MIT World Peace University

Object Oriented Programming with Java and C++
Second Year B. Tech, Semester 1

Developing a Simple Graphical Calculator using Swing in Java

Practical Report
Assignment 8

Prepared By

Krishnaraj Thadesar
Cyber Security and Forensics
Batch A1, PA 20

November 22, 2022

Contents

1 Aim and Objectives

Aim

To Develop a simple calculator using Swing in Java

Objective

1. To understand concept of AWT and Java swings
2. To explore Java Swing containers

2 Problem Statement

Write a Java program to create a simple calculator with the help of java swing.

3 Theory

3.1 Java Swing containers

Containers are an integral part of SWING GUI components. A container provides a space where a component can be located. A Container in AWT is a component itself and it provides the capability to add a component to itself. Following are certain noticable points to be considered.

1. Panel: JPanel is the simplest container. It provides space in which any other component can be placed, including other panels.
2. Frame: A JFrame is a top-level window with a title and a border.
3. Window: A JWindow object is a top-level window with no borders and no menubar.

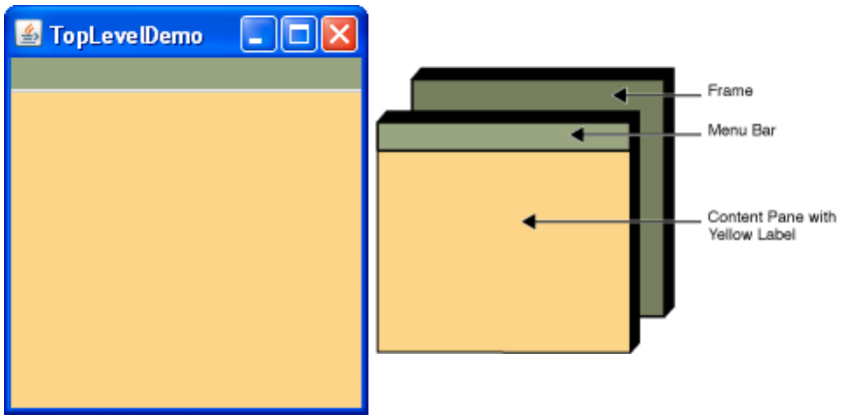

Figure 1: Top level containers in Java

3.2 Container classes of Java Swing with examples

3.2.1 JPanel

A panel is a component that is contained inside a frame window. A frame can have more than one-panel components inside it with each panel component having several other components.

```
import javax.swing.*;
class JPanelExample {
    JPanelExample(){
        JFrame frame = new JFrame("Panel Example"); //create a frame
        JPanel panel = new JPanel(); //Create JPanel Object
        panel.setBounds(40,70,100,100); //set dimensions for Panel
        JButton b = new JButton("ButtonInPanel"); //create JButton object
        b.setBounds(60,50,80,40); //set dimensions for button
        panel.add(b);   //add button to the panel
        frame.add(panel);   //add panel to frame
        frame.setSize(400,400);
        frame.setLayout(null);
        frame.setVisible(true);
    }

}
public class Main {
    public static void main(String[] args) {
      new JPanelExample(); //create an object of FrameInherited class
    }
}
```

3.2.2 JFrame

A Frame, in general, is a container that can contain other components such as buttons, labels, text fields, etc. A Frame window can contain a title, a border, and also menus, text fields, buttons, and other components. An application should contain a frame so that we can add components inside it.

The Frame in Java Swing is defined in class javax.swing.JFrame. JFrame class inherits the java.awt.Frame class. JFrame is like the main window of the GUI application using swing.

```
import javax.swing.*;
class FrameInherited extends JFrame{    //inherit from JFrame class
    JFrame f;
    FrameInherited(){
        JButton b=new JButton("JFrame_Button");//create button object
        b.setBounds(100,50,150, 40);

        add(b);//add button on frame
        setSize(300,200);
        setLayout(null);
        setVisible(true);
    }
}
public class Main {
    public static void main(String[] args) {
      new FrameInherited(); //create an object of FrameInherited class
    }
}
```

3.2.3 JWindow

The class JWindow is a container that can be displayed but does not have the title bar or window-management buttons.

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

class SwingContainerDemo {
    private JFrame mainFrame;
    private JLabel headerLabel;
    private JLabel statusLabel;
    private JPanel controlPanel;
    private JLabel msglabel;

    public SwingContainerDemo() {
        prepareGUI();
    }

    public static void main(String[] args) {
        SwingContainerDemo swingContainerDemo = new SwingContainerDemo();
        swingContainerDemo.showJWindowDemo();
    }

    private void prepareGUI() {
        mainFrame = new JFrame("Java Swing Examples");
        mainFrame.setSize(400, 400);
        mainFrame.setLayout(new GridLayout(3, 1));

        mainFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent windowEvent) {
                System.exit(0);
            }
        });
        headerLabel = new JLabel("", JLabel.CENTER);
        statusLabel = new JLabel("", JLabel.CENTER);
        statusLabel.setSize(350, 100);

        controlPanel = new JPanel();
        controlPanel.setLayout(new FlowLayout());

        mainFrame.add(headerLabel);
        mainFrame.add(controlPanel);
        mainFrame.add(statusLabel);
        mainFrame.setVisible(true);
    }
    private void showJWindowDemo() {
        headerLabel.setText("Container in action: JWindow");
        mainFrame.setVisible(true);
    }

}
```

3.3 Swing components including buttons, checkboxes, sliders, and list boxes, etc.

There are many imporant Swing components that allow us to design GUIs. Here are some of them.

- **JTextArea**: TextArea defines an editable text field. It can have multiple lines. The swing class that defines the text area is JTextArea and it inherits the JTextComponent class.
- **JButton**: A button is a component that is used to create a push button with a name or label on it. In swing, the class that creates a labeled button is JButton. JButton inherits the Abstract-Button class. We can associate the ActionListener event to the button to make it take some action when it is pushed.
- **JList**: A list consists of multiple text items. Users can either select a single item **or** multiple items at a time. The class that implements the list in swing API is JList. JList is a descendent of the JComponent class.
- **JComboBox**: The JCombobox class shows a list of choices from which a user can select an option. The selected choice is at the top. JComboBox derives from the JComponent class.
- **JSlider**: A slider allows us to select a specific range of values. In Java Swing API, JSlider is the class that is used to implement the slider.
- **JCheckBox**: The JcheckBox class is used to create chekbox in swing framework.
- **JRadioButton**: Radio button is a group of related button in which only one can be selected. JRadioButton class is used to create a radio button in Frames.
- **JLabel**: In Java, Swingtoolkit contains a JLabel Class. It is under package javax.swing.JLabel class. It is used for placing text in a box. Only Single line text is allowed and the text can not be changed directly.
- **JPasswordField**: In Java, Swing toolkit contains a JPasswordField Class. It is under package javax.swing.JPasswordField class. It is specifically used for password and it can be edited.

3.4 Heavyweight Components and Lightweight Components

There are two kinds of graphics components in the Java programming language: heavyweight and lightweight.

A component is said to be a heavyweight component if it uses native code provided by your computer's operating system to display buttons, choice lists, text fields, and the like. Such operating-system routines are said to be the components's peers.

A Swing component is said to be a lightweight component because it written entirely in Java and does the high-level display work itself, rather than relying on code provided by your computer's operating system.

3.5 What is Double Buffering?

1. Double-buffering is the process of drawing graphics into an off-screen image buffer and then copying the contents of the buffer to the screen all at once.
2. For the complex graphics, using double-buffering can reduce flickering issues.
3. Java Swing automatically supports double-buffering for all of its components.

4. Double-buffering is memory intensive, its use is only justified for components that are repainted very frequently or have particularly complex graphics to display.

5. If a container uses double-buffering, any double-buffered children it has shared the off-screen buffer of the container, the required off-screen buffer is never larger than the on-screen size of the application.

6. To enable double buffering, simply call the setDoubleBuffered() method (inherited from JComponent) to set the double-buffered property to true for any components that should use double-buffered drawing.

3.6 Difference between applet and Swing

Applet is a Java program that can be embedded into a web page. It runs inside the web browser and works at client side. An applet is embedded in an HTML page using the APPLET or OBJECT tag and hosted on a web server. They are used to make the web site more dynamic and entertaining. All applets are sub-classes (either directly or indirectly) of java.applet.Applet class. Applets are not stand-alone programs. Instead, they run within either a web browser or an applet viewer. JDK provides a standard applet viewer tool called applet viewer.

Swing in Java is a lightweight GUI toolkit which has a wide variety of widgets for building optimized window based applications. It is a part of the JFC(Java Foundation Classes). It is build on top of the AWT API and entirely written in java.

It is platform independent unlike AWT and has lightweight components. It becomes easier to build applications since we already have GUI components like button, checkbox etc. This is helpful because we do not have to start from the scratch.

4 Platform

Operating System: Arch Linux x86-64
IDEs or Text Editors Used: Visual Studio Code
Compilers : g++ and gcc on linux for C++, and javac, with JDK 18.0.2 for Java

5 Input

The numbers and the Operators.

6 Output

The Output of the entered Calculation in the Display Section of the Calculator

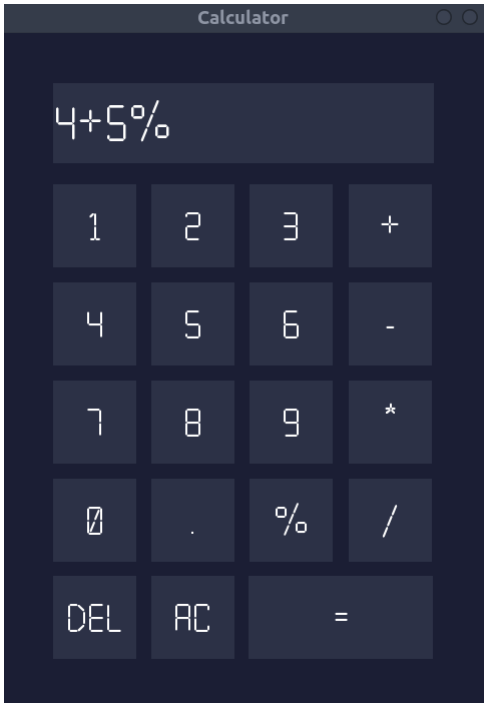

Figure 2: Calculator with Java Swing

7 Code

```
// Krishnaraj Thadesar
// Batch A1, PA20
// OOPCJ Assignment 9
// Making a Calculator in Java using Swing

package org.OOPCJ.Krishnaraj;
import org.mariuszgromada.math.mxparser.*;

import javax.swing.*;
import java.awt.*;
import java.io.*;

class Colors {
    static Color primaryColor = new Color(255, 255, 255); // text color
    static Color bgColor = new Color(27, 30, 52); // background
    static Color secondaryColor = new Color(44, 49, 70); // upper background
    static Color secondaryColorRollover = new Color(53, 59, 80); // upper
    background
    static Color accentColor = new Color(26, 122, 230); // Accent
}

class Numpad extends JPanel {
    JButton[] numbers = new JButton[12];

    Numpad() {
        this.setFocusable(true);
        this.setVisible(true);
        this.setBorder(null);
```

```
28          this.setBounds(50, 150, 280, 380);
29          this.setBackground(Colors.bgColor);
30          this.setLayout(new GridLayout(4, 3, 15, 15));
31          createButtons();
32          for (int i = 0; i < numbers.length; i++) {
33              this.add(numbers[i]);
34          }
35      }
36
37      public void createButtons() {
38          for (int i = 0; i < 12; i++) {
39              numbers[i] = new JButton();
40              numbers[i].setText(String.valueOf(i + 1));
41              numbers[i].setFocusPainted(false);
42              numbers[i].setContentAreaFilled(false);
43              numbers[i].setOpaque(true);
44              numbers[i].setBorder(null);
45              numbers[i].setBackground(Colors.secondaryColor);
46              numbers[i].setForeground(Colors.primaryColor);
47              numbers[i].setFont(Calculator.buttonFont);
48              final JButton temp = numbers[i];
49              temp.addChangeListener(evt -> {
50                  if (temp.getModel().isPressed()) {
51                      temp.setForeground(Colors.primaryColor);
52                      temp.setBackground(Colors.secondaryColorRollover);
53                  } else if (temp.getModel().isRollover()) {
54                      temp.setForeground(Colors.accentColor);
55                      temp.setBackground(Colors.secondaryColorRollover);
56                  } else {
57                      temp.setForeground(Colors.primaryColor);
58                      temp.setBackground(Colors.secondaryColor);
59                  }
60              });
61              temp.addActionListener(e -> {
62                  Calculator.display.setText(Calculator.display.getText() + ((
    JButton) e.getSource()).getText());
63              });
64          }
65          numbers[9].setText("0");
66          numbers[10].setText(".");
67          numbers[11].setText("%");
68      }
69  }
70
71  class Operators_pnl extends JPanel {
72      static JButton[] operators = new JButton[4];
73      static String currentOperator;
74
75      Operators_pnl() {
76          this.setFocusable(true);
77          this.setVisible(true);
78          this.setBorder(null);
79          this.setBounds(345, 150, (int) 250 / 3, 380);
80          this.setBackground(Colors.bgColor);
81          this.setLayout(new GridLayout(4, 1, 0, 15));
82          createButtons();
83          for (int i = 0; i < operators.length; i++) {
84              this.add(operators[i]);
85          }
```

```
86      }
87
88      public void createButtons() {
89          for (int i = 0; i < 4; i++) {
90              operators[i] = new JButton();
91              operators[i].setFocusPainted(false);
92              operators[i].setContentAreaFilled(false);
93              operators[i].setOpaque(true);
94              operators[i].setBorder(null);
95              operators[i].setBackground(Colors.secondaryColor);
96              operators[i].setForeground(Colors.primaryColor);
97              operators[i].setFont(Calculator.buttonFont);
98              final JButton temp = operators[i];
99              temp.addChangeListener(evt -> {
100                 if (temp.getModel().isPressed()) {
101                     temp.setForeground(Colors.primaryColor);
102                     temp.setBackground(Colors.secondaryColorRollover);
103                 } else if (temp.getModel().isRollover()) {
104                     temp.setForeground(Colors.accentColor);
105                     temp.setBackground(Colors.secondaryColorRollover);
106                 } else {
107                     temp.setForeground(Colors.primaryColor);
108                     temp.setBackground(Colors.secondaryColor);
109                 }
110             });
111             temp.addActionListener(e -> {
112                 if (!Calculator.operator_used) {
113                     Calculator.display.setText(Calculator.display.getText() + ((
    JButton) e.getSource()).getText());
114                 }
115             });
116         }
117         operators[0].setText("+");
118         operators[1].setText("-");
119         operators[2].setText("*");
120         operators[3].setText("/");
121     }
122 }
123
124 // Main Calculator Frame no panels
125 class Calculator extends JFrame {
126     static double number_1, number_2;
127     static boolean operator_used = false;
128     JButton clearBtn, backspaceBtn, resultBtn;
129     static JTextField display;
130     Numpad numpad;
131     Operators_pnl operators_pnl;
132     static Font buttonFont;
133
134     Calculator() {
135         this.setTitle("Calculator");
136         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
137         this.setResizable(false);
138         this.setUndecorated(false);
139         this.setPreferredSize(new Dimension(480, 670));
140         this.getContentPane().setBackground(Colors.bgColor);
141         this.setLayout(null);
142         createFonts();
143         createPanels();
```

```
144         createButtons();
145         this.add(display);
146         this.add(numpad);
147         this.add(operators_pnl);
148         this.add(clearBtn);
149         this.add(backspaceBtn);
150         this.add(resultBtn);
151         this.pack();
152         this.setVisible(true);
153         this.setLocationRelativeTo(null);
154     }
155 
156     public void createPanels() {
157         numpad = new Numpad();
158         operators_pnl = new Operators_pnl();
159     }
160 
161     public void createButtons() {
162         display = new JTextField();
163         display.setBounds(50, 50, 380, 80);
164         display.setOpaque(true);
165         display.setAlignmentX(RIGHT_ALIGNMENT);
166         display.setBorder(null);
167         display.setBackground(Colors.secondaryColor);
168         display.setForeground(Colors.primaryColor);
169         display.setFont(Calculator.buttonFont.deriveFont(55f));
170         display.addActionListener(e -> {
171         });
172 
173         clearBtn = new JButton();
174         clearBtn.setText("AC");
175         clearBtn.setBounds(50 + 15 + (int) 250 / 3, 542, (int) 250 / 3, (int) 250
    / 3);
176         clearBtn.setFocusPainted(false);
177         clearBtn.setContentAreaFilled(false);
178         clearBtn.setOpaque(true);
179         clearBtn.setBorder(null);
180         clearBtn.setBackground(Colors.secondaryColor);
181         clearBtn.setForeground(Colors.primaryColor);
182         clearBtn.setFont(Calculator.buttonFont);
183         clearBtn.addChangeListener(evt -> {
184             if (clearBtn.getModel().isPressed()) {
185                 clearBtn.setForeground(Colors.primaryColor);
186                 clearBtn.setBackground(Colors.secondaryColorRollover);
187             } else if (clearBtn.getModel().isRollover()) {
188                 clearBtn.setForeground(Colors.accentColor);
189                 clearBtn.setBackground(Colors.secondaryColorRollover);
190             } else {
191                 clearBtn.setForeground(Colors.primaryColor);
192                 clearBtn.setBackground(Colors.secondaryColor);
193             }
194         });
195         clearBtn.addActionListener(e -> {
196             display.setText("");
197             operator_used = false;
198             Operators_pnl.operators[0].setEnabled(true);
199             Operators_pnl.operators[1].setEnabled(true);
200             Operators_pnl.operators[2].setEnabled(true);
201             Operators_pnl.operators[3].setEnabled(true);
```

```
202         });
203
204         backspaceBtn = new JButton();
205         backspaceBtn.setText("DEL");
206         backspaceBtn.setBounds(50, 542, (int) 250 / 3, (int) 250 / 3);
207         backspaceBtn.setFocusPainted(false);
208         backspaceBtn.setContentAreaFilled(false);
209         backspaceBtn.setOpaque(true);
210         backspaceBtn.setBorder(null);
211         backspaceBtn.setBackground(Colors.secondaryColor);
212         backspaceBtn.setForeground(Colors.primaryColor);
213         backspaceBtn.setFont(Calculator.buttonFont);
214         backspaceBtn.addChangeListener(evt -> {
215             if (backspaceBtn.getModel().isPressed()) {
216                 backspaceBtn.setForeground(Colors.primaryColor);
217                 backspaceBtn.setBackground(Colors.secondaryColorRollover);
218             } else if (backspaceBtn.getModel().isRollover()) {
219                 backspaceBtn.setForeground(Colors.accentColor);
220                 backspaceBtn.setBackground(Colors.secondaryColorRollover);
221             } else {
222                 backspaceBtn.setForeground(Colors.primaryColor);
223                 backspaceBtn.setBackground(Colors.secondaryColor);
224             }
225         });
226         backspaceBtn.addActionListener(e -> {
227             try {
228                 display.setText(display.getText().substring(0, display.getText().
    length() - 1));
229             } catch (Exception f) {
230                 System.out.println("You got nothing on screen then how can you
    delete? ");
231             }
232         });
233
234         resultBtn = new JButton();
235         resultBtn.setText("=");
236         resultBtn.setBounds(245, 542, (int) 184, (int) 250 / 3);
237         resultBtn.setFocusPainted(false);
238         resultBtn.setContentAreaFilled(false);
239         resultBtn.setOpaque(true);
240         resultBtn.setBorder(null);
241         resultBtn.setBackground(Colors.secondaryColor);
242         resultBtn.setForeground(Colors.primaryColor);
243         resultBtn.setFont(Calculator.buttonFont);
244         resultBtn.addChangeListener(evt -> {
245             if (resultBtn.getModel().isPressed()) {
246                 resultBtn.setForeground(Colors.primaryColor);
247                 resultBtn.setBackground(Colors.secondaryColorRollover);
248             } else if (resultBtn.getModel().isRollover()) {
249                 resultBtn.setForeground(Colors.accentColor);
250                 resultBtn.setBackground(Colors.secondaryColorRollover);
251             } else {
252                 resultBtn.setForeground(Colors.primaryColor);
253                 resultBtn.setBackground(Colors.secondaryColor);
254             }
255         });
256         resultBtn.addActionListener(e -> {
257             String currentString = display.getText();
258             Expression expr = new Expression(currentString);
```

```
            display.setText(String.valueOf(expr.calculate()));
        });
    }

    public static void createFonts() {
        try {
            buttonFont = Font.createFont(Font.TRUETYPE_FONT, new File("/run/media/krishnaraj/Classes/University/Second Year/First Semister/OOPJC/Programs/java_implementations/assignment_8/Calculator/src/main/resources/Calculator.ttf")).deriveFont(45f);
            GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
            // register the font
            ge.registerFont(buttonFont);

        } catch (FontFormatException | IOException e) {
            e.printStackTrace();
        }
    }
}

public class Main {
    static Calculator calc;

    public static void main(String[] args) {
        calc = new Calculator();
    }
}
```

Listing 1: Calculator.java

8 Dependencies

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>org.example</groupId>
    <artifactId>org.OOPCJ.Krishnaraj.Calculator</artifactId>
    <version>1.0-SNAPSHOT</version>

    <properties>
        <maven.compiler.source>18</maven.compiler.source>
        <maven.compiler.target>18</maven.compiler.target>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.mariuszgromada.math</groupId>
            <artifactId>MathParser.org-mXparser</artifactId>
            <version>5.0.7</version>
        </dependency>

    </dependencies>

```

```
</project>
```

Listing 2: pom.xml

9 Conclusion

Thus, implemented simple calculator with the help of java swing and performed various operations.

10 FAQs

1. *Methods of component class in Java Swing?*

   A component is an object having a graphical representation that can be displayed on the screen and that can interact with the user. Examples of components are the buttons, checkboxes, and scrollbars of a typical graphical user interface.

   The Component class is the abstract superclass of the nonmenu-related Abstract Window Toolkit components. Class Component can also be extended directly to create a lightweight component. A lightweight component is a component that is not associated with a native window. On the contrary, a heavyweight component is associated with a native window. The isLightweight() method may be used to distinguish between the two kinds of the components.

   Some methods of this class are:

   - **void add(PopupMenu):** Adds the specified popup menu to the component.
   - **void addComponentListener(ComponentListener):** Adds the specified component listener to receive component events from this component.
   - **void addFocusListener(FocusListener):** Adds the specified focus listener to receive focus events from this component when this component gains input focus.
   - **Rectangle getBounds(): Gets** the bounds of this component in the form of a Rectangle object.
   - **Rectangle getBounds(Rectangle):** Stores the bounds of this component into "return value" rv and return rv.

2. *Ways to create a frame in Java Swing? Explain with examples*

   (a) By creating the object of Frame class (association)

```
// Java program to create frames
// using association

import javax.swing.*;
public class test1
{
  JFrame frame;

  test1()
  {
    // creating instance of JFrame with name "first way"
    frame=new JFrame("first way");

    // creates instance of JButton
```

```
    JButton button = new JButton("let's see");

    button.setBounds(200, 150, 90, 50);

    // setting close operation
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // adds button in JFrame
    frame.add(button);

    // sets 500 width and 600 height
    frame.setSize(500, 600);

    // uses no layout managers
    frame.setLayout(null);

    // makes the frame visible
    frame.setVisible(true);
  }

  public static void main(String[] args)
  {
    new test1();
  }
}
```

(b) By extending Frame class (inheritance)

```
// Java program to create a
// frame using inheritance().

import javax.swing.*;

// inheriting JFrame
public class test2 extends JFrame
{
  JFrame frame;
  test2()
  {
    setTitle("this is also a title");

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

    button.setBounds(165, 135, 115, 55);

    // adding button on frame
    add(button);

    // setting close operation
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    setSize(400, 500);
    setLayout(null);
    setVisible(true);
  }

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

```
    new test2();
  }
}


```

(c) Create a frame using Swing inside main()

```
// Java program to create a frame
// using Swings in main().

import javax.swing.*;
public class Swing_example
{
  public static void main(String[] args)
  {
    // creates instance of JFrame
    JFrame frame1 = new JFrame();

    // creates instance of JButton
    JButton button1 = new JButton("click");
    JButton button2 = new JButton("again click");

    // x axis, y axis, width, height
    button1.setBounds(160, 150 ,80, 80);
    button2.setBounds(190, 190, 100, 200);

    // adds button1 in Frame1
    frame1.add(button1);

    // adds button2 in Frame1
    frame1.add(button2);

    // 400 width and 500 height of frame1
    frame1.setSize(400, 500) ;

    // uses no layout managers
    frame1.setLayout(null);

    // makes the frame visible
    frame1.setVisible(true);
  }
}


```

3. *What are the Methods of JLabel class in Java Swing?*

- **String getText()**: Returns the text string that the label displays.
- **LabelUI getUI()**: Returns the LF object that renders this component.
- **String getUIClassID()**: Returns a string that specifies the name of the lf class that renders this component.
- **void setIcon(Icon icon)**: Defines the icon this component will display.
- **void setIconTextGap(int iconTextGap)**: If both the icon and text properties are set, this property defines the space between them.
- **void setLabelFor(Component c)**: Set the component this is labelling.

- **void setText(String text)**: Defines the single line of text this component will display.
- **void setUI(LabelUI ui)**: Sets the LF object that renders this component.
- **void setVerticalAlignment(int alignment)**: Sets the alignment of the label's contents along the Y axis.
- **void setVerticalTextPosition(int textPosition)**: Sets the vertical position of the label's text, relative to its image.
- **void updateUI()**: Resets the UI property to a value from the current look and feel.

4. *What are the Methods of AbstractButton class in Java Swing?*

- **protected ActionListener actionListener**: The button model's ActionListener.
- **protected ChangeEvent changeEvent**: Only one ChangeEvent is needed per button instance since the event's only state is the source property.
- **protected ChangeListener changeListener**: The button model's changeListener.

5. *Write a simple Java Swing program of displaying image on the button?*

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

class test extends JFrame {

    test() {
        JButton bt1 = new JButton("no"); // Creating a Yes Button.
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); // setting close
    operation.
        bt1.setBounds(60, 50, 500, 500);
        ImageIcon imageIcon = new ImageIcon("../../Lab/pic.jpg");
        bt1.setIcon(imageIcon);
        setLayout(null); // setting layout using FlowLayout object
        setSize(700, 700); // setting size of Jframe
        add(bt1); // adding Yes button to frame.
        setVisible(true);
    }

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

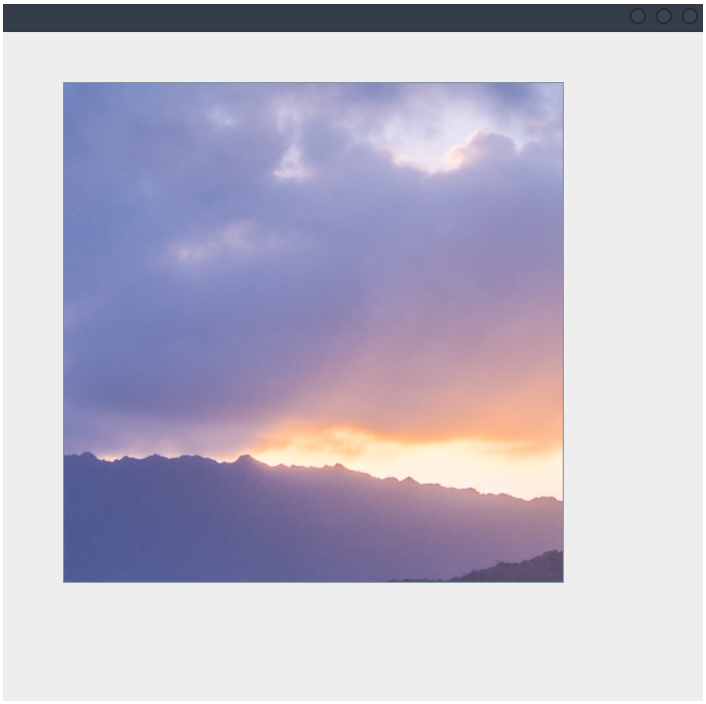

Figure 3: Button with a background image in its background.