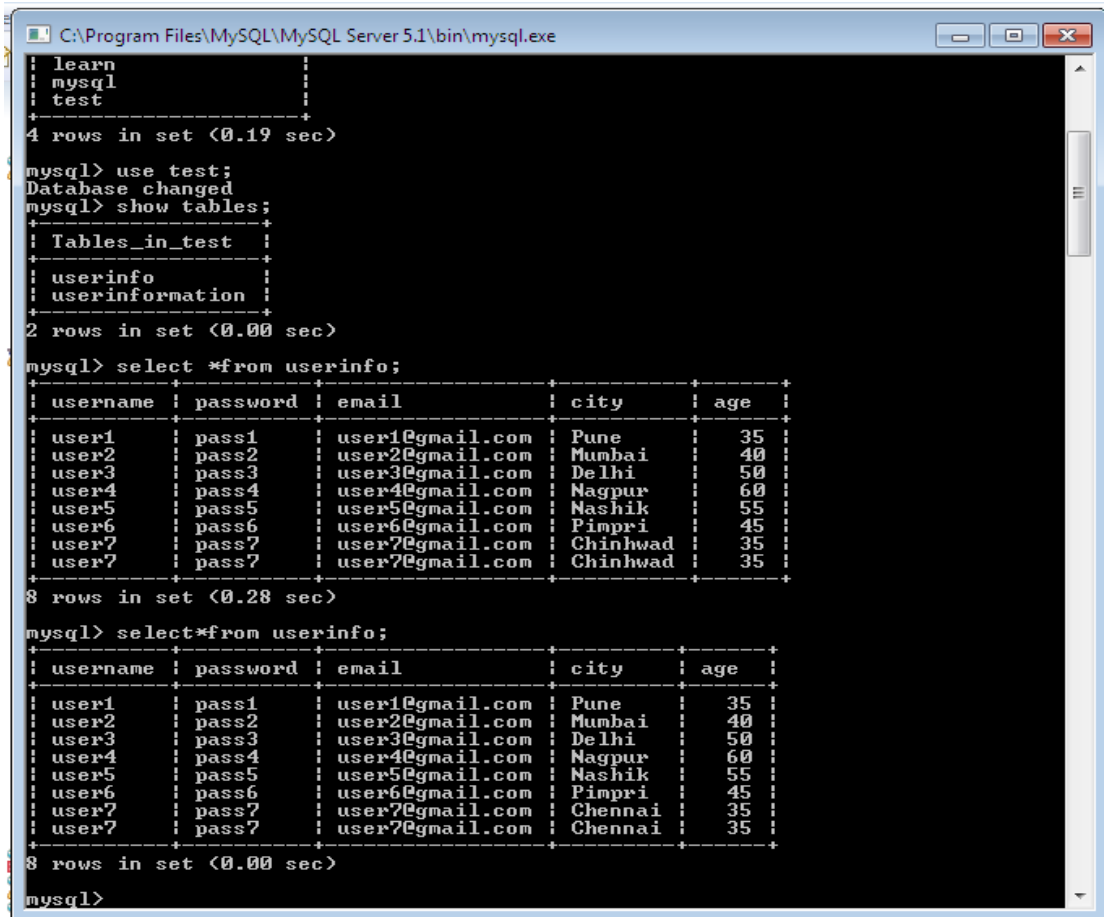
Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)

Exp. No.8









C:\Program Files\MySQL\MySQL Server 5.1\bin\mysql.exe

```
| learn              |
| mysql              |
| test               |
+--------------------+
4 rows in set (0.19 sec)

mysql> use test;
Database changed
mysql> show tables;
+------------------+
| Tables_in_test   |
+------------------+
| userinfo         |
| userinformation  |
+------------------+
2 rows in set (0.00 sec)

mysql> select *from userinfo;
+----------+----------+-----------------+----------+------+
| username | password | email           | city     | age  |
+----------+----------+-----------------+----------+------+
| user1    | pass1    | user1@gmail.com | Pune     |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai   |   40 |
| user3    | pass3    | user3@gmail.com | Delhi    |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur   |   60 |
| user5    | pass5    | user5@gmail.com | Nashik   |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri   |   45 |
| user7    | pass7    | user7@gmail.com | Chinhwad |   35 |
| user7    | pass7    | user7@gmail.com | Chinhwad |   35 |
+----------+----------+-----------------+----------+------+
8 rows in set (0.28 sec)

mysql> select*from userinfo;
+----------+----------+-----------------+----------+------+
| username | password | email           | city     | age  |
+----------+----------+-----------------+----------+------+
| user1    | pass1    | user1@gmail.com | Pune     |   35 |
| user2    | pass2    | user2@gmail.com | Mumbai   |   40 |
| user3    | pass3    | user3@gmail.com | Delhi    |   50 |
| user4    | pass4    | user4@gmail.com | Nagpur   |   60 |
| user5    | pass5    | user5@gmail.com | Nashik   |   55 |
| user6    | pass6    | user6@gmail.com | Pimpri   |   45 |
| user7    | pass7    | user7@gmail.com | Chennai  |   35 |
| user7    | pass7    | user7@gmail.com | Chennai  |   35 |
+----------+----------+-----------------+----------+------+
8 rows in set (0.00 sec)

mysql>
```

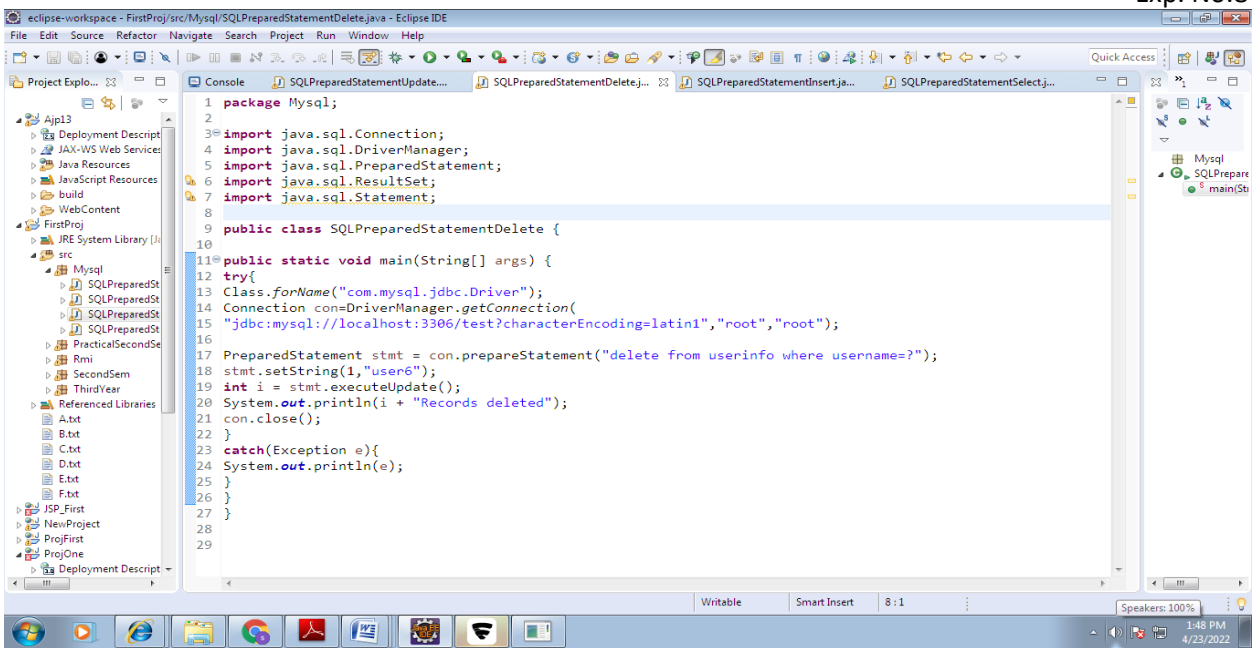
AJP Lab 8D. Delete the data from the database using JDBC.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;

public class SQLPreparedStatementDelete {

public static void main(String[] args) {
try{
Class.forName("com.mysql.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/logininfo?characterEncoding=latin1","root","root");

PreparedStatement stmt = con.prepareStatement("delete from userinfo where username=?");
stmt.setString(1,"user2");
int i = stmt.executeUpdate();
System.out.println(i + "Records deleted");
con.close();
}
catch(Exception e){
System.out.println(e);
}
}
}
```

Output:

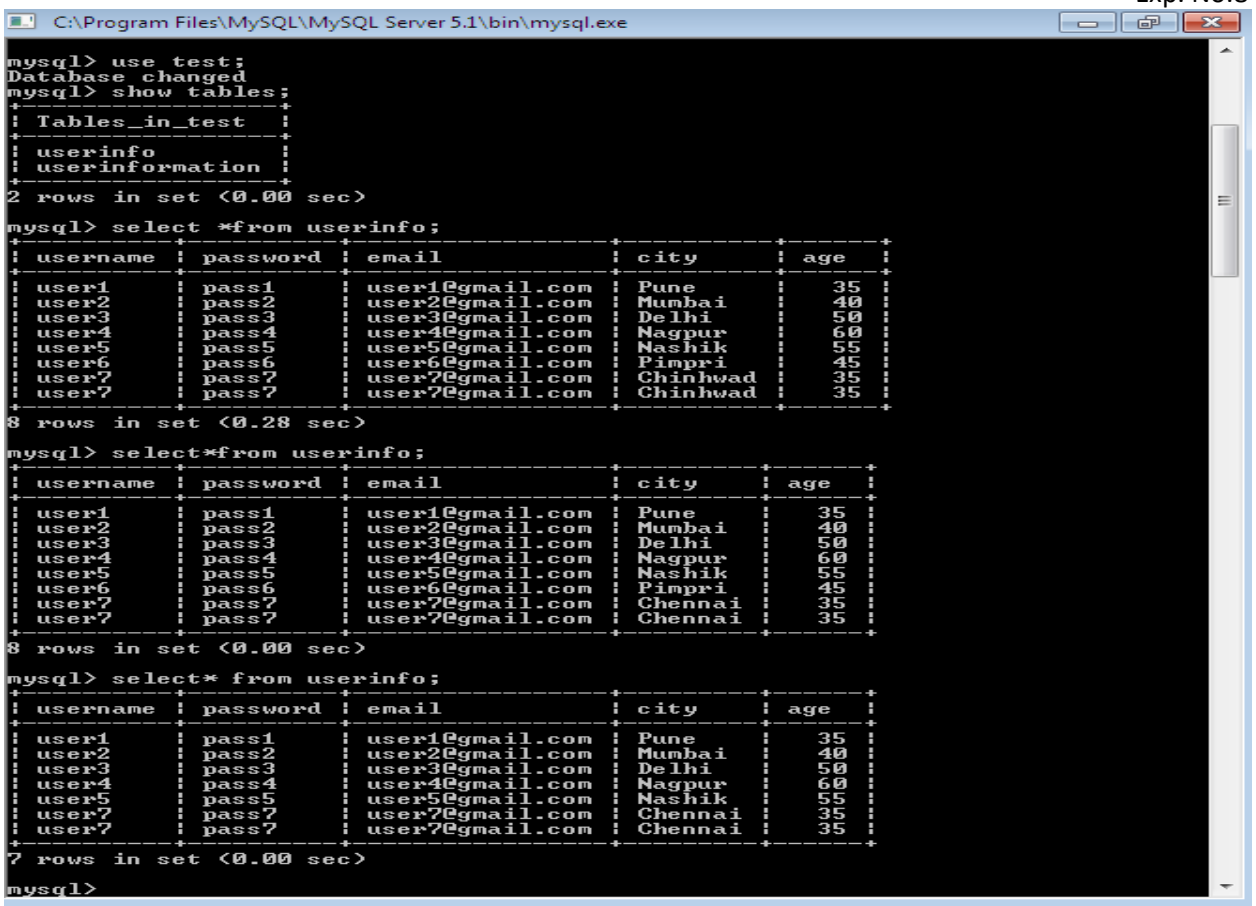
Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)

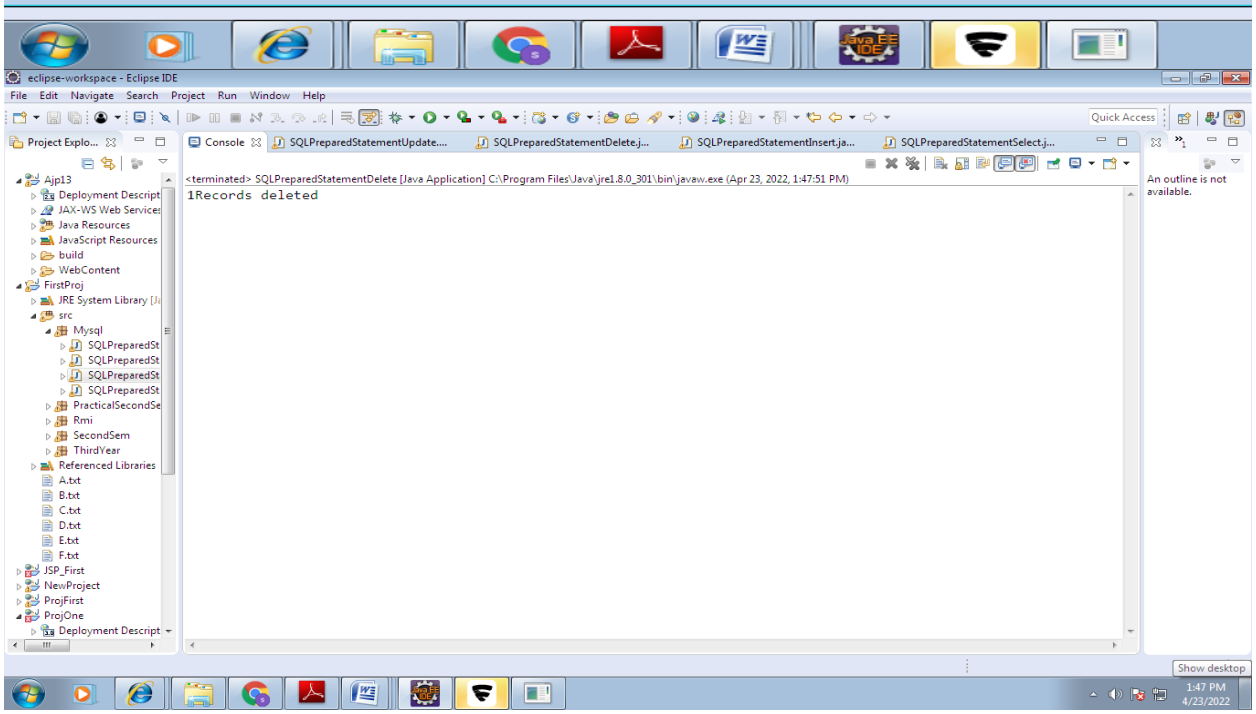
Exp. No.8



Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)

Exp. No.8





Conclusion:

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. What is JDBC driver?
2. What are the different types of JDBC drivers in Java? Explain each with an example.
3. Which JDBC driver is fastest and used more commonly?
4. What is DriverManager in JDBC?

Experiment No. 9

Aim of the Experiment: Write a simple JSP page to display a simple message (It may be a simple html page).

Objective:

To display a message using JSP page.

Resources: Eclipse IDE 2018, JDK 1.8.0 is required, Apache tomcat 9.0 server

Course Outcome Addressed: CO6

Theory:

JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>. A JSP page looks similar to an HTML page, but a JSP page also has Java code in it. We can put any regular Java Code in a JSP file using a scriplet tag which start with <% and ends with %>. JSP pages are used to develop dynamic responses.

A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically.

JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI).

Advantages of JSP in comparison with the CGI:

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
- JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.

- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP, etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

Advantages of JSP:

vs. Active Server Pages (ASP)

The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

vs. Pure Servlets

It is more convenient to write (and to modify!) regular HTML than to have plenty of println statements that generate the HTML.

vs. Server-Side Includes (SSI)

SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

vs. JavaScript

JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

vs. Static HTML

Regular HTML, of course, cannot contain dynamic information.

JSP Processing:

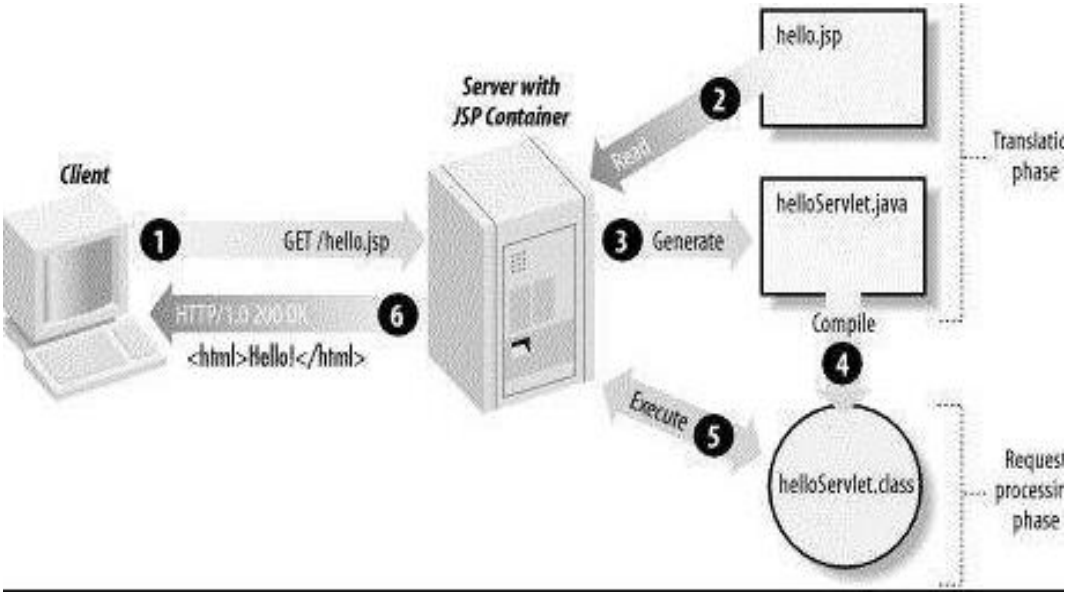
The following steps explain how the web server creates the Webpage using JSP −

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with .jsp instead of .html.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to println( ) statements

and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page.

- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is furthur passed on to the web server by the servlet engine inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be seen in the following diagram



Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with the other scripting languages (such as PHP) and therefore faster.

So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet

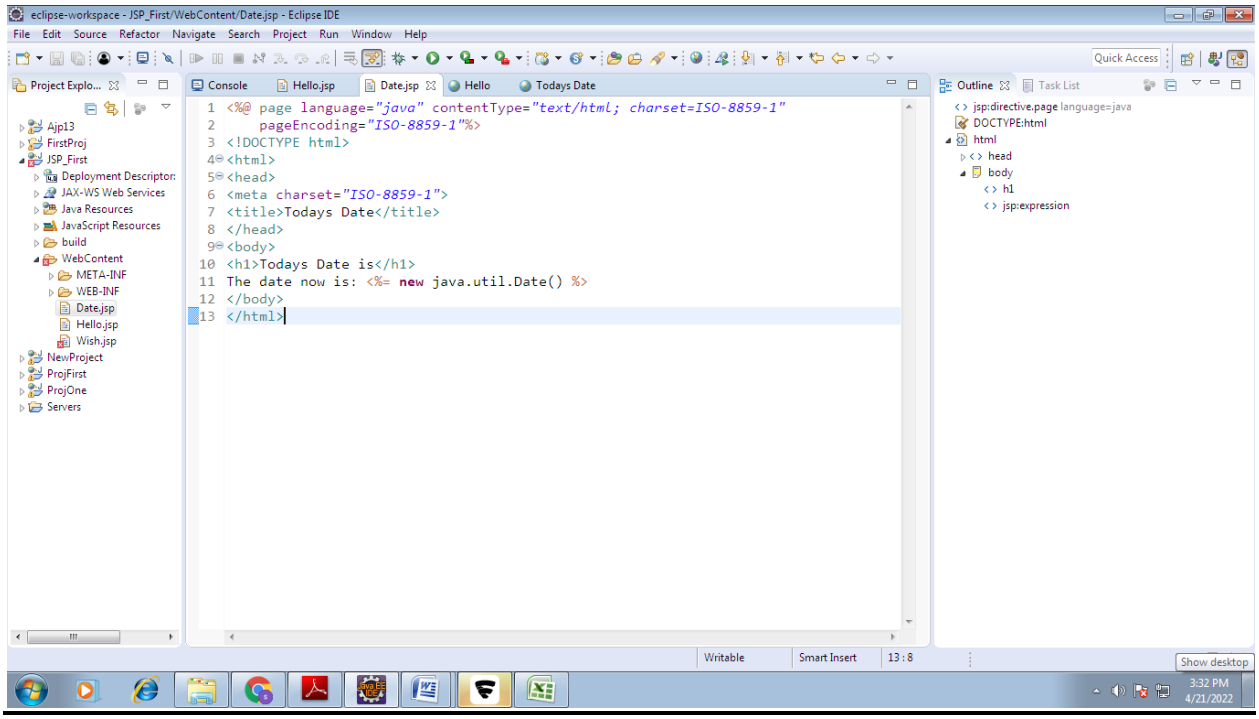
Steps for creating a JSP Page in Eclipse:

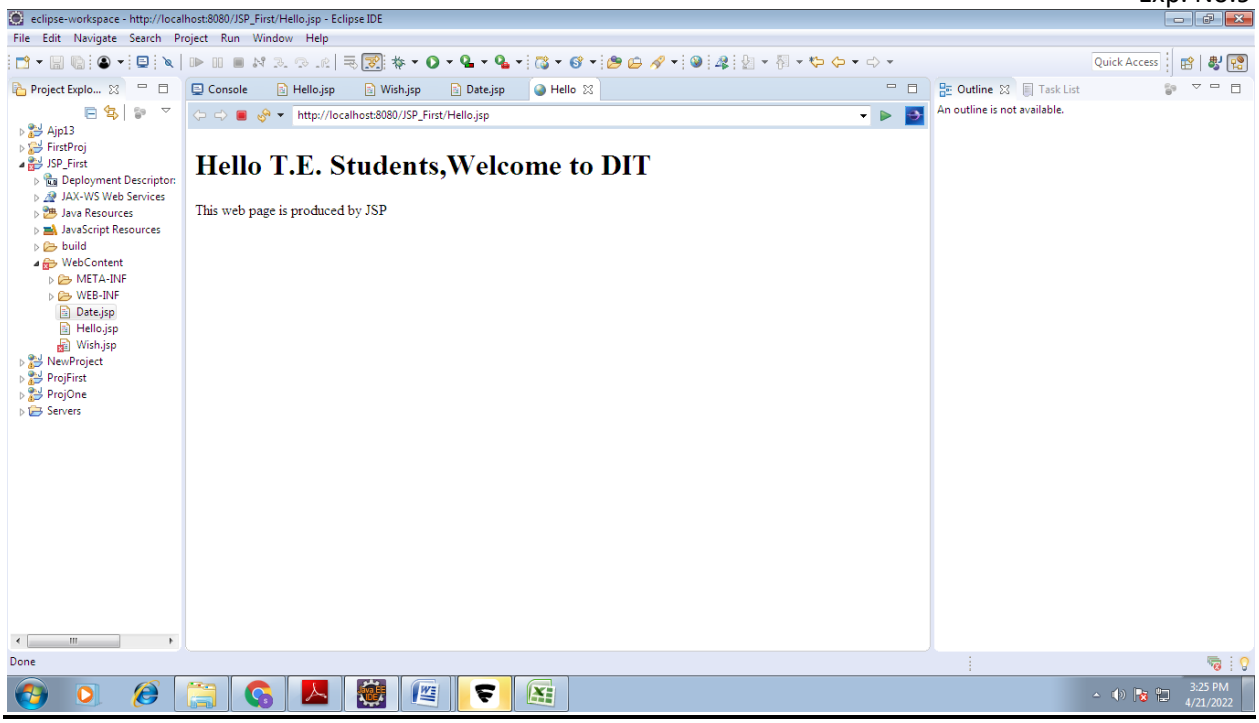
1. Open Eclipse, Click on **New → Dynamic Web Project**
2. Give a name to your project and click on OK
3. You will see a new project created in Project Explorer
4. To create a new JSP file right click on Web Content directory, **New → JSP file**
5. Give a name to your JSP file and click Finish.
6. Write something in your JSP file. The complete HTML and the JSP code, goes inside the <body> tag, just like HTML pages.
7. To run your project, right click on **Project**, select **Run As → Run on Server**
8. To start the server, Choose existing server name and click on finish.
9. See the Output in your browser.

SOURCE CODE for Hello:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Hello</title>
</head>
<body>
<h1>Hello T.E. Students,Welcome to DIT</h1>
<%
  out.println("<p>This web page is produced by JSP");
 %>
</body>
</html>
```

OUTPUT:

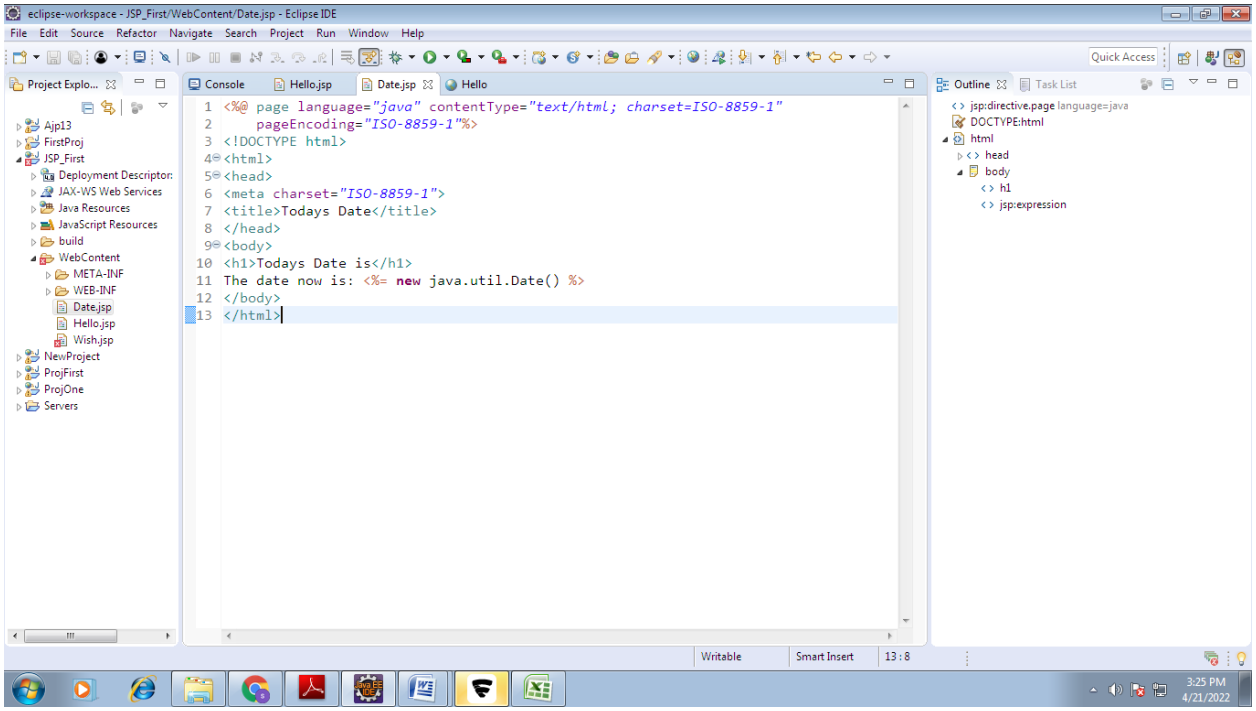

eclipse-workspace - JSP_First/WebContent/Date.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Project Explo...
Ajp13
FirstProj
JSP_First
Deployment Descriptor:
JAX-WS Web Services
Java Resources
JavaScript Resources
build
WebContent
META-INF
WEB-INF
Date.jsp
Hello.jsp
Wish.jsp
NewProject
ProjFirst
ProjOne
Servers
Console
Hello.jsp
Date.jsp
Hello
Todays Date
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2 pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Todays Date</title>
8 </head>
9 <body>
10 <h1>Todays Date is</h1>
11 The date now is: <%= new java.util.Date() %>
12 </body>
13 </html>
Outline
Task List
jsp:directive.page language=java
DOCTYPE:html
html
head
body
h1
jsp:expression
Writable
Smart Insert
13 : 8
Show desktop
3:32 PM
4/21/2022

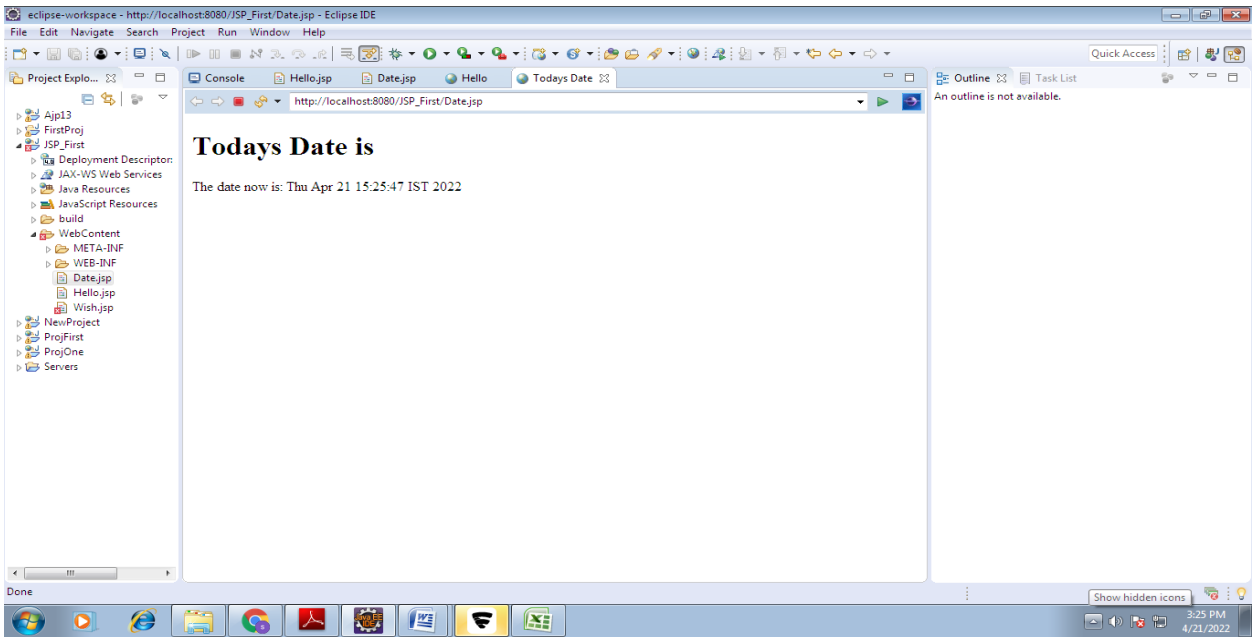


SOURCE CODE for Date:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Todays Date</title>
</head>
<body>
<h1>Todays Date is</h1>
The date now is: <%= new java.util.Date() %>
</body>
</html>
```

OUTPUT:


eclipse-workspace - JSP_First/WebContent/Date.jsp - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explo...
Console
Hello.jsp
Date.jsp
Hello
Outline
Task List
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2 pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Todays Date</title>
8 </head>
9 <body>
10 <h1>Todays Date is</h1>
11 The date now is: <%= new java.util.Date() %>
12 </body>
13 </html>
Writable
Smart Insert
13 : 8


eclipse-workspace - http://localhost:8080/JSP_First/Date.jsp - Eclipse IDE
Todays Date
http://localhost:8080/JSP_First/Date.jsp
Todays Date is
The date now is: Thu Apr 21 15:25:47 IST 2022
An outline is not available.
Done

Conclusion:

.

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions:

1. What is JSP?
2. How JSP work?
3. What are the life-cycle methods for a JSP?
4. List out some advantages of using JSP.
5. What are Servlets?

Experiment No. 10

Aim of the Experiment: Create a simple calculator application using servlet.

Objective:

To create a simple calculator application using servlet.

Resources: Eclipse IDE 2018, JDK 1.8.0 is required, Apache tomcat 9.0 server

Course Outcome Addressed: CO6

Theory:

Servlets are small programs that execute on the server side. Servlets are pieces of Javasource code that add functionality to a web server.

Servlet provides full support for sessions, a way to keep track of a particular user over time as a website's pages are being viewed. They also can communicate directly with aweb server using a standard interface.

Servlets can be created using the **javax.servlet** and **javax.servlet.http** packages, which are a standard part of the Java's enterprise edition, an expanded version of the Java classlibrary that supports large-scale development projects.

Running servlets requires a server that supports the technologies. Several web servers, each of which has its own installation, security and administration procedures, support Servlets. The most popular one is the Tomcat- an open source server developed by the Apache Software Foundation in cooperation with Sun Microsystems version 5.5 of Tomcat supports Java Servlet.

Getting Tomcat

The software is available a free download from Apache's website at the address http://jakarta.apache.org/tomcat. Several versions are available: Linux users should download the **rpm** of Tomcat.

The javax.servlet package

The important interfaces and classes are described in the table below.

Interface	Description
Servlet	A java servlet must implement the Servlet interface. This interface defines methods to initialize a servlet, to service requests, and to remove a servlet from the server. These are known as life-cycle methods.

ServletConfig	The ServletConfig interface is used by the server to pass configuration information to a servlet. Its methods are used by the servlet to retrieve this information.
ServletRequest	The ServletRequest interface encapsulates a client request for service. It defines a number of methods for obtaining information about the server, requester, and request.
ServletResponse	The ServletResponse interface is used by a servlet to respond to a request by sending information back to the client.
ServletContext	The ServletContext interface defines the environment in which an applet is executed. It provides methods that are used by applets to access environment information.
SingleThreadModel	The SingleThreadModel interface is used to identify servlets that must be thread-safe. If a servlet implements this interface, the Web server will not concurrently execute the service() method of more than one instance of the servlet.
Class	**Description**
GenericServlet	The GenericServlet class implements the Servlet interface. You can subclass this class to define your own servlets.
ServletInputStream	The ServletInputStream class is used to access request information supplied by a Web client. An object of this class is returned by the getInputStream() method of the ServletRequest interface.
ServletOutputStream	The ServletOutputStream class is used to send response information to a Web client. An object of this class is returned by the getOutputStream() method of the ServletResponse interface.

The javax.servlet.http package

Interface	Description
HttpServletRequest	The HttpServletRequest interface extends the ServletRequest interface and adds methods for accessing the details of an HTTP request.
HttpServletResponse	The HttpServletResponse interface extends the ServletResponse interface and adds constants and methods for returning HTTP-specific responses.
HttpSession	This interface is implemented by servlets to enable them to support browser- server sessions that span multiple HTTP request-response pairs. Since HTTP is a stateless protocol, session state is maintained externally using client-side cookies or URL rewriting. This interface provides methods for reading and writing state values and managing sessions.
HttpSessionContext	This interface is used to represent a collection of HttpSession

	objects that areassociated with session IDs.
Class	**Description**
HttpServlet	Used to create HTTP servlets. The HttpServlet class extends the GenericServlet class.
Cookie	This class represents an HTTP cookie. Cookies are used to maintain session state over multiple HTTP requests. They are named data values that are created on the Web server and stored on individual browser clients. The Cookie class provides the method for getting and setting cookie values and attributes.

Servlet Life Cycle

A servlet's life cycle methods function similarly to the life cycle methods of applets.

- The **init(ServletConfig)** method is called automatically when a web server first begins a servlet to handle the user's request. The init() method is called only once. ServletConfig is an interface in the javax.servlet package, containing the methods to find out more about the environment in which a servlet is running.
- The servlet action is in the **service()** method. The service() method checks the HTTP request type (GET, POST, PUT, DELETE etc.) and calls doGet(), doPost(),doPut(), doDelete() etc. methods. A GET request results from normal request for a URL or from an HTML form that has no METHOD specified. The POST requestresults from an HTML form that specifically lists POST as the METHOD.
- The **destroy()** method is called when a web server takes a servlet offline.

Using Servlets

One of the main tasks of a servlet is to collect information from a web user and present something back in response. Collection of information is achieved using form, which is a group of text boxes, radio buttons, text areas, and other input fields on the web page. Each field on a form stores information that can be transmitted to a web server and then sent to a Java servlet. web browsers communicate with servers by using Hypertext Transfer Protocol (HTTP).

- Form data can be sent to a server using two kinds of HTTP requests: get and post. When web page calls a server using **get** or **post**, the name of the program that handles the request must be specified as a web address, also called uniform resource locator (URL). A get request affixes all data on a form to the end of a URL. A post request includes form data as a header and sent separately from the URL. This is generally preferred, and it's required when confidential information is being collected on the form.
- Java servlets handle both of these requests through methods inherited from the HTTPServlet class: **doGet(HttpServletRequest, HttpServletResponse)** and **doPost(HttpServletRequest, HttpServletResponse).** These methods throw two kinds of exceptions:

ServletException, part of javax.servlet package, and IOException, an exception in the java.io package.

- The **getparameter(String)** method is used to retrieve the fields in a servlet withthe name of the field as an argument. Using an HTML document a servlet communicates with the user.
- While preparing the response you have to define the kind of content the servlet issending to a browser. The **setContentType(String)** method is used to decide the type of response servlet is communicating. Most common form of response is written using an HTML as: **setContentType("text/html").**
- To send data to the browser, you create a servlet output stream associated with the browser and then call the **println(String)** method on that stream. The **getWriter()** method of HttpServletResponse object returns a stream. which can be used to send a response back to the client.

Example

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

import javax.servlet.http.*;

public class MyHttpServlet extends HttpServlet

{

  public void doGet(HttpServletRequest req,HttpServletResponse res) throws
ServletException, IOException

{

    // Use "req" to read incoming request

    // Use "res" to specify the HTTP response status

    //Use req.getParameter(String) or getParameterValues(String) to obtain
parameters

        PrintWriter out = res.getWriter();//stream for output

    // Use "out" to send content to browser

  }

}
```

Request and Response methods

ServletRequest methods	
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 postrequest.
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.
String getServerName()	Returns the host name of the server that received therequest.
int getServerPort()	Returns the port number on which this request was received.
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, if the user has been authenticated, or null if the user has not been authenticated.

ServletResponse methods	
ServletOutputStream 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(intlen)	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(Cookiecookie)	Used to add a Cookie to the header of the response tothe client.
void sendError(int ec)	Sends an error response to the client using the specifiedstatus.
void sendError(int ec, String messg)	Sends an error response to the client using the specifiedstatus code and descriptive message.
void sendRedirect(Stirng 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.

Writing, Compiling and Running Servlet

Type the first sample program of the self-activity section. After saving this servlet,compile it with the Java compiler as: javac SimpleServlet.java. After compilation a class file with name SimpleServlet.class is created.

To make the servlet available, you have to publish this class file in a folder on your web server that has been designated for Java servlets. Tomcat provides the classes sub-folder to deploy this servlet's class file. Copy this class file in this classes sub-folder, which is available on the path: tomcat/webapps/ WEB-INF/classes. Now edit the web.xml file available under WEB-INF sub-folder with the following lines:

```
<servlet>
        <servlet-name>SimpleServlet</servlet-name>
        <servlet-class>SimpleServlet</servlet-class>

</servlet>
```

```
<servlet-mapping>
        <servlet-name>SimpleServlet</servlet-name>
        <url-pattern>/SimpleServlet</url-pattern>
</servlet-mapping>
```

Repeat the above sequence of line to run every newly created servlet. Remember, these line lines must be placed somewhere after the <web-app> tag and before the closing

</web-app> tag.

After adding these lines, save web.xml file. Restart the Tomcat service and run the servlet by loading its address with a web browser as: http://localhost:8080/FirstServlet.

Source Code:

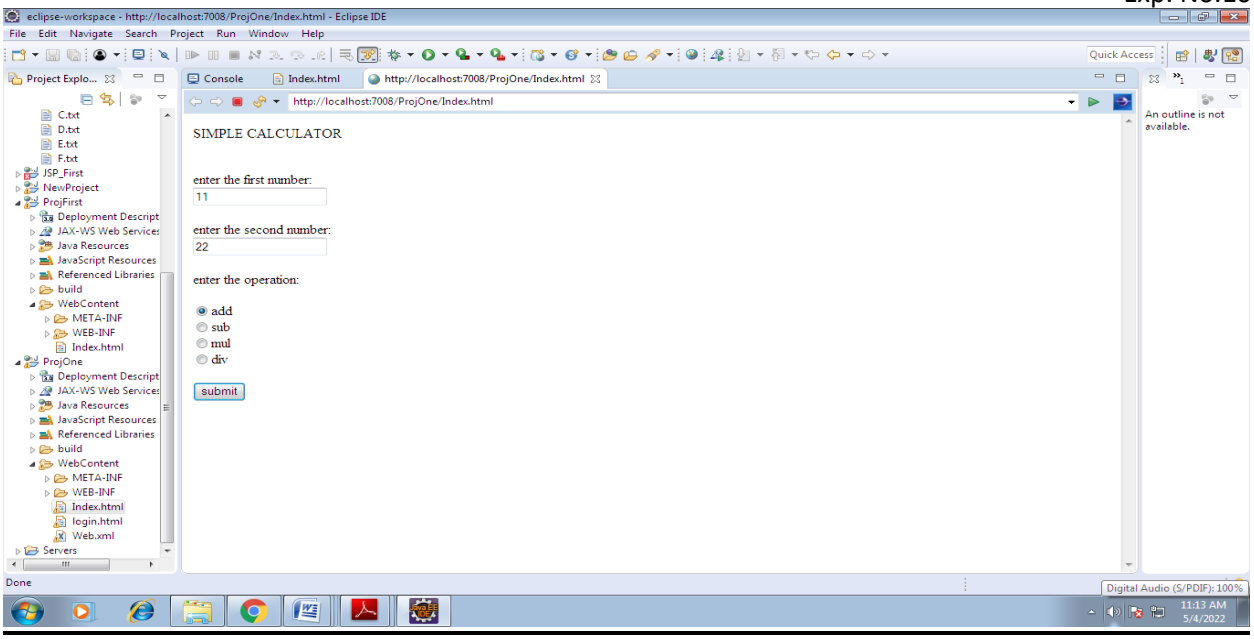
Index.html

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Calculator Application Using Servlet</title>
</head>
<body>
<form method=get action="CalculatorServlet" >
Enter First Number <input type="text" name="num1"><br>
Enter Second Number <input type="text" name="num2" ><br>
Select an Operation<input type="radio" name="opr" value="+">
ADDTION <input type="radio" name="opr" value="-">
SUBSTRACTION <input type="radio" name="opr" value="*">
MULTIPLY <input type="radio" name="opr" value="/">
DIVIDE <br><input type="reset">
<input type="submit" value="Calculate" >
</form>
</body>
</html>
```

CalculatorServle.java

```
package AjpPr10;

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

public class CalculatorServlet extends HttpServlet {
public void doGet(HttpServletRequest req, HttpServletResponse res) throws ServletException {
int result = 0;
try {
String number1 = req.getParameter("num1");
String number2 = req.getParameter("num2");
String operator = req.getParameter("opr");
int x = Integer.parseInt(number1);
int y = Integer.parseInt(number2);
if(operator == "+") {
result = x + y;
}
else if(operator == "-") {
result = x - y;
}
else if(operator == "*") {
result = x * y;
}
else if(operator == "/") {
result = x/y;
}
PrintWriter p = res.getWriter();
p.println(result);
}
catch(Exception e) {}
}
}
```

OUTPUT:



Conclusion:

.

References:

Herbert Schildt , "Java : The Complete Reference" Tata McGraw-Hill (7th Edition).

Questions: What are Servlets?

1. What are the major tasks of servlets?
2. Explain servlet life cycle.
3. What is difference between Get and Post method?
4. What is HTTPServletRequest class?