Module 6

Ex No 6.1 Develop an Applet program to accept two numbers from the user and output the sum and difference in the respective text boxes.

Aim

To develop an Applet program to accept two numbers from the user and output the sum and difference in the respective text boxes.

Algorithm

- **Initialize Applet Components**:
1. Create two TextField components for the user to enter the two numbers.
2. Create two TextField components to display the sum and difference.
3. Create a Button to trigger the calculation.

- **Accept User Input**:
  Use TextField components to get the two numbers from the user.

- **Perform Calculation**:
1. On pressing the button, retrieve the numbers entered by the user.
2. Convert these input values (strings) to integers.
3. Calculate the sum and the difference of the two numbers.

- **Display the Results**:
  Set the sum and difference values to the respective result text boxes

.

- **Handle Events**:
  Implement an event listener for the button click to trigger the calculation.

Program

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

public class SumDifferenceApplet extends Applet implements ActionListener {
   TextField num1Field, num2Field, sumField, diffField;
   Button calculateButton;

   public void init() {
      // Labels
```

```
    Label num1Label = new Label("Enter Number 1:");
    Label num2Label = new Label("Enter Number 2:");
    Label sumLabel = new Label("Sum:");
    Label diffLabel = new Label("Difference:");

    // Text Fields
    num1Field = new TextField(10);
    num2Field = new TextField(10);
    sumField = new TextField(10);
    diffField = new TextField(10);

    sumField.setEditable(false);  // Output field (readonly)
    diffField.setEditable(false); // Output field (readonly)

    // Button
    calculateButton = new Button("Calculate");
    calculateButton.addActionListener(this);

    // Layout
    setLayout(new GridLayout(5, 2)); // 5 Rows, 2 Columns

    // Adding Components
    add(num1Label);
    add(num1Field);
    add(num2Label);
    add(num2Field);
    add(sumLabel);
    add(sumField);
    add(diffLabel);
    add(diffField);
    add(calculateButton);
}

public void actionPerformed(ActionEvent e) {
    try {
        int num1 = Integer.parseInt(num1Field.getText());
        int num2 = Integer.parseInt(num2Field.getText());

        int sum = num1 + num2;
        int diff = num1 - num2;
```

```
            sumField.setText(String.valueOf(sum));
            diffField.setText(String.valueOf(diff));
        } catch (NumberFormatException ex) {
            sumField.setText("Invalid Input");
            diffField.setText("Invalid Input");
        }
    }
}
```

Steps to Run the Applet

Save the file as SumDifferenceApplet.java

Compile the Java file
javac SumDifferenceApplet.java

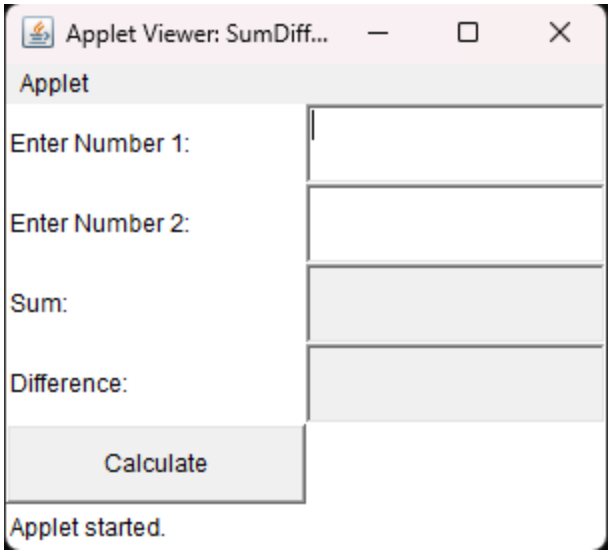
Create an HTML file (applet.html)

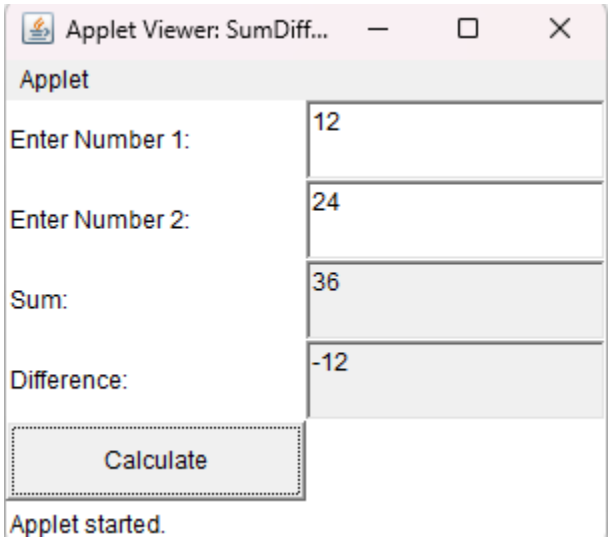
```
<html>
<body>

    <applet code="SumDifferenceApplet.class" width="300" height="200"></applet>

</body>
</html>
```

Run using Applet Viewer
appletviewer applet.html

Output

```
C:\Users\Admin\Documents\OOPTW>javac SumDifferenceApplet.java

C:\Users\Admin\Documents\OOPTW>appletviewer applet.html
Warning: Can't read AppletViewer properties file: C:\Users\Admin\.hotjava\properties Using defaults.

C:\Users\Admin\Documents\OOPTW>
```





Result

Thus the program to develop an Applet program to accept two numbers from the user and output the sum and difference in the respective text boxes is completed successfully and the output is verified.

Ex No 6.2 Write a program that identifies key-up key-down event user entering text in a Applet

Aim

To write a program that identifies key-up key-down event user entering text in a Applet

Algorithm

1. Initialize Applet Components:
    - Create a TextField or TextArea for user text input.
    - Create Label or TextField components to display the messages for keyPressed (key-down) and keyReleased (key-up) events.
2. Add KeyListener:
    - Attach a KeyListener to the text input component (e.g., TextField).
    - Implement the methods keyPressed, keyReleased, and keyTyped of the KeyListener interface.
3. Handle Key Events:
    - keyPressed(KeyEvent e): This method will be triggered when a key is pressed down (key-down event).
    - keyReleased(KeyEvent e): This method will be triggered when a key is released (key-up event).
    - keyTyped(KeyEvent e): This method can be used to identify the key typed (if necessary).
4. Display Event Information:
    - Use Label or TextField components to display the key information (such as the key code or the key character) for both keyPressed and keyReleased.

Program

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

public class KeyEventApplet extends Applet implements KeyListener {
   String message = "";
```

```
    public void init() {
        addKeyListener(this);  // Register key listener
        setFocusable(true);    // Ensure the applet can receive keyboard input
    }

    public void keyPressed(KeyEvent e) {
        message = "Key Down: " + e.getKeyChar();
        repaint();
    }

    public void keyReleased(KeyEvent e) {
        message = "Key Up: " + e.getKeyChar();
        repaint();
    }

    public void keyTyped(KeyEvent e) {
        // Key typed event (for character keys)
        message = "Key Typed: " + e.getKeyChar();
        repaint();
    }

    public void paint(Graphics g) {
        g.drawString(message, 20, 50);
    }
}
```

Output

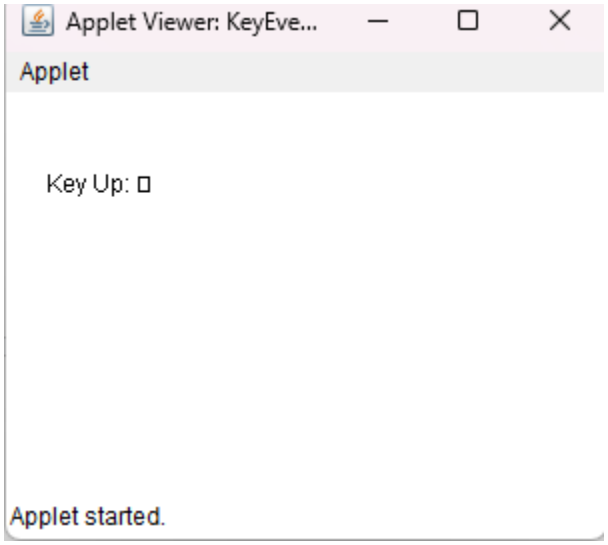
Save the file as KeyEventApplet.java

Compile the Java file
javac KeyEventApplet.java
Create an HTML file (applet.html)

```
<html>
<body>
    <applet code="KeyEventApplet.class" width="300" height="200"></applet>
</body>
</html>
```

Run using Applet Viewer
appletviewer applet.html

```
C:\Users\Admin\Documents\OOPTW>javac KeyEventApplet.java

C:\Users\Admin\Documents\OOPTW>appletviewer appletkeyevent.html
```



Result

Thus to write a program that identifies key-up key-down event user entering text in a Applet is completed successfully and output is verified.

Ex No 6.3 Write a Java program to design student registration form using Swing controls.

Aim

To write a Java program to design student registration forms using Swing controls.

Algorithm

Initialize the Swing Component

1. Create a JFrame for the main window.
2. Add JLabels for form fields (Name, Age, Gender, Course, etc.).
3. Add JTextFields for text input.
4. Add JRadioButtons for gender selection (Group them using ButtonGroup).
5. Add JComboBox for selecting a course.
6. Add JCheckBox for agreement or terms selection.
7. Add JButton for form submission and reset.

Set Layout and Position Components
Use GridLayout or GridBagLayout for proper alignment. Place labels and corresponding input fields properly.

Add Action Listeners
Implement an ActionListener for the submit button to fetch input values. Implement an ActionListener for the reset button to clear fields.

Perform Data Validation
Check if all required fields are filled.Validate the age field to accept only numbers.Ensure gender is selected.

Display Confirmation Message
Show a message dialog on successful registration.

Run the Application
Create the frame object in main and set it to be visible.

Program

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

public class StudentRegistrationForm extends JFrame implements ActionListener {
   JTextField nameField, ageField;
   JRadioButton male, female;
   JComboBox<String> departmentBox;
   JCheckBox javaCheck, pythonCheck, cppCheck;
   JButton submitButton, resetButton;
```

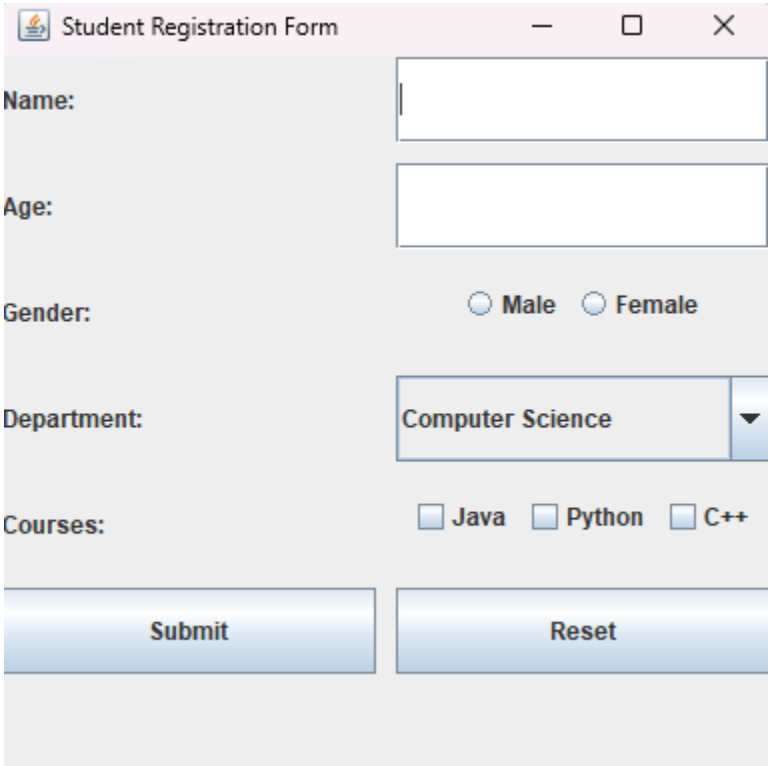
```
public StudentRegistrationForm() {
    setTitle("Student Registration Form");
    setSize(400, 400);
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    setLayout(new GridLayout(7, 2, 10, 10)); // 7 rows, 2 columns

    // Name Field
    add(new JLabel("Name:"));
    nameField = new JTextField();
    add(nameField);

    // Age Field
    add(new JLabel("Age:"));
    ageField = new JTextField();
    add(ageField);

    // Gender Selection
    add(new JLabel("Gender:"));
    JPanel genderPanel = new JPanel();
    male = new JRadioButton("Male");
    female = new JRadioButton("Female");
    ButtonGroup genderGroup = new ButtonGroup();
    genderGroup.add(male);
    genderGroup.add(female);
    genderPanel.add(male);
    genderPanel.add(female);
    add(genderPanel);

    // Department Selection
    add(new JLabel("Department:"));
    String[] departments = { "Computer Science", "Electronics", "Mechanical", "Civil" };
    departmentBox = new JComboBox<>(departments);
    add(departmentBox);

    // Course Selection
    add(new JLabel("Courses:"));
    JPanel coursePanel = new JPanel();
    javaCheck = new JCheckBox("Java");
    pythonCheck = new JCheckBox("Python");
    cppCheck = new JCheckBox("C++");
    coursePanel.add(javaCheck);
    coursePanel.add(pythonCheck);
    coursePanel.add(cppCheck);
```

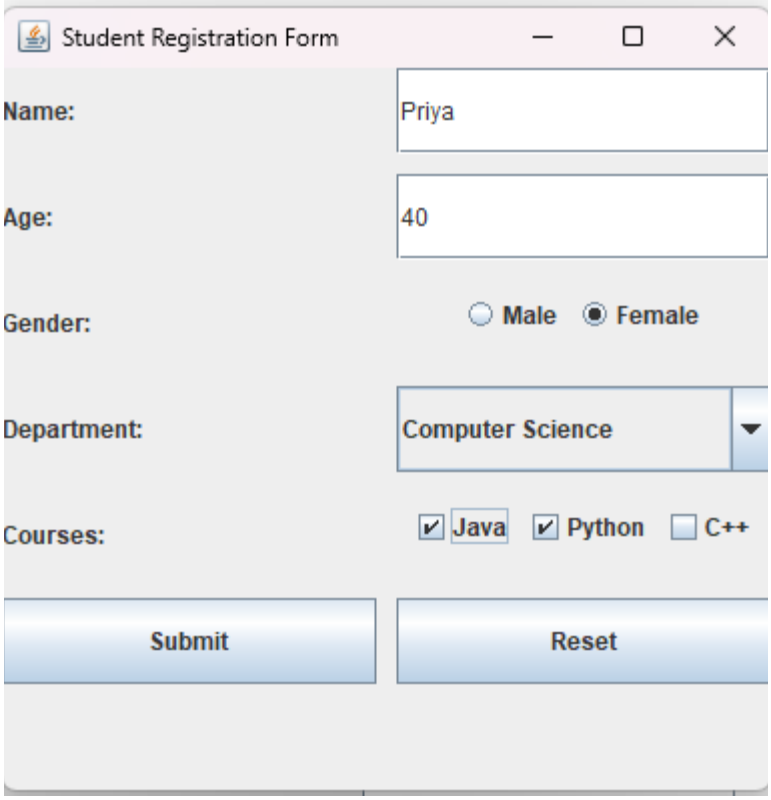
```
        add(coursePanel);

        // Submit Button
        submitButton = new JButton("Submit");
        submitButton.addActionListener(this);
        add(submitButton);

        // Reset Button
        resetButton = new JButton("Reset");
        resetButton.addActionListener(e -> {
            nameField.setText("");
            ageField.setText("");
            genderGroup.clearSelection();
            departmentBox.setSelectedIndex(0);
            javaCheck.setSelected(false);
            pythonCheck.setSelected(false);
            cppCheck.setSelected(false);
        });
        add(resetButton);

        setVisible(true);
    }

    public void actionPerformed(ActionEvent e) {
        String name = nameField.getText();
        String age = ageField.getText();
        String gender = male.isSelected() ? "Male" : female.isSelected() ? "Female" : "Not Selected";
        String department = (String) departmentBox.getSelectedItem();
        String courses = "";
        if (javaCheck.isSelected()) courses += "Java ";
        if (pythonCheck.isSelected()) courses += "Python ";
        if (cppCheck.isSelected()) courses += "C++ ";

        JOptionPane.showMessageDialog(this,
            "Name: " + name + "\nAge: " + age + "\nGender: " + gender + "\nDepartment: " + department +
"\nCourses: " + courses,
            "Registration Details",
            JOptionPane.INFORMATION_MESSAGE);
    }

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

Output

Save the file as: StudentRegistrationForm.java

Compile the program:
javac StudentRegistrationForm.java
Run the program:
java StudentRegistrationForm





Result

Thus to write a Java program to design student registration forms using Swing controls has been completed successfully and output is verified.

Ex No 6.4 Write a program to display the digital watch in swing

Aim

To write a program to display the digital watch in swing

Algorithm

Initialize Swing Components

Create a JFrame for the main window.Add a JLabel to display the time.

Set Layout and Styling

Use a suitable layout (FlowLayout, BorderLayout, etc.).Set font size and style for better visibility.

Create a Timer Using javax.swing.Timer

Set a timer to trigger every second (1000ms).

Get the Current Time

Use LocalTime or SimpleDateFormat to fetch the system time.

Format the Time Properly

Convert time to a readable format (HH:mm:ss).

Update the JLabel with Current Time

Set the formatted time as text in the label.

Start the Timer

Use timer.start() to continuously update the time.

Handle Window Closing

Set setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE).

Make the Frame Visible

Use setVisible(true) to display the watch.

Run the Application in the Main Method

Instantiate the watch class and execute it.

Program

```
import javax.swing.*;
import java.awt.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DigitalClock extends JFrame {
   private JLabel timeLabel;
   private SimpleDateFormat timeFormat;
```

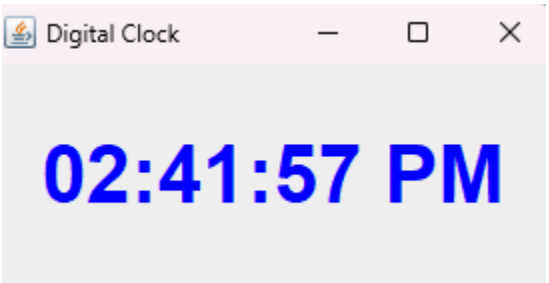
```
    public DigitalClock() {
        setTitle("Digital Clock");
        setSize(300, 150);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new BorderLayout());

        // Time format (24-hour or 12-hour format)
        timeFormat = new SimpleDateFormat("hh:mm:ss a"); // 12-hour format with AM/PM
        // timeFormat = new SimpleDateFormat("HH:mm:ss"); // Uncomment for 24-hour format

        // JLabel to display the time
        timeLabel = new JLabel();
        timeLabel.setHorizontalAlignment(SwingConstants.CENTER);
        timeLabel.setFont(new Font("Arial", Font.BOLD, 40));
        timeLabel.setForeground(Color.BLUE);

        add(timeLabel, BorderLayout.CENTER);

        // Timer to update the time every second
        Timer timer = new Timer(1000, e -> updateTime());
        timer.start();

        updateTime(); // Initial time display
        setVisible(true);
    }

    private void updateTime() {
        String time = timeFormat.format(new Date());
        timeLabel.setText(time);
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(DigitalClock::new);
    }
}
```

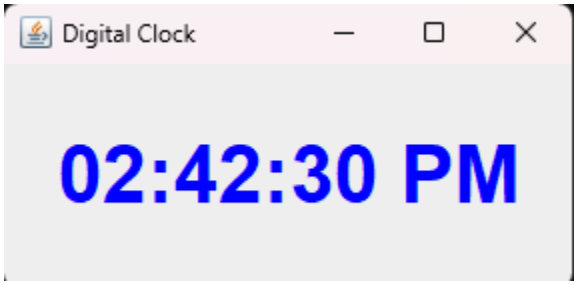
Output

Save the file as: DigitalClock.java

Compile the program:
javac DigitalClock.java
Run the program:
java DigitalClock

Result





Result

Thus to write a java program to display the digital watch in swing has been completed successfully and output is verified.