A MINOR PROJECT

ON

JAVA CALCULATOR

Building a basic arithmetic tool

Submitted in partial fulfillment of the requirements for the award of the certificate

In

PROGRAMMING IN JAVA

By

VALLURI POOJAPRASANNA

Under Guidance of

istop
Keep Growing

CONTENTS

ABSTRACT

Java Swing is a GUI (graphical user Interface) widget toolkit for Java. Java Swing is a part of Oracle's Java foundation classes . Java Swing is an API for providing graphical user interface elements to Java Programs.Swing was created to provide more powerful and flexible components than Java AWT (Abstract Window Toolkit).

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.

Swing's flexibility and extensive feature set make it a popular choice for building desktop applications with complex GUIs. It also includes advanced components like tables, trees, and tabbed panes, making it suitable for a wide range of applications.

OBJECTIVE

The purpose of this project is to implement a traditional simple digital calculator that performs the four basic operations; Addition, Subtraction, Multiplication, and Division of 8 bits unsigned numbers.

To develop an simple calculator using java which provides:

→ Simple arithmetic operation.
→ The operation performed are
→ Addition.
→ Subtraction.
→ Multiplication.
→ Division.
→ Clearing the content of the screen.

INTRODUCTION

Swing API is a set of extensible GUI Components to ease the developer's life to create JAVA based Front End/GUI Applications. It is built on top of AWT API and acts as a replacement of AWT API since it has almost every control corresponding to AWT controls.

Java Swing is a powerful toolkit for creating graphical user interfaces (GUIs) in Java. It provides a wide range of components, such as buttons, text fields, and menus, that can be used to create attractive and functional GUIs. Swing is platform-independent, which means that your GUIs will look the same on any platform, including Windows, Mac OS X, and Linux.

- What is Java Swing?
- How to install Java Swing
- Creating a simple Swing GUI
- Using Swing components
- Layout managers
- Event handling

What is Java Swing?

Java Swing is a set of graphical user interface (GUI) components that are part of the Java platform. Swing components are built on top of the Abstract Window Toolkit (AWT), but they provide a number of advantages over AWT components. For example, Swing components are more lightweight and efficient, and they are platform-independent.

How to install Java Swing

Java Swing is included in the Java Development Kit (JDK). If you already have the JDK installed, you can skip this section.

To install the JDK, go to the Oracle website and download the latest version of the JDK for your operating system. Once the JDK is installed, you can start creating Swing GUIs.

Creating a simple Swing GUI

To create a simple Swing GUI, you will need to create a new Java project and add a Swing library to the project. Once you have added the Swing library, you can start creating Swing components.

Using Swing components

Swing provides a wide range of components that can be used to create GUIs. Some of the most commonly used components include:

- Buttons
- Text fields
- Labels
- Menus
- Panels
- Layout managers

Layout managers

Layout managers are used to control the layout of components in a GUI. Swing provides a number of different layout managers, including:

- BorderLayout
- FlowLayout
- GridLayout
- GridBagLayout

Event handling

Swing components can respond to events, such as mouse clicks and key presses. To handle events, you need to add event listeners to the components. Event listeners are objects that implement the ActionListener, KeyListener, or other event listener interfaces.

Let's start by creating a simple GUI using Swing.

1. Step 1: Importing Swing Libraries. ...
2. Step 2: Creating a New JFrame. ...
3. Step 3: Setting the Frame Size. ...
4. Step 4: Making the Frame Visible. ...

5. Adding Buttons with JButton. ...
6. Creating Menus with JMenuItem. ...
7. Incorporating Text Fields with JTextField. ...
8. Exploring Different Layout Managers.

To create a simple Swing GUI, you will need to create a new Java project and add a Swing library to the project. Once you have added the Swing library, you can start creating Swing components. This code will create a simple GUI with a button and a text field.

Main Features of Swing Toolkit

- Platform Independent.
- Customizable.
- Extensible.
- Configurable.
- Lightweight.
- Rich Controls.
- Pluggable Look and Feel.

Below are the different components of swing in java:

→ ImageIcon. The ImageIcon component creates an icon sized-image from an image residing at the source URL. ...
→ JButton. JButton class is used to create a push-button on the UI. ...
→ JLabel. ...
→ JTextField. ...
→ JTextArea. ...
→ JPasswordField. ...
→ JCheckBox. ...
→ JRadioButton.

METHODOLOGY

- public void add(Component c)

Component clearly shows that is the graphical representation of an Object. Important methods of Component Class: public void add(Component c): This method inserts a component into a Container by taking a component as a parameter.

This method inserts the specified element E at the specified position in this list.It shifts the element currently at that position (if any) and any subsequent elements to the right (will add one to their indices).

This method inserts the specified element E at the specified position in this list.It shifts the element currently at that position (if any) and any subsequent elements to the right (will add one to their indices).

public means that the method will be visible from classes in other packages. static means that the method is not attached to a specific instance, and it has no " this ". It is more or less a function. void is the return type. It means "this method returns nothing".

- public void setSize(int width,int height)

public void setSize(int width, int height) Sets the size of this Dimension object to the specified width and height. This method is included for completeness, to parallel the setSize method defined by Component .

void means the return type in here you not return anything that's why use void if you need to return integer value you have to use int instead of the void key word.

setSize(int width, int height) Sets the size of this Dimension object to the specified width and height. java.lang.String. toString() Returns a string representation of the values of this Dimension object's height and width fields.

The setSize() method of Java Vector class is used to set the size of a vector. If the new size is greater than the current size, null items are added to the end of the vector. Otherwise, all components at index newSize and greater are discarded.

setSize() is a method of Vector class that is used to set the new size of the vector.

Java set methods

1. int size(): to get the number of elements in the Set.
2. boolean isEmpty(): to check if Set is empty or not.
3. boolean contains(Object o): Returns true if this Set contains the specified element.
4. Iterator iterator(): Returns an iterator over the elements in this set. The elements are returned in no particular order.
5. Object[] toArray(): Returns an array containing all of the elements in this set. If this set makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.
6. boolean add(E e): Adds the specified element to this set if it is not already present (optional operation).
7. boolean remove(Object o): Removes the specified element from this set if it is present (optional operation).
8. boolean removeAll(Collection c): Removes from this set all of its elements that are contained in the specified collection (optional operation).
9. boolean retainAll(Collection c): Retains only the elements in this set that are contained in the specified collection (optional operation).
10. void clear(): Removes all the elements from the set.
11. Iterator iterator(): Returns an iterator over the elements in this set.

- public void setLayout(LayoutManager m)

The setLayout(...) method allows you to set the layout of the container, often a JPanel, to say FlowLayout, BorderLayout, GridLayout, null layout, or whatever layout desired. The layout manager helps lay out the components held by this container. The setBounds(...).

setLayout method to change the layout manager, and you can define your own layout manager by implementing the java. awt. LayoutManager interface. This article describes the predefined AWT layout managers in this list. Java.awt.BorderLayout.

Each container has a layout manager, which is responsible for arranging the components in a container. The container's setLayout method can be used to set a layout manager. Certain types of containers have default layout managers.

You can install a new layout manager at any time by using the setLayout() method. For example, we can set the layout manager of a Swing container to a type called BorderLayout like so: myContainer . setLayout (new BorderLayout ());

public void setLayout(LayoutManager m) Defines the layout manager for the component.

Java LayoutManagers

There are the following classes that represent the layout managers:

- java.awt.BorderLayout
- java.awt.FlowLayout
- java.awt.GridLayout
- java.awt.CardLayout
- java.awt.GridBagLayout
- javax.swing.BoxLayout
- javax.swing.GroupLayout
- javax.swing.ScrollPaneLayout
- javax.swing.SpringLayout

SOURCE CODE

```
package calculator;


import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;


public class JavaCalculator implements ActionListener{


double input,result;

String cal;

JFrame frame;

JLabel label = new JLabel();

JTextField textView = new JTextField();



JButton symClr = new JButton("CLR");

JButton symDel = new JButton("DEL");

JButton symMul = new JButton("x");
```

```
JButton symDiv= new JButton("/");

JButton numSeven = new JButton("7");
JButton numEight = new JButton("8");
JButton numNine = new JButton("9");
JButton symMinus = new JButton("-");

JButton numFour = new JButton("4");
JButton numFive = new JButton("5");
JButton numSix = new JButton("6");
JButton symPlus = new JButton("+");

JButton numOne = new JButton("1");
JButton numTwo = new JButton("2");
JButton numThree = new JButton("3");
JButton symEqual = new JButton("=");

JButton numZero = new JButton("0");
JButton symDot = new JButton(".");


JavaCalculator(){
CreateInterface();
InterfaceComponents();
```

```
AddInterfaceEventListener();


}
public void CreateInterface() {
frame = new JFrame();
frame.setTitle("Java Calculator");
frame.getContentPane().setLayout(null);
frame.setLocationRelativeTo(null);
frame.setResizable(false);
frame.setSize(305,400);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public void InterfaceComponents() {
label.setBounds(200,0,70,40);
frame.add(label);

textView.setBounds(10, 40, 270, 60);
textView.setEditable(false);
textView.setHorizontalAlignment(SwingConstants.RIGHT);
frame.add(textView);

//First row
```

```
symClr.setBounds(10,110,60,40);
frame.add(symClr);
symDel.setBounds(80,110,60,40);
frame.add(symDel);
symMul.setBounds(150,110,60,40);
frame.add(symMul);
symDiv.setBounds(220,110,60,40);
frame.add(symDiv);

//Second row
numSeven.setBounds(10,160,60,40);
frame.add(numSeven);
numEight.setBounds(80,160,60,40);
frame.add(numEight);
numNine.setBounds(150,160,60,40);
frame.add(numNine);
symMinus.setBounds(220,160,60,40);
frame.add(symMinus);


//third row
numFour.setBounds(10,210,60,40);
frame.add(numFour);
numFive.setBounds(80,210,60,40);
```

```
frame.add(numFive);
numSix.setBounds(150,210,60,40);
frame.add(numSix);
symPlus.setBounds(220,210,60,40);
frame.add(symPlus);

//fourth row
numOne.setBounds(10,260,60,40);
frame.add(numOne);
numTwo.setBounds(80,260,60,40);
frame.add(numTwo);
numThree.setBounds(150,260,60,40);
frame.add(numThree);
symEqual.setBounds(220,260,60,90);
frame.add(symEqual);

//Fifth row
numZero.setBounds(10,310,130,40);
frame.add(numZero);
symDot.setBounds(150,310,60,40);
frame.add(symDot);

}
public void AddInterfaceEventListener() {
```

```
//First row
symClr.addActionListener(this);
symDel.addActionListener(this);
symMul.addActionListener(this);
symDiv.addActionListener(this);
//Second row
numSeven.addActionListener(this);
numEight.addActionListener(this);
numNine.addActionListener(this);
symMinus.addActionListener(this);
//Third row
numFour.addActionListener(this);
numFive.addActionListener(this);
numSix.addActionListener(this);
symPlus.addActionListener(this);
//Four row
numOne.addActionListener(this);
numTwo.addActionListener(this);
numThree.addActionListener(this);
symEqual.addActionListener(this);
//fifth row
numZero.addActionListener(this);
symDot.addActionListener(this);
}
```

```
@Override
public void actionPerformed(ActionEvent e) {
Object event = e.getSource();

if(event == numOne){
textView.setText(textView.getText()+"1");
}else if (event == numTwo){
textView.setText(textView.getText()+"2");
}else if(event == numThree){
textView.setText(textView.getText()+"3");
}else if(event == numFour){
textView.setText(textView.getText()+"4");
}else if(event == numFive){
textView.setText(textView.getText()+"5");
}else if(event == numSix){
textView.setText(textView.getText()+"6");
}else if (event == numSeven){
textView.setText(textView.getText()+"7");
} else if (event == numEight){
textView.setText(textView.getText()+"8");
}else if (event == numNine){
textView.setText(textView.getText()+"9");
}else if(event == numZero){
```

```
if (textView.getText().equals("0")) {
return;
} else {
textView.setText(textView.getText()+"0");
}
} else if(event == symDot) {
if (textView.getText().equals(0)) {
return;
}else {
textView.setText(textView.getText()+".");
}
}else if(event == symClr) {
label.setText("");
textView.setText("");
}else if(event == symDel) {
int length = textView.getText().length();
int number = length-1;
if(length>0) {
StringBuilder numString = new StringBuilder(textView.getText());
numString.deleteCharAt(number);
textView.setText(numString.toString());
}
if (textView.getText().endsWith("")) {
label.setText("");
```

```
}
} else if(event == symMul) {
String presentNumber = textView.getText();
input = Double.parseDouble(textView.getText());
textView.setText("");
label.setText(presentNumber+" * ");
cal = "*";
}else if(event == symDiv) {
String presentNumber = textView.getText();
input = Double.parseDouble(textView.getText());
textView.setText("");
label.setText(presentNumber+" / ");
cal = "/";
} else if(event == symMinus) {
String presentNumber = textView.getText();
input = Double.parseDouble(textView.getText());
textView.setText("");
label.setText(presentNumber+" - ");
cal = "-";
} else if(event == symPlus) {
String presentNumber = textView.getText();
input = Double.parseDouble(textView.getText());
textView.setText("");
label.setText(presentNumber+" + ");
```

```
cal = "+";
} else if(event == symEqual) {
switch(cal) {
case "*" : result = input * (Double.parseDouble(textView.getText()));
        if (Double.toString(result).endsWith(".0")) {
            textView.setText(Double.toString(result).replace(".0",""));
        }else {
            textView.setText(Double.toString(result));
        }
        label.setText("");
        break;
case "/" : result = input / (Double.parseDouble(textView.getText()));
                if (Double.toString(result).endsWith(".0")) {
                    textView.setText(Double.toString(result).replace(".0",""));
                }else {
                    textView.setText(Double.toString(result));
                }
                label.setText("");
                break;
case "-" : result = input - (Double.parseDouble(textView.getText()));
                    if(Double.toString(result).endsWith(".0")) {
                    textView.setText(Double.toString(result).replace(".0",""));
                    }else {
                    textView.setText(Double.toString(result));
```

```
                    }
                        label.setText("");
                        break;
case "+" : result = input + (Double.parseDouble(textView.getText()));
                    if (Double.toString(result).endsWith(".0")) {
                    textView.setText(Double.toString(result).replace(".0",""));
                    }else {
                    textView.setText(Double.toString(result));
                    }
                        label.setText("");
                        break;
}
}
}

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

OUTPUT

Input(Multiplication)

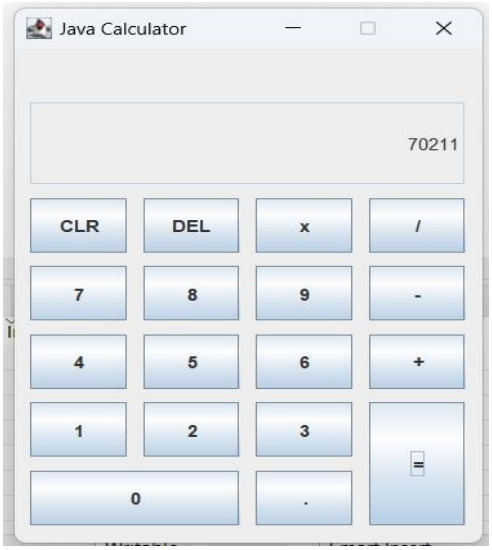
Output(Multiplication)

Input(Substraction)

Output(Substraction)

Input(Addition)

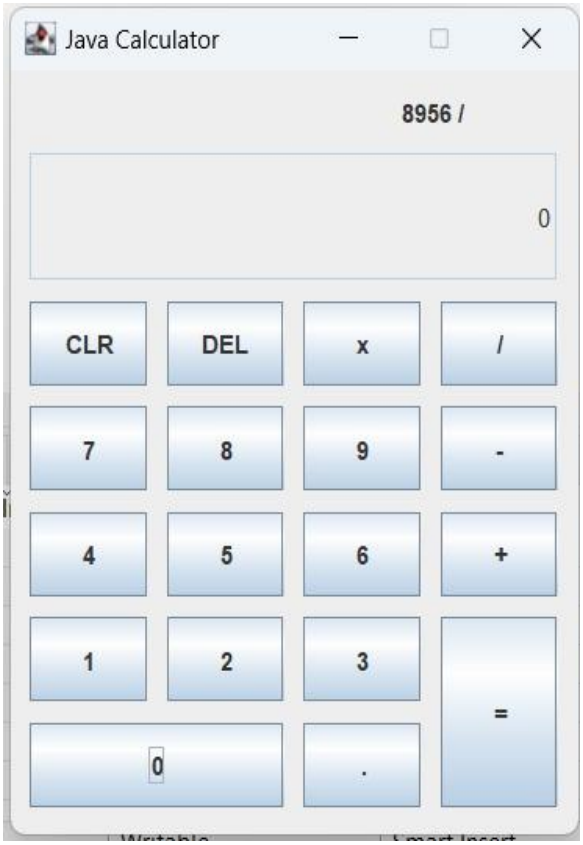
Output(Addition)



Input(Division)

Output(Division)

Input(Division by Zero)



Output(Division by Zero)

CONCLUSION

Swing is more portable and more flexible than AWT, the Swing is built on top of the AWT. Swing is Entirely written in Java. Java Swing Components are Platform-independent, and The Swing Components are lightweight. Swing Supports a Pluggable look and feel and Swing provides more powerful components.

Java Swing is a powerful GUI toolkit that provides a rich set of components for creating desktop applications. Java Swing is a popular choice for desktop applications, and it continues to be widely used by developers worldwide.

This project helps developers **develop** real-world projects to hone their skills and materialise their theoretical knowledge into practical experience.