Lab 1: JAVA AWT AND SWING

Objective:

- To get familiar with java awt and swing components .

Theory:

Java AWT: It is one of the original libraries provided by Java for building graphical user interfaces (GUIs). It provides basic components like buttons, labels, and text fields. However, AWT is platform-dependent, meaning that it uses native system resources (OS components), which can lead to inconsistent appearance and behavior across platforms.

Java Swing: It is a more advanced GUI toolkit built on top of AWT. Unlike AWT, Swing is platform-independent, lightweight, and more flexible. It offers a rich set of components like tables, trees, buttons, sliders, and text boxes, all customizable in terms of look and feel. Swing components are written in Java and rendered by the Java runtime, ensuring consistency across platforms. Key Swing components include JFrame, JButton, JPanel, JTable, and many more.

Layout Manager

A **Layout Manager** in Java is responsible for arranging the components in a container (like a window or panel). It controls how components are positioned and sized within the container, allowing the GUI to adapt to different screen sizes or resolutions. Java provides several built-in layout managers, such as:

- **Flow Layout**: Arranges components in a row, wrapping them to the next line if necessary.
- **Border Layout**: Divides the container into five regions: North, South, East, West, and Center.
- **Grid Layout**: Arranges components in a grid, with equal-sized cells.
- **Box Layout**: Arranges components either vertically or horizontally in a box-like fashion.

Action Listener

An **ActionListener** is an interface in Java used for handling action events like button clicks, menu selections, etc. When a user interacts with a component (e.g., clicks a button), the ActionListener is triggered, executing the corresponding code. To use it, you implement the actionPerformed(ActionEvent e) method, which defines the actions that occur when an event is fired.

LAB WORK

1. Write a Java swing program to display a frame. Set the background color of frame to CYAN and use setdefaultCloseOperation() method.

```
import java.awt.*;
import javax.swing.*;
public class Question1 {
   public static void main(String[] args) {
       JFrame f = new JFrame("lab 1");
       f.setSize(500, 500);
       f.getContentPane().setBackground(Color.CYAN);
       f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       f.setVisible(true);
   }
}
```

2. Create a frame as below illustrating the concept of Graphics2D.

```
import javax.swing.*;
import java.awt.*;
public class Question2 extends JPanel {
   public static void main(String[] args) {
       JFrame f = new JFrame("lab 1");
       f.setSize(250, 250);
       f.getContentPane().setBackground(Color.white);
       f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       Question2 obj = new Question2();
       f.add(obj);
       f.setVisible(true);
   }
   @Override
```

```
    protected void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        g2d.setColor(Color.yellow);
        g2d.fillOval(10, 10, 200, 200);
       g2d.setColor(Color.BLACK);
        g2d.fillOval(55, 65, 30, 30);
        g2d.fillOval(135, 65, 30, 30);
        g2d.fillOval(50, 130, 120, 30);
        g2d.setColor(Color.yellow);
        g2d.fillOval(50, 130, 120, 26);
    }
}
```

3. Ilustrate the concept of Layout Management in Java Swing.

Hint: Add components in a frame with different layouts

```
import javax.swing.*;
import java.awt.*;
public class Question3 {
    public static void main(String[] args) {
        JFrame f = new JFrame("Lab 1");
        f.setSize(800, 600);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        f.setLayout(new BorderLayout(10, 10));
        JPanel p = new JPanel(new BorderLayout(5, 5));
        p.setBorder(BorderFactory.createTitledBorder("BorderLayout"));
        p.add(new JButton("North"), BorderLayout.NORTH);
        p.add(new JButton("South"), BorderLayout.SOUTH);
        p.add(new JButton("East"), BorderLayout.EAST);
        p.add(new JButton("West"), BorderLayout.WEST);
```

```
p.add(new JButton("Center"), BorderLayout.CENTER);

JPanel fp = new JPanel(new FlowLayout(FlowLayout.LEFT, 10, 10));
fp.setBorder(BorderFactory.createTitledBorder("FlowLayout"));
fp.add(new JButton("Button 1"));
fp.add(new JButton("Button 2"));
fp.add(new JButton("Button 3"));
fp.add(new JButton("Button 4"));

JPanel bp = new JPanel();
bp.setLayout(new BoxLayout(bp, BoxLayout.Y_AXIS));
bp.setBorder(BorderFactory.createTitledBorder("BoxLayout"));
bp.add(new JButton("Button A"));
bp.add(new JButton("Button B"));
bp.add(new JButton("Button C"));

JPanel gp = new JPanel(new GridLayout(2, 2, 10, 10));
gp.setBorder(BorderFactory.createTitledBorder("GridLayout"));
gp.add(new JButton("Button 1"));
gp.add(new JButton("Button 2"));
gp.add(new JButton("Button 3"));
gp.add(new JButton("Button 4"));

f.add(p, BorderLayout.NORTH);
f.add(fp, BorderLayout.WEST);
f.add(bp, BorderLayout.CENTER);
f.add(gp, BorderLayout.SOUTH);
```

```
        f.setVisible(true);
    }
}
```

4. WAP in Java to add following components on the frame:

a. JLabel displaying “First Swing Application”

b. JCheckBox – to select subjects you like

c. JRadio Button- to select your favorite Subject

d. JComboBox to select your interest

e. JButton to submit data

Also use actionListener for button and display the selected data using a JOptionPane.

```
import java.awt.*;
import javax.swing.*;
public class Question4 extends JFrame implements ActionListener {

    JCheckBox c1, c2, c3;
    JRadioButton r1, r2, r3;
    JComboBox jc;
    JButton b;
    ButtonGroup g;
  public Question4() {
        JFrame f = new JFrame("Lab1:form ");
        f.setSize(500, 500);
        f.setLayout(null);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JLabel l = new JLabel("Normal Java Swing Apllication");
        l.setBounds(100, 0, 500, 20);
```

```
f.add(l);
JPanel p1 = new JPanel();
p1.setBounds(10, 20, 500, 40);
JLabel l1 = new JLabel("Choose Subjects You Like :");
l1.setBounds(10, 20, 100, 20);
p1.add(l1);
c1 = new JCheckBox("Java");
c1.setBounds(130, 20, 10, 10);
c1.addActionListener(this);
p1.add(c1);
c2 = new JCheckBox("DSA");
c2.setBounds(145, 20, 10, 10);
c2.addActionListener(this);
p1.add(c2);
c3 = new JCheckBox("Web Tech");
c3.setBounds(160, 20, 10, 10);
c3.addActionListener(this);
p1.add(c3);
f.add(p1);
JPanel p2 = new JPanel();
p2.setBounds(0, 60, 500, 40);
JLabel l2 = new JLabel("Choose Your Best Subjects:");
l2.setBounds(10, 100, 100, 20);
p2.add(l2);
r1 = new JRadioButton("java");
r1.setBounds(130, 100, 10, 10);
r1.addActionListener(this);
p2.add(r1);
```

```
    r2 = new JRadioButton("DSA");
    r2.setBounds(145, 100, 10, 10);
    r2.addActionListener(this);
    p2.add(r2);
    r3 = new JRadioButton("Web Tech");
    r3.setBounds(160, 100, 10, 10);
    r3.addActionListener(this);
    p2.add(r3);
    ButtonGroup g = new ButtonGroup();
    g.add(r1);
    g.add(r2);
    g.add(r3);
    f.add(p2);
    JLabel l3 = new JLabel("Select Your Interest:");
    l3.setBounds(50, 110, 150, 50);
    f.add(l3);
    String[] interest = {"Reading", "Coding", "Playing", "Dancing"};
    jc = new JComboBox(interest);
    jc.setBounds(190, 110, 100, 50);
    jc.addActionListener(this);
    f.add(jc);
    b = new JButton("Submit");
    b.setBounds(240, 200, 50, 20);
    b.addActionListener(this);
    f.add(b);
    f.setVisible(true);
}
@Override
```

```
public void actionPerformed(ActionEvent e) {
    if (e.getSource() == b) {

        StringBuilder LikedSubject = new StringBuilder();
        if (c1.isSelected()) {
            LikedSubject.append("Java ");
        }
        if (c2.isSelected()) {
            LikedSubject.append("DSA ");
        }
        if (c3.isSelected()) {
            LikedSubject.append("Web Tech ");
        }
        String LikedSubjects = LikedSubject.toString().trim();
        String MostLikedSubject = "Not selected";
        if (r1.isSelected()) {
            MostLikedSubject = "java";
        } else if (r2.isSelected()) {
            MostLikedSubject = "DSA";
        } else if (r3.isSelected()) {
            MostLikedSubject = "Web Tech";
        }
        String Interested = jc.getSelectedItem().toString();
        JOptionPane.showMessageDialog(this,
                "Subjects liked By User: " + LikedSubjects + "\n"
                + "Subject Most Liked By User: " + MostLikedSubject + "\n"
                + "User is Interested In: " + Interested,
                "Details Submited ",
```

```
                JOptionPane.PLAIN_MESSAGE);
        }
    }

    public static void main(String[] args) {
        new Question4();

    }
}
```

Lab 2: Java Swing Basics and Event Handling

Objective :

- To get familiar with java swing and event handling.

Theory :

Event Handling in Java

Event handling in Java is a mechanism that allows you to handle user actions like mouse clicks, keyboard inputs, or other forms of user interaction in a graphical user interface (GUI). In Java, event handling is primarily used in GUI applications like those created using Swing or AWT.

Key Components of Event Handling in Java

1. Event Source: The object that generates the event. This can be a button, a text field, a window, etc.
2. Event Object: An instance of the EventObject class (or its subclasses), which encapsulates information about the event (like mouse position, key pressed, etc.). Some common event classes:
    - ActionEvent: Represents an action event (e.g., button clicks).
    - MouseEvent: Represents a mouse event.
    - KeyEvent: Represents a keyboard event.
3. Event Listener: An interface that defines a method for responding to specific events. Some commonly used listener interfaces are:
    - ActionListener: Used for handling action events like button clicks.
    - MouseListener: Used for handling mouse events.
    - KeyListener: Used for handling keyboard events.
4. Event Handling Method: A method that gets invoked automatically when an event occurs.

Steps to Implement Event Handling

5. Implement the Listener Interface: You need to implement one or more listener interfaces in your class.
6. Register the Listener: Register the listener object with the event source. This is done using methods like addActionListener(), addMouseListener(), etc.
7. Override the Listener Methods: Implement the methods defined in the listener interface to specify what actions should be performed when the event occurs.

Lab Work:

1. Write a Java Swing program to illustrate the concept of MouseListener, MouseWheelListener, WindowListener and KeyListener.

```
public class Question1 {
    public static class MouseKeyListener extends JFrame implements MouseListener {
        public MouseKeyListener() {
            this.setTitle("Mouse Event Listener");
            this.setSize(500, 500);
            this.addMouseListener(this);
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setVisible(true);
        }
        @Override
        public void mouseClicked(MouseEvent e) {
            System.out.println("Mouse clicked");
        }
        @Override
        public void mousePressed(MouseEvent e) {
            System.out.println("Mouse pressed");
        }
        @Override
        public void mouseReleased(MouseEvent e) {
            System.out.println("Mouse released");
        }
        @Override
        public void mouseEntered(MouseEvent e) {
            System.out.println("Mouse entered");
        }
        @Override
        public void mouseExited(MouseEvent e) {
            System.out.println("Mouse exited");
```

```
    }
}
public static class WindowKeyListener extends JFrame implements WindowListener {
    public WindowKeyListener() {
        this.setTitle("Window Event Listener");
        this.setSize(500, 500);
        this.addWindowListener(this);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }
    @Override
    public void windowOpened(WindowEvent e) {
        System.out.println("Window opened");
    }
    @Override
    public void windowClosing(WindowEvent e) {
        System.out.println("Window is closing");
    }
    @Override
    public void windowClosed(WindowEvent e) {
        System.out.println("Window closed");
    }
    @Override
    public void windowIconified(WindowEvent e) {
        System.out.println("Window iconified");
    }
    @Override
    public void windowDeiconified(WindowEvent e) {
```

```
        System.out.println("Window deiconified");
    }
    @Override
    public void windowActivated(WindowEvent e) {
        System.out.println("Window activated");
    }
    @Override
    public void windowDeactivated(WindowEvent e) {
        System.out.println("Window deactivated");
    }
}
public static class KeyEventListener extends JFrame implements KeyListener {
    public KeyEventListener() {
        this.setTitle("Key Event Listener");
        this.setSize(500, 500);
        this.addKeyListener(this);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
    }
    @Override
    public void keyTyped(KeyEvent e) {
        System.out.println("Key typed");
    }
    @Override
    public void keyPressed(KeyEvent e) {
        System.out.println("Key pressed");
    }
    @Override
```

```
    public void keyReleased(KeyEvent e) {
        System.out.println("Key released");
    }
  }
  public static void main(String[] args) {
      new MouseKeyListener();
    new WindowKeyListener();
    new KeyEventListener();
  }
}
```

2. Create a frame having menu as below. Also give a message to user using JOptionPane of which menu-item user has clicked.

```
import java.awt.*;
import javax.swing.*;
public class Question2 {
  public static void main(String[] args) {
    JFrame jf = new JFrame("Menu Example ");
    jf.setSize(500, 500);
    jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    jf.setLayout(null);
    JMenuBar m = new JMenuBar();
    m.setBounds(0, 0, 500, 20);
    JMenu m1 = new JMenu("file");
    JMenu m2 = new JMenu("tools");
    JMenuItem jm1 = new JMenuItem("open");
    JMenuItem jm2 = new JMenuItem("save");
    JMenuItem jm3 = new JMenuItem("exit");
    JMenu jm4 = new JMenu("load");
```

```
        JMenuItem jm11 = new JMenuItem("load from site");
        JMenuItem jm12 = new JMenuItem("load from device");
        m1.add(jm1);
        m1.add(jm2);
        m1.add(jm3);
        m1.add(jm4);
        jm4.add(jm11);
        jm4.add(jm12);
        m.add(m1);
        m.add(m2);
        jf.setJMenuBar(m);
        jf.setResizable(false);
        jf.setVisible(true);
        jm1.addActionListener(e -> JOptionPane.showMessageDialog(jf, "You clicked 'Open'"));
        jm2.addActionListener(e -> JOptionPane.showMessageDialog(jf, "You clicked 'Save'"));
        jm3.addActionListener(e -> {
            JOptionPane.showMessageDialog(jf, "You clicked 'Exit'");
            System.exit(0);
        });
        jm11.addActionListener(e -> JOptionPane.showMessageDialog(jf, "You clicked 'Load from
Site'"));
        jm12.addActionListener(e -> JOptionPane.showMessageDialog(jf, "You clicked 'Load from
Device'"));
    }
}
```

3. Write a Java Program to create a window where user can draw anything by dragging mouse on it.

```
import java.awt.Color;
```

```
import java.awt.Graphics;
import java.awt.event.*;
import java.awt.event.MouseEvent;
import javax.swing.*;
public class Question3 extends MouseMotionAdapter{
    JFrame f;
   public Question3(){
    f= new JFrame("Draw Anything by Dragging Mouse");
    f.setSize(1000, 1000);
    f.setLayout(null);
    f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    f.addMouseMotionListener(this);
    f.setVisible(true);
   }
   @Override
   public void mouseDragged(MouseEvent e){
       Graphics g=f.getGraphics();
       g.setColor(Color.red);
       g.fillOval(e.getX(),e.getY(),20,20);
   }
    public static void main(String[] args) {
       Question3 ax = new Question3();
    }
}
```

4. Write a Java program in Java to generate a frame as below: Application above should perform appropriate actions as per the button

Click.

```
public class Question4 implements ActionListener {
```

```
    private JTextArea area;
    private String currentInput = "";
    private String operator = "";
    private double operand1 = 0;
    private boolean isOperatorClicked = false;
    public static void main(String[] args) {
        new Question4();
    }
    public Question4() {
        JFrame j = new JFrame("Calculator");
        j.setLayout(null);
        area = new JTextArea();
        area.setBounds(0, 0, 550, 100);
        area.setEditable(false);
        j.add(area);
        JPanel jp1 = new JPanel();
        jp1.setBounds(0, 100, 550, 420);
        String[] buttonLabels = {"7", "8", "9", "C", "4", "5", "6", "+", "3", "2", "1", "*", "0", "%", "/",
"="};
        JButton[] buttons = new JButton[buttonLabels.length];
        jp1.setLayout(new GridLayout(4, 4));
        for (int i = 0; i < buttonLabels.length; i++) {
            buttons[i] = new JButton(buttonLabels[i]);
            buttons[i].addActionListener(this);
            jp1.add(buttons[i]);
        }
        j.add(jp1);
        j.setSize(550, 550);
```

```
    j.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    j.setVisible(true);
}
@Override
public void actionPerformed(ActionEvent e) {
    JButton source = (JButton) e.getSource();
    String command = source.getText();
    switch (command) {
        case "C":
            currentInput = "";
            operator = "";
            operand1 = 0;
            isOperatorClicked = false;
            area.setText("");
            break;
        case "=":
            if (!currentInput.isEmpty()) {
                double result = calculate(Double.parseDouble(currentInput));
                area.setText(String.valueOf(result));
                currentInput = String.valueOf(result);
                operator = "";
            }
            break;
        case "+":
        case "-":
        case "*":
        case "/":
        case "%":
```

```
            if (!currentInput.isEmpty()) {
                operand1 = Double.parseDouble(currentInput);
                operator = command;
                currentInput = "";
                isOperatorClicked = true;
            }
            break;
        default:
            if (isOperatorClicked) {
                currentInput = command;
                isOperatorClicked = false;
            } else {
                currentInput += command;
            }
            area.setText(currentInput);
            break;
    }
}
private double calculate(double operand2) {
    switch (operator) {
        case "+":
            return operand1 + operand2;
        case "-":
            return operand1 - operand2;
        case "*":
            return operand1 * operand2;
        case "/":
            return operand1 / operand2;
```

```
        case "%":
            return operand1 % operand2;
        default:
            return operand2;
    }
}}
```

Lab 3: SQL Basics and JDBC

Objective :

- To get familiar with SQL basics and JDBC(java database connectivity).

Theory :

SQL Basics

SQL (Structured Query Language) is used to manage and interact with relational databases. It consists of several key components: DDL (Data Definition Language), which includes commands like CREATE, ALTER, and DROP to manage database structures; DML (Data Manipulation Language), such as INSERT, UPDATE, and DELETE to modify data; DQL (Data Query Language), primarily SELECT, to query data; DCL (Data Control Language) like GRANT and REVOKE to manage permissions; and TCL (Transaction Control Language), including COMMIT and ROLLBACK, to manage transactions. SQL is essential for querying, updating, and managing relational databases.

JDBC

JDBC (Java Database Connectivity) is an API in Java that enables communication between a Java application and a database using SQL. To interact with a database using JDBC, you first load the JDBC driver, establish a connection using DriverManager, create a Statement or PreparedStatement to execute SQL queries, and retrieve results using a ResultSet. After executing queries like SELECT, INSERT, UPDATE, or DELETE, always close the connection to free resources. JDBC supports both basic SQL execution and more advanced database operations like transaction management.

Lab Work:

1. Write a Java program to connect with MySQL database.

2. Write a Java program to create a database whose name provided by user from console..

3. Inside the database created above, create a table named employee having following attributes using Java code.

a. RollNo(PRIMARY KEY)

b. Firstname

c. Lastname

d. Address

e. Email (UNIQUE)

f. DateOfBirth

4. Write a Java Program to insert record of students in the student table created above from console where user can insert as much data as s/he wants.

Hint: Ask a question want to insert more data?(y/n) and insert accordingly

5. Write a Java program to display all the records of student who live in Kathmandu.

6. Write a Java Program to update the name of 5th record you have in your student table to Ram Sharma.

7. Write a Java Program to delete the record of last student.

```
public class Lab3 {
    public static void main(String[] args) {
        String DBName = JOptionPane.showInputDialog("Enter The Name Of Database You want to Create:");
        try {
            Connection conn = DriverManager.getConnection("Jdbc:mysql://localhost/", "root", "");
            String query = "CREATE DATABASE " + DBName;
            PreparedStatement pst = conn.prepareStatement(query);
            pst.executeUpdate();
            System.out.println("Database created successfully!");
```

```
Connection con = DriverManager.getConnection("Jdbc:mysql://localhost/" + DBName, "root", "");

String tquery = "CREATE TABLE employee( RollNo INT PRIMARY KEY, Firstname VARCHAR(50), Lastname VARCHAR(50), Address VARCHAR(100), Email VARCHAR(100) UNIQUE, DateOfBirth DATE)";

PreparedStatement pst1 = con.prepareStatement(tquery);

pst1.executeUpdate();

System.out.println("Table created successfully!");

String iquery = "INSERT INTO employee (RollNo, Firstname, Lastname, Address, Email, DateOfBirth) VALUES (?, ?, ?, ?, ?, ?)";

PreparedStatement pst2 = con.prepareStatement(iquery);

Scanner sc = new Scanner(System.in);

String continueInput;

do {

   System.out.print("Enter RollNo: ");

   int rollNo = sc.nextInt();

   sc.nextLine();

   System.out.print("Enter Firstname: ");

   String firstname = sc.nextLine();

   System.out.print("Enter Lastname: ");

   String lastname = sc.nextLine();

   System.out.print("Enter Address: ");

   String address = sc.nextLine();

   System.out.print("Enter Email: ");

   String email = sc.nextLine();

   System.out.print("Enter DateOfBirth (yyyy-mm-dd): ");

   String dob = sc.nextLine();

   pst2.setInt(1, rollNo);

   pst2.setString(2, firstname);
```

```
    pst2.setString(3, lastname);
    pst2.setString(4, address);
    pst2.setString(5, email);
    pst2.setDate(6, java.sql.Date.valueOf(dob));
    pst2.executeUpdate();
    System.out.print("Want to insert more data? (y/n): ");
    continueInput = sc.nextLine();
} while (continueInput.equalsIgnoreCase("y"));
System.out.println("Data inserted successfully!");
String squery = "SELECT * FROM employee WHERE Address = ?";
PreparedStatement pst3 = con.prepareStatement(squery);
pst3.setString(1, "Kathmandu");
ResultSet rs = pst3.executeQuery();
while (rs.next()) {
    System.out.println("RollNo: " + rs.getInt("RollNo"));
    System.out.println("Firstname: " + rs.getString("Firstname"));
    System.out.println("Lastname: " + rs.getString("Lastname"));
    System.out.println("Address: " + rs.getString("Address"));
    System.out.println("Email: " + rs.getString("Email"));
    System.out.println("DateOfBirth: " + rs.getDate("DateOfBirth"));
}
System.out.println("End of data.");
String uquery = "SELECT RollNo FROM employee ORDER BY RollNo LIMIT 4, 1";
PreparedStatement pst4 = con.prepareStatement(uquery);
ResultSet rs1 = pst4.executeQuery();
if (rs1.next()) {
    int rollNo = rs1.getInt("RollNo");
    System.out.println("rollno is " + rollNo);
```

```
            String updatequery = "UPDATE employee SET Firstname = 'Ram', Lastname = 'Sharma' WHERE RollNo = ?";
            PreparedStatement pstmt = con.prepareStatement(updatequery);
            pstmt.setInt(1, rollNo);
            pstmt.executeUpdate();
            System.out.println("Record updated successfully!");
        } else {
            System.out.println("5th record does not exist.");
        }
        String dquery = "SELECT RollNo FROM employee ORDER BY RollNo DESC LIMIT 1";
        PreparedStatement pst5 = con.prepareStatement(dquery);
        ResultSet rs2 = pst5.executeQuery();
        if (rs2.next()) {
            int rollNo = rs2.getInt("RollNo");
            String deletequery = "DELETE FROM employee WHERE RollNo = ?";
            PreparedStatement pstmt = con.prepareStatement(deletequery);
            pstmt.setInt(1, rollNo);
            pstmt.executeUpdate();
            System.out.println("Last record deleted successfully!");
        } else {
            System.out.println("No records to delete.");
        }
    } catch (Exception e) {
        System.out.println("SQL Error: " + e.getMessage());
    }
  }
}
```

Lab 4: JDBC and Swing

1. Write a Java program to illustrate the concept of Scrollable ResultSet.

```
public class Question1 {
   public Question1() {
      try {
         Connection conn = DriverManager.getConnection("Jdbc:mysql://localhost/bca", "root", "");
String query = "Select * from student";
PreparedStatement pst = conn.prepareStatement(query, ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
         ResultSet rs = pst.executeQuery();
         Scanner sc = new Scanner(System.in);
         int row = -1;
         while (row != 0) {
            System.out.println("Enter a row to read:");
            row = sc.nextInt();
            if (rs.absolute(row)) {
System.out.println("Name:" + rs.getString("name") + "  phone number:" + rs.getString("phone number"));
            } else {
               System.out.println("There is no data at row " + row);
            }
         }

      } catch (Exception e) {
         System.out.println("Error" + e.getMessage());
      }
   }
```

```
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection("Jdbc:mysql://localhost/bca", "root", "");
            String query = "Select * from student";
            PreparedStatement pst = conn.prepareStatement(query, ResultSet.TYPE_SCROLL_INSENSITIVE, ResultSet.CONCUR_READ_ONLY);
            ResultSet rs = pst.executeQuery();
            System.out.println("First Data is:");
            rs.first();
            System.out.println("Name:" + rs.getString("name") + "  phone number:" + rs.getString("phone number"));
            System.out.println("Relative data is:");
            rs.relative(3);
  System.out.println("Name:" + rs.getString("name") + "  phone number:" + rs.getString("phone number"));
            System.out.println("Previous Data is:");
            rs.previous();
System.out.println("Name:" + rs.getString("name") + "  phone number:" + rs.getString("phone number"));
            System.out.println("Last Data is:");
            rs.last();
   System.out.println("Name:" + rs.getString("name") + "  phone number:" + rs.getString("phone number"));
            System.out.println("Relative Data is:");
            rs.relative(-2);
 System.out.println("Name:" + rs.getString("name") + "  phone number:" + rs.getString("phone number"));
            System.out.println("Absolute Data is:");
            rs.absolute(-1);
```

```
            System.out.println("Name:" + rs.getString("name") + "      phone number:" + rs.getString("phone number"));


        } catch (Exception e) {
            System.out.println("Error" + e.getMessage());
        }
        new Question1();
    }
}
```

2. Write a Java program to illustrate the concept of Updatable ResultSet.

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Scanner;
public class Question2 {
    public static void main(String[] args) {
        try {
            Connection conn = DriverManager.getConnection("Jdbc:mysql://localhost/bca", "root", "");
            String query = "Select * from student";
            PreparedStatement pst = conn.prepareStatement(query, ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
            ResultSet rs = pst.executeQuery();
            Scanner sc = new Scanner(System.in);
            int row = -1;
```

```
        while (row != 0) {
            System.out.println("Enter a row to update:");
            row = sc.nextInt();
            if (rs.absolute(row)) {
                rs.updateString("name", "Milan Acharya");
                rs.updateRow();
                System.out.println("Name:" + rs.getString("name") + "    phone number:" + rs.getString("phone number"));
            } else {
                System.out.println("There is no data at row " + row);
            }
        }

    } catch (Exception e) {
        System.out.println("Error" + e.getMessage());
    }
  }
}
```

3. Write a Java program to illustrate the concept of RowSet.

```
import java.sql.*;
import javax.sql.rowset.*;
public class Question3 {
  public static void main(String[] args) {
    try {
      JdbcRowSet rowset = RowSetProvider.newFactory().createJdbcRowSet();
      rowset.setUrl("Jdbc:mysql://localhost/bca");
      rowset.setUsername("root");
```

```
            rowset.setPassword("");
            rowset.setCommand("Select * from student");


            rowset.execute();
            while (rowset.next()) {
               System.out.println("Name:" + rowset.getString("name") + "    phone number:" + rowset.getString("phone number"));
            }


        } catch (SQLException e) {
            System.out.println("error:" + e.getMessage());
        }
    }
}
```

4. Write a program using Java that has the JDBC connectivity and must be able to perform basic CRUD operations as shown below:

```
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.sql.*;


public class Question4 extends JFrame implements ActionListener {
    JRadioButton r1, r2;
    JComboBox<String> box;
    JButton submit, update, select_for_update, Delete;
    JTextField t1, t2, t3, t4;
```

```
    ButtonGroup b;
    JTable table;
    DefaultTableModel model;
    Connection conn;
    int selectedUserId = -1;
    public Question4() {
        JFrame f = new JFrame("Form Sample");
        f.setSize(1000, 500);
        f.setLayout(null);
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        try {
            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/BCA2077", "root",
"");
        } catch (SQLException e) {
            e.printStackTrace();
        }
        JLabel heading = new JLabel("Form Handling ");
        heading.setBounds(400, 10, 200, 40);
        f.add(heading);

        JLabel l1 = new JLabel("First Name:");
        l1.setBounds(20, 65, 100, 20);
        f.add(l1);
        t1 = new JTextField();
        t1.setBounds(120, 65, 200, 20);
        f.add(t1);
        JLabel l2 = new JLabel("Last Name:");
        l2.setBounds(20, 125, 100, 20);
```

```
f.add(l2);
t2 = new JTextField();
t2.setBounds(120, 125, 200, 20);
f.add(t2);
JLabel l3 = new JLabel("Choose Gender:");
l3.setBounds(20, 195, 100, 20);
f.add(l3);
r1 = new JRadioButton("Male");
r1.setBounds(120, 195, 80, 20);
r1.addActionListener(this);
r2 = new JRadioButton("Female");
r2.setBounds(200, 195, 80, 20);
r2.addActionListener(this);
b = new ButtonGroup();
b.add(r1);
b.add(r2);
f.add(r1);
f.add(r2);
JLabel l4 = new JLabel("Address:");
l4.setBounds(20, 225, 100, 20);
f.add(l4);
t3 = new JTextField();
t3.setBounds(120, 225, 200, 20);
f.add(t3);
JLabel l5 = new JLabel("Email:");
l5.setBounds(20, 275, 100, 20);
f.add(l5);
t4 = new JTextField();
```

```
t4.setBounds(120, 275, 200, 20);
f.add(t4);
JLabel l6 = new JLabel("Choose Blood Group:");
l6.setBounds(20, 350, 150, 20);
f.add(l6);
String[] blood = {"Select one", "A+", "B+", "O+", "AB+", "O-", "AB-", "A-", "B-"};
box = new JComboBox<>(blood);
box.setBounds(170, 350, 150, 20);
f.add(box);
submit = new JButton("Submit");
submit.setBounds(150, 420, 80, 20);
submit.addActionListener(this);
f.add(submit);
update = new JButton("Update");
update.setBounds(400, 420, 80, 20);
update.addActionListener(this);
f.add(update);
select_for_update = new JButton("Select For Update");
select_for_update.setBounds(600, 420, 120, 20);
select_for_update.addActionListener(this);
f.add(select_for_update);
Delete = new JButton("Delete");
Delete.setBounds(800, 420, 80, 20);
Delete.addActionListener(this);
f.add(Delete);
model = new DefaultTableModel();
model.setColumnIdentifiers(new Object[]{"ID", "First Name", "Last Name", "Gender", "Address", "Email", "Blood Group"});
```

```
        table = new JTable(model);
        JScrollPane pane = new JScrollPane(table);
        pane.setBounds(350, 50, 630, 350);
        f.add(pane);
        loadTableData();
        f.setVisible(true);
    }
    public static void main(String[] args) {
        new Question4();
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == submit) {
        try {
            String gender = r1.isSelected() ? "Male" : "Female";
            String iquery = "INSERT INTO users(first_name, last_name, gender, address, email, blood_group) VALUES (?, ?, ?, ?, ?, ?)";
            PreparedStatement pst = conn.prepareStatement(iquery);
            pst.setString(1, t1.getText());
            pst.setString(2, t2.getText());
            pst.setString(3, gender);
            pst.setString(4, t3.getText());
            pst.setString(5, t4.getText());
            pst.setString(6, (String) box.getSelectedItem());
            pst.executeUpdate();
            loadTableData();
            clearForm();
            } catch (SQLException ex) {
```

```
            ex.printStackTrace();
        }
        } else if (e.getSource() == update) {
            if (selectedUserId != -1) {
            try {
                String gender = r1.isSelected() ? "Male" : "Female";
                String uquery = "UPDATE users SET first_name = ?, last_name = ?, gender = ?, address
= ?, email = ?, blood_group = ? WHERE id = ?";
                PreparedStatement pst = conn.prepareStatement(uquery);
                pst.setString(1, t1.getText());
                pst.setString(2, t2.getText());
                pst.setString(3, gender);
                pst.setString(4, t3.getText());
                pst.setString(5, t4.getText());
                pst.setString(6, (String) box.getSelectedItem());
                pst.setInt(7, selectedUserId);
                pst.executeUpdate();
                loadTableData();
                clearForm();
                selectedUserId = -1;
            } catch (SQLException ex) {
                ex.printStackTrace();
            }
        } else {
            JOptionPane.showMessageDialog(null, "Please select a user to update");
        }
        } else if (e.getSource() == select_for_update) {
            int row = table.getSelectedRow();
```

```
if (row >= 0) {
    selectedUserId = (int) model.getValueAt(row, 0);
    t1.setText(model.getValueAt(row, 1).toString());
    t2.setText(model.getValueAt(row, 2).toString());
    String gender = model.getValueAt(row, 3).toString();
    if (gender.equals("Male")) {
        r1.setSelected(true);
    } else {
        r2.setSelected(true);
    }
    t3.setText(model.getValueAt(row, 4).toString());
    t4.setText(model.getValueAt(row, 5).toString());
    box.setSelectedItem(model.getValueAt(row, 6).toString());
} else {
    JOptionPane.showMessageDialog(null, "Please select a user from the table");
}
} else if (e.getSource() == Delete) {
     int row = table.getSelectedRow();
if (row >= 0) {
    int userId = (int) model.getValueAt(row, 0);
    try {
        String dquery = "DELETE FROM users WHERE id = ?";
        PreparedStatement pst = conn.prepareStatement(dquery);
        pst.setInt(1, userId);
        pst.executeUpdate();
        loadTableData();
    } catch (SQLException ex) {
        ex.printStackTrace();
```

```
        }
    } else {
        JOptionPane.showMessageDialog(null, "Please select a user to delete");
    }
    }
}
private void loadTableData() {
    try {
        model.setRowCount(0); // Clear previous data
        String squery = "SELECT * FROM users";
        PreparedStatement pst = conn.prepareStatement(squery);
        ResultSet rs = pst.executeQuery();
        while (rs.next()) {
            model.addRow(new Object[]{
                rs.getInt("id"),
                rs.getString("first_name"),
                rs.getString("last_name"),
                rs.getString("gender"),
                rs.getString("address"),
                rs.getString("email"),
                rs.getString("blood_group")
            });
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}
```

```
    private void clearForm() {
        t1.setText("");
        t2.setText("");
        b.clearSelection();
        t3.setText("");
        t4.setText("");
        box.setSelectedIndex(0);
    }
}
```

Lab 5: Basics of JavaBeans

Objective :

- To get familiar with JavaBeans.

Theory :

JavaBeans:

JavaBeans are reusable software components in Java that follow a specific convention, allowing for easy encapsulation and management of properties. A JavaBean must have the following features:

1. Private Fields: All properties (variables) should be private.
2. Public Getter and Setter Methods: It should provide public methods to access and modify the private properties.
3. No-Arg Constructor: A JavaBean must have a public no-argument constructor.
4. Serializable: The bean must implement the Serializable interface to allow its objects to be saved and restored.

Introspector: Used to analyze the properties, methods, and events of a JavaBean at runtime by providing information about the bean's structure.

PropertyDescriptor: Describes a property of a JavaBean, including its getter and setter methods, allowing for dynamic access to the bean's property.

EventSetDescriptor: Represents a set of event listeners for a JavaBean, detailing how the bean fires and handles events.

MethodDescriptor: Provides information about a method of a JavaBean, including the method's name, parameters, and return type for dynamic method handling.

Lab Work

1. Write a Java program to illustrate the concepts of JavaBeans in JSP. Make use of jsp:useBean, jsp:setProperty, jsp:getProperty

#StudentBeans.java

```
package javabeans;

import java.io.Serializable;

public class StudentBeans implements Serializable{
    private int id;
    private String name;
    private String address ;
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getAddress() {
        return address;
    }
     public void setId(int id) {
        this.id = id;
    }
    public void setName(String name) {
        this.name = name;
    }
    public void setAddress(String address) {
        this.address = address;
    }
```

```
}
```

#index.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>JavaBeans Example</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <h1>Welcome to JavaBeans Example Page</h1>
      <p>Click the link below to see a practical example of JavaBeans in a JSP page.</p>
      <a href="JavabeansExample.jsp">Click Here for JavaBeans Example</a>
   </body>
</html>
```

#JavabeansExample.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>Javabeans Example</title>
   </head>
   <body>
       <h1> welcome </h1>
         <jsp:useBean id="Student" class="javabeans.StudentBeans">
            <jsp:setProperty name="Student" property = "name" value="milan"/>
```

```
            <jsp:setProperty name="Student" property = "address" value="lalitpur"/>
            <jsp:setProperty name="Student" property = "id" value="101"/>
        </jsp:useBean>
    <p> Student id is :
        <jsp:getProperty name="Student" property="id"/>
    </p>
        <p> Student Name is :
        <jsp:getProperty name="Student" property="name"/>
        </p>
        <p> Student Address is :
        <jsp:getProperty name="Student" property="address"/>
        </p>
  </body>
</html>
```

2. Write a Java program illustrating the concept of Introspector, PropertyDescriptor, EvenetSetDescriptor and MethodDescriptor.

```
public class BeanIntrospectionExample {
  public static void main(String[] args) {
    try {
      MyBean myBean = new MyBean();
      BeanInfo beanInfo = Introspector.getBeanInfo(MyBean.class);
      System.out.println("Properties:");
      PropertyDescriptor[] propertyDescriptors = beanInfo.getPropertyDescriptors();
      for (PropertyDescriptor pd : propertyDescriptors) {
        System.out.println("Property Name: " + pd.getName());
        System.out.println("Read Method: " + pd.getReadMethod());
        System.out.println("Write Method: " + pd.getWriteMethod());
        System.out.println();
```

```
            }
            System.out.println("Events:");
            EventSetDescriptor[] eventSetDescriptors = beanInfo.getEventSetDescriptors();
            for (EventSetDescriptor esd : eventSetDescriptors) {
                System.out.println("Event Set Name: " + esd.getName());
                System.out.println("Event Listener Type: " + esd.getListenerType());
                Method[] listenerMethods = esd.getListenerMethods();
                for (Method method : listenerMethods) {
                    System.out.println("Listener Method: " + method);
                }
                System.out.println();
            }
            System.out.println("Methods:");
            MethodDescriptor[] methodDescriptors = beanInfo.getMethodDescriptors();
            for (MethodDescriptor md : methodDescriptors) {
                System.out.println("Method Name: " + md.getName());
                System.out.println("Method: " + md.getMethod());
                System.out.println();
            }

        } catch (IntrospectionException e) {
            e.printStackTrace();
        }
    }
}
class MyBean {
    private String name;
    private int age;
```

```
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public void doSomething() {
        System.out.println("Doing something...");
    }
    public void addMyBeanListener(MyBeanListener listener) {
    }
    public void removeMyBeanListener(MyBeanListener listener) {
    }
}
interface MyBeanListener {
    void onEvent();
}
```

Lab 6: Servlets and JSP

Objective :

- To get familiar with java Servlet and JSP.

Theory :

Servlets:

Servlets are Java programs that run on a web server and handle client requests by generating dynamic web content. They are part of the Java EE (Enterprise Edition) and extend the capabilities of a server by processing requests, usually over HTTP. Servlets follow a request-response model, where the client sends an HTTP request (like a form submission), and the servlet processes it, often interacting with a database or performing logic, and then generates a response (usually HTML). Servlets are managed by a servlet container (e.g., Tomcat), which handles the lifecycle of servlets from initialization to destruction. Servlets are powerful, but require manually writing a lot of boilerplate code to generate HTML.

JSP

JSP (JavaServer Pages) is a technology used for developing dynamic web pages based on HTML, XML, or other document types. Unlike servlets, JSP allows embedding Java code directly into HTML, making it easier to create dynamic content. The JSP pages are compiled into servlets by the server, and thus can use all the functionality that servlets provide. JSP simplifies development by separating the presentation logic from the business logic, allowing developers to focus more on the UI while still having access to Java code for dynamic content. It also supports tag libraries (JSTL) to further reduce Java code in JSP files and enable easier maintenance. JSPs are typically used for the view layer in MVC architectures.

Lab Work

1. Write a servlet program illustrating the concept of GenericServlet class with proper illustration of the Deployment Descriptor.

#index.html

```
<!DOCTYPE html>

<html>

<head>

    <title>TODO supply a title</title>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <style>

        body { font-family: Arial, sans-serif; background-color: #f4f4f4; margin: 0; padding: 0; }

        button { background-color: #0099ff; color: white; border: none; padding: 15px 32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin: 4px 2px; cursor: pointer; border-radius: 4px; }

        button:hover { background-color: #45a049; }

        a { text-decoration: none; }

        a button { margin: 10px; }

    </style>

</head>

<body>

    <a href="question1"><button>Example of GenericServletClass</button></a><br>

    <a href="question2"><button>Example of HTTPServlet</button></a><br>

    <a href="form.html"><button>Open Registration form</button></a><br>

    <a href="cookie_session.html"><button>Cookie and Session Example</button></a><br>

    <a href="question9.jsp"><button>Question 9</button></a><br>

    <a href="question10a.html"><button>Question 10a</button></a><br>

    <a href="question10b.jsp"><button>Question 10b</button></a><br>
```

```
    <a href="question11a"><button>Question 11a</button></a><br>
    <a href="question11b"><button>Question 11b</button></a><br>
    <a href="question11c"><button>Question 11c</button></a><br>
</body>
</html>
#question1.java
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class question1 extends GenericServlet {
    private static final long serialVersionUID = 1L;

    @Override
    public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException {
        res.setContentType("text/html;charset=UTF-8");
        PrintWriter out = res.getWriter();
        out.print("""
                <html>
                <head><title>GenericServlet Example</title></head>
                <body>illustrating the concept of GenericServlet class with proper illustration of
the Deployment Descriptor.</body>
                </html>
```

```
            """);
    }
}
```

2. Write a servlet program illustrating the concept of HttpServlet class with proper illustration of the Deployment Descriptor.

#question2.java

```
public class question2 extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html><html><head><title>Servlet question2</title></head><body>");
            out.println("illustrating the concept of HTTPServlet class with proper illustration of the Deployment Descriptor.");
            out.println("</body></html>");
        }
    }
}
```

3. Write a sevlet program to illustrate form handling. Design a html page consisting of fields like Name, Address, Email, Gender (Radio Button), Maximum Qualification (Drop Down consisting values like Select a Qualification Level, SLC, +2, Bachelor, Masters, etc) and a submit button. The servlet must display the data entered by user.

#form.html

```
<!DOCTYPE html>
<html>
<head><title>Form Handling</title></head>
```

```
<body>
    <h2>User Information Form</h2>
    <form action="question3" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>

        <label for="address">Address:</label>
        <input type="text" id="address" name="address" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>

        <label>Gender:</label>
        <input type="radio" id="male" name="gender" value="Male" required>
        <label for="male">Male</label>
        <input type="radio" id="female" name="gender" value="Female">
        <label for="female">Female</label><br><br>

        <label for="qualification">Maximum Qualification:</label>
        <select id="qualification" name="qualification">
            <option value="">Select a Qualification Level</option>
            <option value="SLC">SLC</option>
            <option value="+2">+2</option>
            <option value="Bachelor">Bachelor</option>
            <option value="Masters">Masters</option>
        </select><br><br>
```

```
        <input type="submit" value="Submit">
    </form>
</body>
</html>
#question3.java
 public class question3 extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            String name = request.getParameter("name");
            String address = request.getParameter("address");
            String email = request.getParameter("email");
            String gender = request.getParameter("gender");
            String qualification = request.getParameter("qualification");
            out.println("<!DOCTYPE                                    html><html><head><title>Servlet
Question3</title></head><body>");
            out.println("<h2>Form Data Submitted</h2>");
            out.println("<p>Name: " + name + "</p>");
            out.println("<p>Address: " + address + "</p>");
            out.println("<p>Email: " + email + "</p>");
            out.println("<p>Gender: " + gender + "</p>");
            out.println("<p>Qualification: " + qualification + "</p>");
            try {
                Class.forName("com.mysql.cj.jdbc.Driver");
try (Connection conn = DriverManager.getConnection("jdbc:mysql://localhost/user_info", "root",
"")) {
```

```
String createTableQuery = "CREATE TABLE IF NOT EXISTS user_details (id INT AUTO_INCREMENT PRIMARY KEY, name VARCHAR(250), address VARCHAR(250), email VARCHAR(250), gender VARCHAR(10), qualification VARCHAR(250))";
        try (PreparedStatement pst = conn.prepareStatement(createTableQuery)) {
            pst.executeUpdate();
        }
        String insertQuery = "INSERT INTO user_details (name, address, email, gender, qualification) VALUES (?, ?, ?, ?, ?)";
        try (PreparedStatement pst1 = conn.prepareStatement(insertQuery)) {
            pst1.setString(1, name);
            pst1.setString(2, address);
            pst1.setString(3, email);
            pst1.setString(4, gender);
            pst1.setString(5, qualification);
            pst1.executeUpdate();
        }
        String selectQuery = "SELECT * FROM user_details";
        try (PreparedStatement pst2 = conn.prepareStatement(selectQuery); ResultSet rs = pst2.executeQuery()) {
            out.println("<h2>Data from Database</h2>");
            out.println("<table border='1'><tr><th>ID</th><th>Name</th><th>Address</th><th>Email</th><th>Gender</th><th>Qualification</th></tr>");
            while (rs.next()) {
                out.println("<tr><td>" + rs.getInt("id") + "</td><td>" + rs.getString("name") + "</td><td>" + rs.getString("address") + "</td><td>" + rs.getString("email") + "</td><td>" + rs.getString("gender") + "</td><td>" + rs.getString("qualification") + "</td></tr>");
            }
            out.println("</table>");
        }
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        out.println("</body></html>");
    }
}
}
```

4. Write a servlet program to illustrate the concept of:

a. Cookie

#cookieexample.java

```
public class cookieexample extends HttpServlet {
    private static final long serialVersionUID = 1L;
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html><html><head><title>Cookie Handling</title></head><body>");
            String cookieValue = request.getParameter("cname");
            if (cookieValue != null && !cookieValue.isEmpty()) {
                Cookie cookie = new Cookie("mycookie", cookieValue);
                response.addCookie(cookie);
                out.println("Cookie set successfully!<br>");
            } else {
                out.println("No cookie value provided.<br>");
            }
```

```
        Cookie[] cookies = request.getCookies();
        if (cookies != null) {
            out.println("<h3>Cookies:</h3>");
            for (Cookie c : cookies) {
                out.println("Cookie Name: " + c.getName() + " | Value: " + c.getValue() + "<br>");
            }
        } else {
            out.println("No cookies found.<br>");
        }
        out.println("</body></html>");
    }}}
```

b. Session and Session Tracking

#cookie_session.html

```
<!DOCTYPE html>
<html>
<head>
    <title>Form Handling</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <form action="question4a" method="post">
        <label>Enter the name to set session on:</label>
        <input type="text" name="name"/>
        <input type="submit" value="Submit"/>
    </form>
    <br>
    <form action="question4b" method="post">
```

```
        <label>Enter the value of cookie:</label>
        <input type="text" name="cname"/>
        <input type="submit" value="Submit"/>
    </form>
</body>
</html>
```

#question4b.java

```
public class question4a extends HttpServlet {
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html><html><head><title>Session Handling</title></head><body>");
            String sessionName = request.getParameter("name");

            if (sessionName != null && !sessionName.isEmpty()) {
                HttpSession session = request.getSession();
                session.setAttribute("username", sessionName);
                out.println("Session set successfully!<br>");
                out.println("Stored session value is: " + session.getAttribute("username") + "<br>");
            } else {
                out.println("No session name provided.<br>");
            }
            out.println("<a href=\"index.html\">Back to Home</a>");
            out.println("</body></html>");
      }}}
```

5. Insert the form data of question 3 to database. Create necessary database and table.

6. Display the data inserted in question 5 in tabular format using a Servlet.

7. Write a program to read the value of principal, interest rate and time from user and display the simple interest value using JSP.

8. Write a program to read a String value from user and its reverse using JSP.

9. Write a JSP program to retrieve data stored in question 5 and display in tabular form.

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>User Information Form</title>
</head>
<body>
    <h2>User Information Form</h2>
    <form action="" method="post">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>

        <label for="address">Address:</label>
        <input type="text" id="address" name="address" required><br><br>

        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>

        <label>Gender:</label>
        <input type="radio" id="male" name="gender" value="Male" required>
        <label for="male">Male</label>
        <input type="radio" id="female" name="gender" value="Female">
```

```
    <label for="female">Female</label><br><br>

    <label for="qualification">Maximum Qualification:</label>
    <select id="qualification" name="qualification">
        <option value="">Select a Qualification Level</option>
        <option value="SLC">SLC</option>
        <option value="+2">+2</option>
        <option value="Bachelor">Bachelor</option>
        <option value="Masters">Masters</option>
    </select><br><br>

    <input type="submit" value="Submit">
</form>

<% if ("post".equalsIgnoreCase(request.getMethod())) { %>
    <h3>Submitted User Information</h3>
    <table border="1" cellpadding="5" cellspacing="0">
        <tr>
            <th>Name</th>
            <td><%= request.getParameter("name") %></td>
        </tr>
        <tr>
            <th>Address</th>
            <td><%= request.getParameter("address") %></td>
        </tr>
        <tr>
            <th>Email</th>
```

```
            <td><%= request.getParameter("email") %></td>
        </tr>
        <tr>
            <th>Gender</th>
            <td><%= request.getParameter("gender") %></td>
        </tr>
        <tr>
            <th>Qualification</th>
            <td><%= request.getParameter("qualification") != null && !request.getParameter("qualification").isEmpty() ? request.getParameter("qualification") : "Not specified" %></td>
        </tr>
    </table>
<% } %>
</body>
</html>
```

10.Illustrate the concept of login using:

a. Servlet

```
<!DOCTYPE html>
<html>
<head>
    <title>Question 10a</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <form method="post" action="question10a">
```

```
        <label>Username:</label>
        <input type="text" name="username"/>
        <label>Password:</label>
        <input type="password" name="password"/>
        <input type="submit" value="Submit"/>
    </form>
</body>
</html>
```

#question10a.java

```
public class question10a extends HttpServlet {
    private static final String DEFINED_USERNAME = "milan";
    private static final String DEFINED_PASSWORD = "milan";
    private static final long serialVersionUID = 1L;
    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {
            out.println("<!DOCTYPE html><html><head><title>Servlet question10a</title></head><body>");
            String username = request.getParameter("username");
            String password = request.getParameter("password");
            if (username != null && !username.isEmpty() && password != null && !password.isEmpty()) {
                out.println(username.equals(DEFINED_USERNAME) && password.equals(DEFINED_PASSWORD) ?
                    "Login successful!!" : "Invalid Credentials!!");
            } else {
                out.println("Please enter both username and password.");
```

```
        }
        out.println("</body></html>");
    }   }}
```

b. JSP

#question10b.jsp

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>question 10b</title>
</head>
<body>
    <form method="post">
        <label>Username</label>
        <input type="text" name="username"/>
        <label>Password</label>
        <input type="password" name="password"/>
        <input type="submit" value="Submit"/>
    </form>
    <%
        final String DEFINED_USERNAME = "milan";
        final String DEFINED_PASSWORD = "milan";
        if ("post".equalsIgnoreCase(request.getMethod())) {
            String username = request.getParameter("username");
            String password = request.getParameter("password");
```

```
        if (username != null && !username.isEmpty() && password != null
&& !password.isEmpty()) {
            out.println(username.equals(DEFINED_USERNAME)                    &&
password.equals(DEFINED_PASSWORD) ?
                "Login successful!!" : "Invalid Credentials!!");
        } else {
            out.println("Please enter both username and password.");
        }
    }
  %>
</body>
</html>
```

11.Write a simple JSP program to illustrate:

a. Passing data from Servlet to JSP

#question11.java

```
public class question11 {
   private String name ;
   private String address;
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getAddress() {
      return address;
   }
```

```
 public void setAddress(String address) {
        this.address = address;
    }}
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>question 11</title>
</head>
<body>
    <%
        out.print("Message from servlet to JSP: ");
        out.print(request.getAttribute("message"));
    %>
</body>
</html>
public class question11a extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        String message = "Hello from Servlet!";
        request.setAttribute("message", message);
        request.getRequestDispatcher("question11a.jsp").forward(request, response);
    }}
```

b. Passing object from servlet to JSP

```
<%@page import="question11.question11"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Object from Servlet to JSP</title>
</head>
<body>
    <h2>Example of Passing Object from Servlet to JSP</h2>
    <%
        question11 q = (question11) request.getAttribute("data");
        if (q != null) {
            out.println("Name: " + q.getName() + "<br>");
            out.println("Address: " + q.getAddress());
        } else {
            out.println("No data available.");
        }
    %>
</body>
</html>
public class question11b extends HttpServlet {
    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        question11 q = new question11();
        q.setName("milan");
```

```
        q.setAddress("kathmandu");
        request.setAttribute("data", q);
        request.getRequestDispatcher("question11b.jsp").forward(request, response);
    }}
```

c. Passing list from servlet to JSP

```
<%@page import="java.util.List"%>
<%@page import="question11.question11"%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Passing ArrayList from Servlet to JSP</title>
</head>
<body>
    <h1>Passing ArrayList from Servlet to JSP</h1>
    <%
        List<question11> personList = (List<question11>) request.getAttribute("data");
        if (personList != null) {
            for (question11 q : personList) {
    %>
            <p>Name: <%= q.getName() %><br>
            Address: <%= q.getAddress() %></p>
    <%
            }
        } else {
            out.println("No data available.");
        }
```

```
    %>
</body>
</html>
public class question11c extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @Override
    protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        List<question11> personList = new ArrayList<>();
        personList.add(createPerson("milan", "kathmandu"));
        personList.add(createPerson("Sahil", "kathmandu"));
        personList.add(createPerson("Prashant", "kathmandu"));

        request.setAttribute("data", personList);
        request.getRequestDispatcher("question11c.jsp").forward(request, response);
    }
    private question11 createPerson(String name, String address) {
        question11 person = new question11();
        person.setName(name);
        person.setAddress(address);
        return person;
    }}
```

Lab 7: RMI

Objective :

- To get familiar with RMI.

Theory :

Remote Method Invocation (RMI) is a Java API that allows objects to invoke methods on remote objects, which are located on different Java Virtual Machines (JVMs). Here's a concise overview:

- Purpose: RMI enables communication between objects in different JVMs, whether they are on the same machine or on different machines across a network.
- Architecture: It consists of a client and a server. The client invokes methods on a remote object, and the server implements the methods and provides the remote object.
- Key Components:
    - Remote Interface: Defines the methods that can be called remotely.
    - Remote Object: Implements the remote interface and provides the actual method implementations.
    - RMI Registry: A service that allows remote objects to be looked up by clients.
- Steps:
    a. Define Remote Interface: Declare the methods to be called remotely.
    b. Implement Remote Object: Provide implementations of the remote methods.
    c. Register Remote Object: Bind the remote object to the RMI registry.
    d. Lookup Remote Object: Client retrieves the remote object reference from the RMI registry and invokes methods.
- Serialization: Objects sent over RMI must implement Serializable to ensure proper data transmission

Lab Work

1. Write a client server application to find the product of two numbers.

2. Write a client server application to display the reverse of a String value.

3. Write a Client Server Application in Java where the Server provides price of different laptop models in dollar. Client Program can ask for the price of a laptop model and displays it.

#Server.java

```
public class Server {
    public static void main(String[] args) {
        try {
            RMIImplements obj = new RMIImplements();
            RMIExample skeleton = (RMIExample) UnicastRemoteObject.exportObject(obj, 0);
            Registry registry = LocateRegistry.getRegistry();
            registry.bind("price", skeleton);
            System.out.println("server ready ");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

#Client.java

```
public class Client {
    public static void main(String[] args) {
        try {
            Registry registry = LocateRegistry.getRegistry();
            RMIExample stub = (RMIExample) registry.lookup("price");
            String price = stub.ProvidedPrice("Mac");
            int result = stub.product(10, 10);
```

```
            String reverseString = stub.Reverse("Milan");
            System.out.println("price =" + price);
            System.out.println("the product of two number is :" + result);
            System.out.println("the reverse string of the provide string is :" + reverseString);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
#RMIExample.java
public interface RMIExample extends Remote{
      public int product(int a , int b) throws RemoteException;
      public String Reverse (String name) throws RemoteException;
      public String ProvidedPrice(String ModelName) throws RemoteException;
}
#RMIImplements.java
public class RMIImplements implements RMIExample{
     private Map<String,String> prices ;
    public RMIImplements(){
        prices=new HashMap<>();
        prices.put("DEll","1000000");
        prices.put("HP","1200000");
        prices.put("Mac","1300000");
    }
    @Override
    public String ProvidedPrice(String ModelName) throws RemoteException {
        return prices.get(ModelName);
    }
```

```
 @Override
    public int product(int a, int b) throws RemoteException {
       return a*b;
    }
    @Override
    public String Reverse(String name) throws RemoteException {
        StringBuilder sb = new StringBuilder(name);
        sb.reverse();
        return sb.toString();
    }
}
```