**WEEK 9 – TUTORIAL ASSIGNMENT**

1. **Write a java program using swing by inheritance.**

PROGRAM:

import javax.swing.\*;

public class SwingInheritanceExample extends JFrame{//inheriting JFrame

JFrame f;

SwingInheritanceExample(){

// Set the frame title

setTitle("Swing Inheritance Example");

JButton b=new JButton("click");//create button

b.setBounds(130,100,100, 40);

add(b);//adding button on frame

setSize(400,500);

setLayout(null);

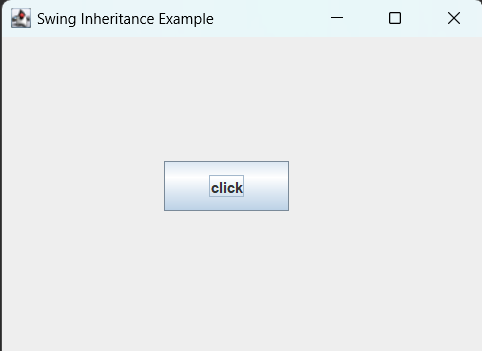
setVisible(true);

}

public static void main(String[] args) {

new SwingInheritanceExample();

}}



1. **Write a java program using swing with ActionListener.**

PROGRAM:

import javax.swing.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class SwingActionListenerExample {

    private int clickCount = 0; // Counter for button clicks

    public SwingActionListenerExample() {

        // Create a JFrame

        JFrame frame = new JFrame("Swing ActionListener Example");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(300, 150);

        // Create a JPanel to hold components

        JPanel panel = new JPanel();

        // Create a button

        JButton button = new JButton("Click Me!");

        // Create a label to display the click count

        JLabel label = new JLabel("Click Count: 0");

        // Add the button and label to the panel

        panel.add(button);

        panel.add(label);

        // Add an ActionListener to the button

        button.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Increment the click count

                clickCount++;

                // Update the label text

                label.setText("Click Count: " + clickCount);

            }

        });

        // Add the panel to the JFrame

        frame.add(panel);

        // Center the JFrame on the screen

        frame.setLocationRelativeTo(null);

        // Make the JFrame visible

        frame.setVisible(true);

    }

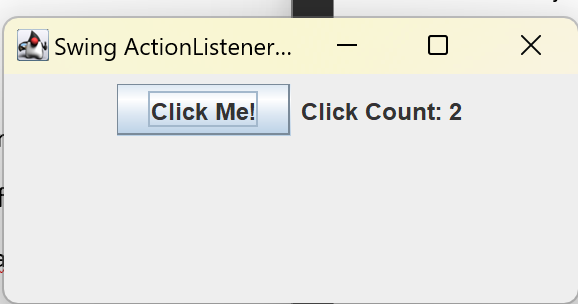
    public static void main(String[] args) {

        // Create an instance of SwingActionListenerExample

        SwingActionListenerExample example = new SwingActionListenerExample();

    }

}



**3. Using Java JMenuItem and JMenu implement application swing.**

PROGRAM:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class MenuExample {

    public static void main(String[] args) {

        JFrame frame = new JFrame("Swing Menu Example");

        frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        frame.setSize(400, 300);

        // Create a menu bar

        JMenuBar menuBar = new JMenuBar();

        // Create a File menu

        JMenu fileMenu = new JMenu("File");

        // Create File menu items

        JMenuItem newItem = new JMenuItem("New");

        JMenuItem openItem = new JMenuItem("Open");

        JMenuItem saveItem = new JMenuItem("Save");

        JMenuItem exitItem = new JMenuItem("Exit");

        // Add action listeners to menu items

        newItem.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Implement your "New" functionality here

                JOptionPane.showMessageDialog(null, "New File was selected.");

            }

        });

        openItem.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Implement your "Open" functionality here

                JOptionPane.showMessageDialog(null, "Open File was selected.");

            }

        });

        saveItem.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Implement your "Save" functionality here

                JOptionPane.showMessageDialog(null, "Save File was selected.");

            }

        });

        exitItem.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Implement your "Exit" functionality here

                System.exit(0);

            }

        });

        // Add menu items to the File menu

        fileMenu.add(newItem);

        fileMenu.add(openItem);

        fileMenu.add(saveItem);

        fileMenu.addSeparator(); // Add a separator line

        fileMenu.add(exitItem);

        // Add the File menu to the menu bar

        menuBar.add(fileMenu);

        // Set the menu bar to the frame

        frame.setJMenuBar(menuBar);

        // Center the frame on the screen

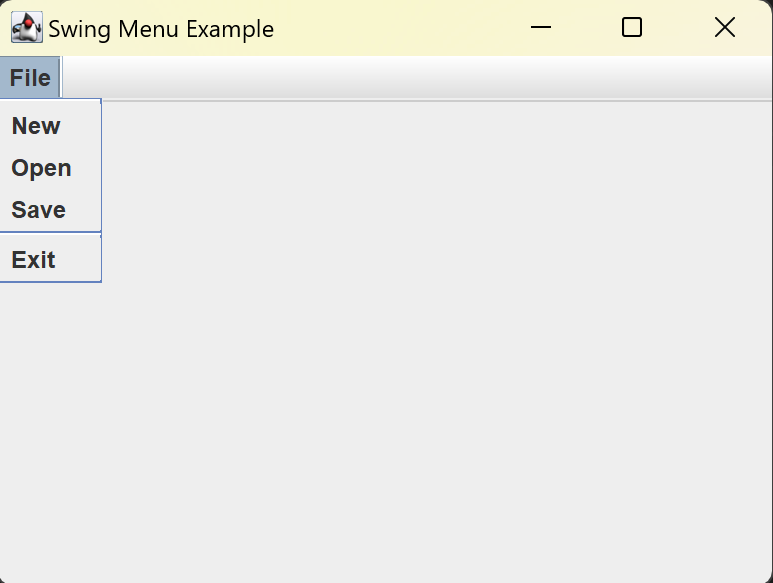
        frame.setLocationRelativeTo(null);

        // Make the frame visible

        frame.setVisible(true);

    }

}



**4. Develop a student registration form using SWING components**.

PROGRAM:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class StudentRegistrationForm extends JFrame {

    // Components

    private JTextField nameField, rollNumberField, emailField;

    private JButton submitButton;

    public StudentRegistrationForm() {

        // Set frame properties

        setTitle("Student Registration Form");

        setSize(400, 200);

        setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

        setLocationRelativeTo(null);

        // Create a panel to hold components

        JPanel panel = new JPanel();

        panel.setLayout(new GridLayout(4, 2, 10, 10)); // 4 rows, 2 columns

        // Add components to the panel

        panel.add(new JLabel("Name:"));

        nameField = new JTextField(20);

        panel.add(nameField);

        panel.add(new JLabel("Roll Number:"));

        rollNumberField = new JTextField(10);

        panel.add(rollNumberField);

        panel.add(new JLabel("Email:"));

        emailField = new JTextField(30);

        panel.add(emailField);

        submitButton = new JButton("Submit");

        panel.add(submitButton);

        // Add an ActionListener to the submit button

        submitButton.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                // Retrieve the entered data

                String name = nameField.getText();

                String rollNumber = rollNumberField.getText();

                String email = emailField.getText();

                // Display the entered data in a message dialog

                String message = "Name: " + name + "\nRoll Number: " + rollNumber + "\nEmail: " + email;

                JOptionPane.showMessageDialog(StudentRegistrationForm.this, message, "Registration Details", JOptionPane.INFORMATION\_MESSAGE);

            }

        });

        // Add the panel to the frame

        add(panel);

        // Make the frame visible

        setVisible(true);

    }

    public static void main(String[] args) {

        SwingUtilities.invokeLater(new Runnable() {

            @Override

            public void run() {

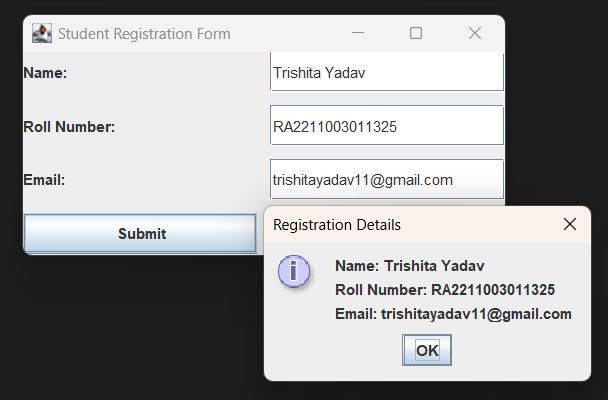
                new StudentRegistrationForm();

            }

        });

    }

}



**5. Implement Employment registration form using SWING components.**

PROGRAM:

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

public class EmploymentRegistrationForm {

public static void main(String[] args) {

// Create the main frame

JFrame frame = new JFrame("Employment Registration Form");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.setSize(400, 300);

// Create a panel to hold the form components

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(6, 2)); // 6 rows and 2 columns

// Create labels and text fields for employment information

JLabel nameLabel = new JLabel("Name:");

JTextField nameField = new JTextField(20);

JLabel emailLabel = new JLabel("Email:");

JTextField emailField = new JTextField(20);

JLabel phoneLabel = new JLabel("Phone:");

JTextField phoneField = new JTextField(15);

JLabel addressLabel = new JLabel("Address:");

JTextArea addressArea = new JTextArea(3, 20);

JScrollPane addressScrollPane = new JScrollPane(addressArea);

JLabel positionLabel = new JLabel("Position:");

JTextField positionField = new JTextField(15);

// Create a Submit button

JButton submitButton = new JButton("Submit");

// Add action listener to the Submit button

submitButton.addActionListener(new ActionListener() {

@Override

public void actionPerformed(ActionEvent e) {

// Retrieve and display the entered information

String name = nameField.getText();

String email = emailField.getText();

String phone = phoneField.getText();

String address = addressArea.getText();

String position = positionField.getText();

// Display the information in a dialog

String message = "Name: " + name + "\nEmail: " + email + "\nPhone: " + phone

+ "\nAddress: " + address + "\nPosition: " + position;

JOptionPane.showMessageDialog(frame, message, "Registration Details", JOptionPane.PLAIN\_MESSAGE);

}

});

// Add components to the panel

panel.add(nameLabel);

panel.add(nameField);

panel.add(emailLabel);

panel.add(emailField);

panel.add(phoneLabel);

panel.add(phoneField);

panel.add(addressLabel);

panel.add(addressScrollPane);

panel.add(positionLabel);

panel.add(positionField);

// Add the Submit button to the panel

panel.add(submitButton);

// Add the panel to the frame

frame.add(panel);

// Center the frame on the screen

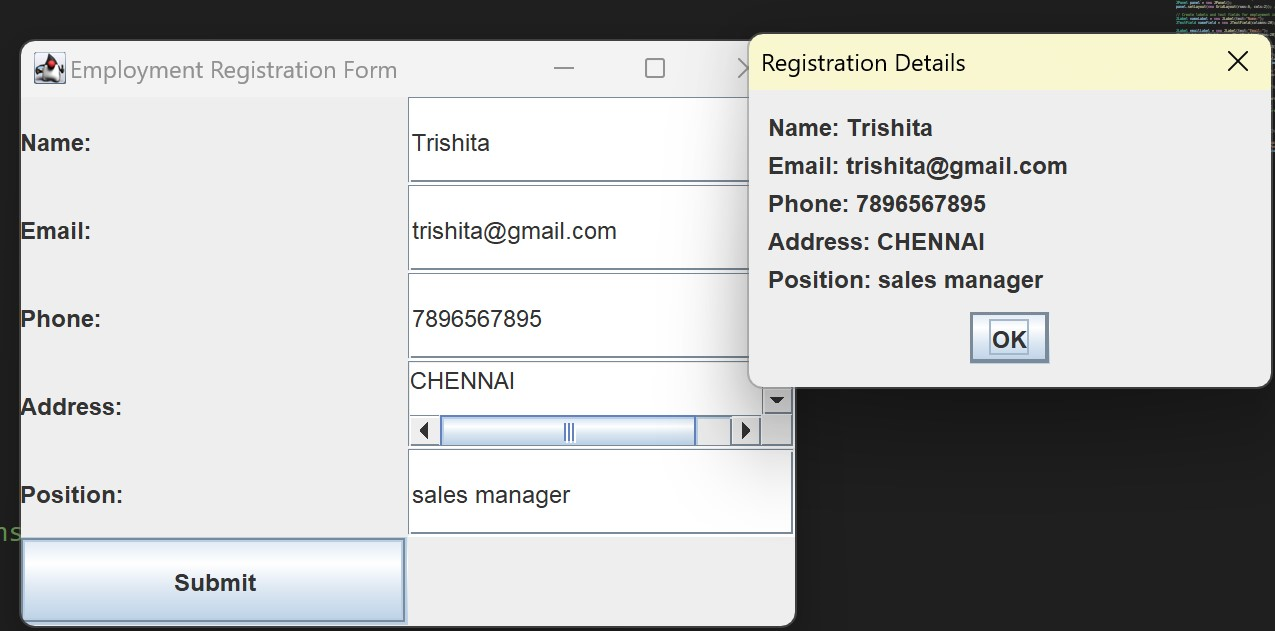
frame.setLocationRelativeTo(null);

// Make the frame visible

frame.setVisible(true);

}

}



**6. Write a java program to draw Oval, Rectangle, Line and fill the colour in it and display it on Applet**.

PROGRAM:

import java.awt.\*;

import java.applet.\*;

public class ShapeDrawingApplet extends Applet {

public void paint(Graphics g) {

// Draw an oval

g.setColor(Color.RED);

g.fillOval(50, 50, 100, 80);

// Draw a rectangle

g.setColor(Color.BLUE);

g.fillRect(200, 50, 100, 80);

// Draw a line

g.setColor(Color.GREEN);

g.drawLine(50, 200, 250, 200);

}

}

**7. Draw a chessboard in java applet.**

PROGRAM:

import java.applet.Applet;

import java.awt.Color;

import java.awt.Graphics;

public class ChessboardApplet extends Applet {

public void paint(Graphics g) {

int squareSize = 50; // Size of each square

int numRows = 8; // Number of rows on the chessboard

int numCols = 8; // Number of columns on the chessboard

// Loop through rows

for (int row = 0; row < numRows; row++) {

// Loop through columns

for (int col = 0; col < numCols; col++) {

int x = col \* squareSize;

int y = row \* squareSize;

// Alternate between white and black squares

if ((row + col) % 2 == 0) {

g.setColor(Color.WHITE);

} else {

g.setColor(Color.BLACK);

}

// Fill the square with the selected color

g.fillRect(x, y, squareSize, squareSize);

}

}

}

}

**8. Write a java program that handles all mouse events and shows the event name at the center of the window when mouse event is fired (Use Adapter classes and applet).**

PROGRAM:

import java.applet.Applet;

import java.awt.\*;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

public class MouseEventApplet extends Applet {

private String eventName;

public void init() {

addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

eventName = "Mouse Clicked";

repaint();

}

public void mousePressed(MouseEvent e) {

eventName = "Mouse Pressed";

repaint();

}

public void mouseReleased(MouseEvent e) {

eventName = "Mouse Released";

repaint();

}

public void mouseEntered(MouseEvent e) {

eventName = "Mouse Entered";

repaint();

}

public void mouseExited(MouseEvent e) {

eventName = "Mouse Exited";

repaint();

}

});

}

public void paint(Graphics g) {

// Clear the applet

g.clearRect(0, 0, getWidth(), getHeight());

// Set font and color for displaying event name

g.setFont(new Font("Arial", Font.BOLD, 18));

g.setColor(Color.BLUE);

// Center the text

FontMetrics fm = g.getFontMetrics();

int x = (getWidth() - fm.stringWidth(eventName)) / 2;

int y = (getHeight() + fm.getAscent()) / 2;

// Draw the event name

g.drawString(eventName, x, y);

}

}

**9. Implement java MVC pattern application with Student object Model, StudentView and StudentController.**

import java.util.Scanner;

class Student {

private String name;

private int rollNumber;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getRollNumber() {

return rollNumber;

}

public void setRollNumber(int rollNumber) {

this.rollNumber = rollNumber;

}

}

class StudentView {

public void printStudentDetails(String studentName, int studentRollNumber) {

System.out.println("Student Details:");

System.out.println("Name: " + studentName);

System.out.println("Roll Number: " + studentRollNumber);

}

}

class StudentController {

private Student model;

private StudentView view;

public StudentController(Student model, StudentView view) {

this.model = model;

this.view = view;

}

public void setStudentName(String name) {

model.setName(name);

}

public String getStudentName() {

return model.getName();

}

public void setStudentRollNumber(int rollNumber) {

model.setRollNumber(rollNumber);

}

public int getStudentRollNumber() {

return model.getRollNumber();

}

public void updateView() {

view.printStudentDetails(model.getName(), model.getRollNumber());

}

}

public class MVCPatternDemo {

public static void main(String[] args) {

// Create the Model, View, and Controller

Student model = new Student();

StudentView view = new StudentView();

StudentController controller = new StudentController(model, view);

// Update student data through the controller

controller.setStudentName("John");

controller.setStudentRollNumber(101);

// Display student details through the controller

controller.updateView();

// Update the model data directly

model.setName("Alice");

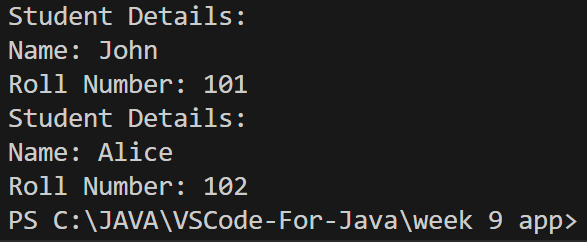
model.setRollNumber(102);

// Display student details directly

view.printStudentDetails(model.getName(), model.getRollNumber());

}

}



**10. Implement java MVC to display Employee details.**

PROGRAM:

public class Employee {

private String name;

private int employeeId;

public Employee(String name, int employeeId) {

this.name = name;

this.employeeId = employeeId;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getEmployeeId() {

return employeeId;

}

public void setEmployeeId(int employeeId) {

this.employeeId = employeeId;

}

}

public class EmployeeView {

public void printEmployeeDetails(String name, int employeeId) {

System.out.println("Employee Details:");

System.out.println("Name: " + name);

System.out.println("Employee ID: " + employeeId);

}

}

public class EmployeeController {

private Employee model;

private EmployeeView view;

public EmployeeController(Employee model, EmployeeView view) {

this.model = model;

this.view = view;

}

public void updateView() {

view.printEmployeeDetails(model.getName(), model.getEmployeeId());

}

public void setEmployeeName(String name) {

model.setName(name);

}

public void setEmployeeId(int employeeId) {

model.setEmployeeId(employeeId);

}

}

public class MVCEmployeeDemo {

public static void main(String[] args) {

// Create the Model, View, and Controller

Employee employee = new Employee("John Doe", 101);

EmployeeView view = new EmployeeView();

EmployeeController controller = new EmployeeController(employee, view);

// Display employee details

controller.updateView();

// Update employee data through the controller

controller.setEmployeeName("Alice Smith");

controller.setEmployeeId(102);

// Display updated employee details

controller.updateView();

}

}