

# Python & Java {4} Teachers



# GUI Calculator

Level 3 - Java



Funded by the  
Erasmus+ Programme  
of the European Union

{4}

Python & Java 4 Teachers

# JAVA

{4}

Python & Java 4 Teachers

Java is a high-level, class-based, object-oriented programming language that is designed to have as few implementation dependencies as possible.

It is a general-purpose programming language intended to let programmers write once, run anywhere (WORA),[17] meaning that compiled Java code can run on all platforms that support Java without the need for recompilation.

Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle)

The Minecraft logo, featuring the word "MINECRAFT" in a stylized, blocky, 3D font with a dark grey and brown color scheme.

# Task

A Maths class has run out of calculators!

Using Java, create a GUI-based calculator that the class can use while they wait for their replacements.



Funded by the  
Erasmus+ Programme  
of the European Union

{4}

Python & Java 4 Teachers





# Process

In this project we will be making a GUI based calculator in Java. In this project we will:

- Let the user type in 2 numbers
- Allow the user to select an operator
- Perform the operation and display the result to the user

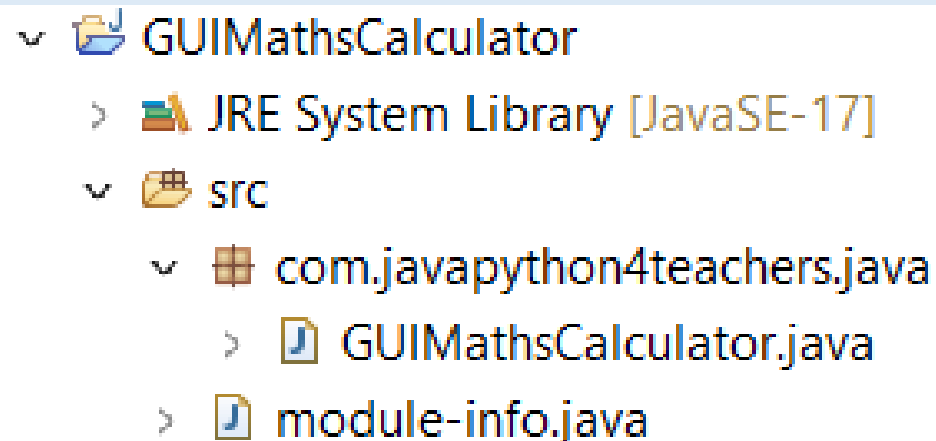
## Extension

- Develop and create your own design for the calculator

# Before You Start...

Create a new Java project. If you are not sure what steps to take, use the [Intro To Java](#) resource.

Have your IDE file system set up in the following way:



```

v [Folder Icon] GUIMathsCalculator
  > [Library Icon] JRE System Library [JavaSE-17]
  v [Folder Icon] src
    v [Folder Icon] com.javapython4teachers.java
      > [File Icon] GUIMathsCalculator.java
      > [File Icon] module-info.java
```





# Note on Errors

Java is slightly more complex than Python – with more complexity, comes more possibility! But also more potential errors.

Hovering over red underlined code often shows a list of possible solutions.

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Polygon;

import javax.swing.JFrame;
import javax.swing.SwingUtilities;

public class RoadSi
{
    public void paint()
    {
        super.paint();

        Polygon b = new Polygon();
        for (int i = 0; i < 100; i++)
        {
            b.addPoint(i, i);
        }

        g.setColor(Color.BLUE);
        g.fillPolygon(b);
        g.drawPolygon(b);
    }
}
```

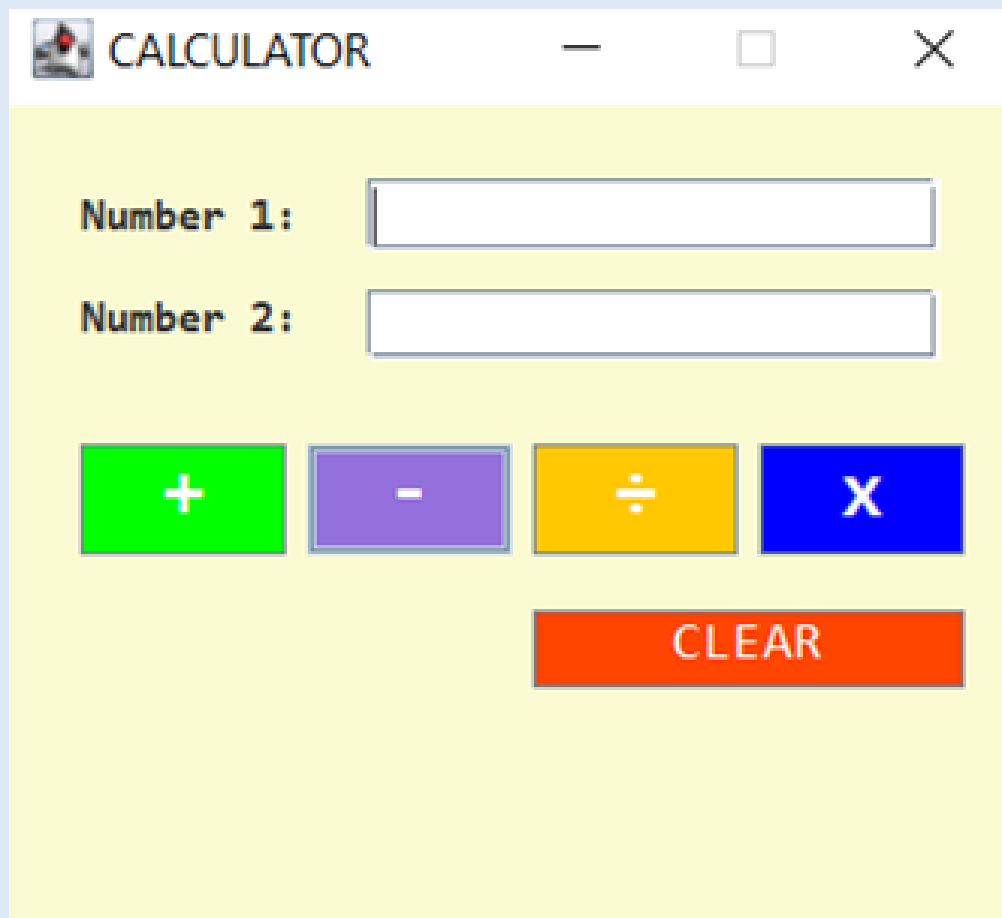
The type javax.swing.SwingUtilities is not accessible

7 quick fixes available:

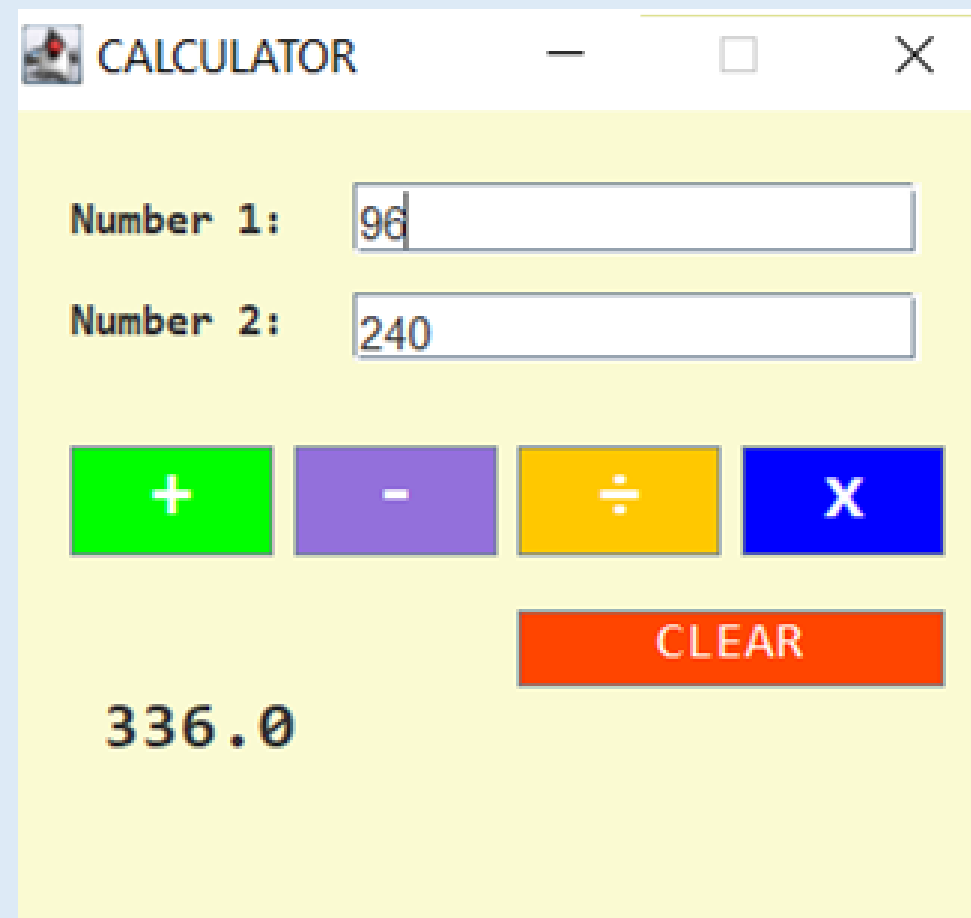
- Create class 'SwingUtilities' in package 'javax.swing'
- Create record 'SwingUtilities' in package 'javax.swing'
- Create interface 'SwingUtilities' in package 'javax.swing'
- Create annotation 'SwingUtilities' in package 'javax.swing'
- Create enum 'SwingUtilities' in package 'javax.swing'
- Add 'requires java.desktop' to module-info.java
- Fix project setup...

For example, if you get an error such as 'this package is not accessible', hover over your code and select the following.

# What it will look like...



A screenshot of a calculator application window titled "CALCULATOR". The window has a yellow background. It contains two input fields labeled "Number 1:" and "Number 2:". Below the input fields are four buttons: a green "+" button, a purple "-" button, an orange "÷" button, and a blue "X" button. At the bottom center is a red "CLEAR" button.

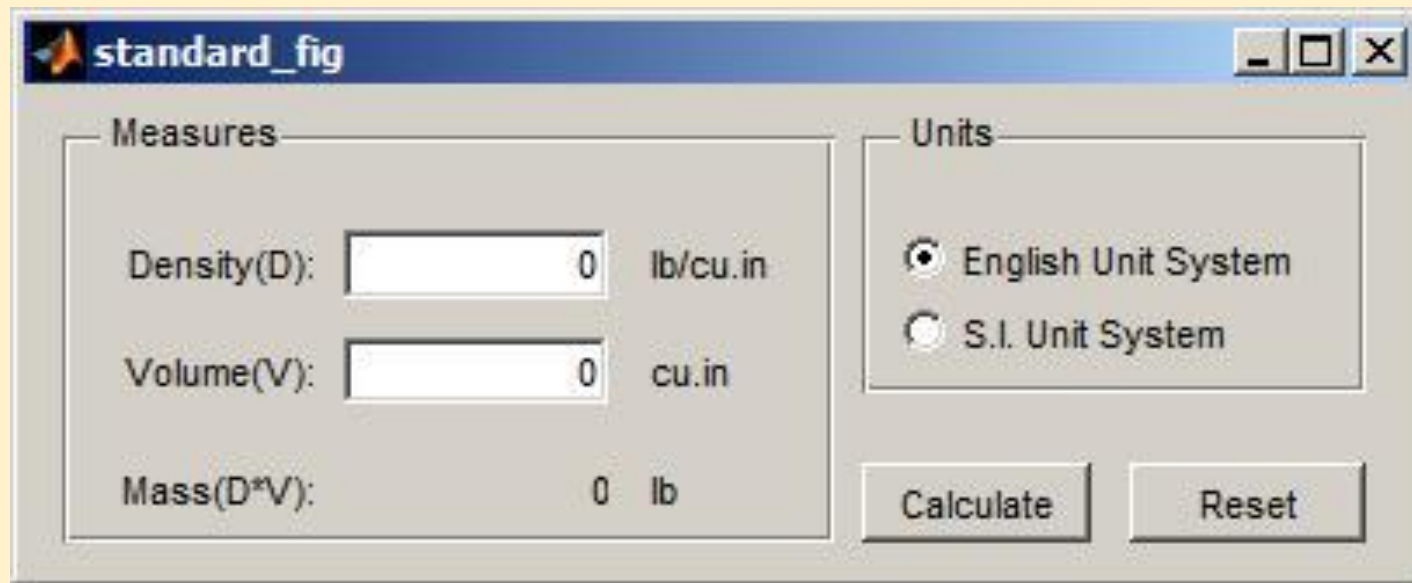


A screenshot of the same calculator application window. The "Number 1:" field contains the value "96" and the "Number 2:" field contains the value "240". The result "336.0" is displayed at the bottom of the window. The buttons and layout are the same as in the previous screenshot.



# Graphical User Interface

A GUI (or Graphical User Interface) is a form of user interface that allows users to interact with electronic devices through graphical icons and buttons.



# Step 1

## Create a GUI window

Once you have created a class, set up your code in the following way

```
public class GUIDemo {  
  
    private JFrame windowCalculator;  
  
    public static void main(String[] args) {  
        EventQueue.invokeLater(new Runnable() {  
            public void run() {  
                try {  
                    GUIDemo window = new GUIDemo();  
                    window.windowCalculator.setVisible(true);  
                } catch (Exception e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
    }  
  
    public GUIDemo() {  
        initialize();  
    }  
  
    private void initialize() {  
        windowCalculator = new JFrame();  
        windowCalculator.setResizable(false);  
        windowCalculator.setTitle("CALCULATOR");  
        windowCalculator.setBounds(100, 100, 285, 258);  
        windowCalculator.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        windowCalculator.getContentPane().setLayout(null);  
    }  
}
```

Inside of this class, we create a JFrame called 'windowCalculator' – this is the window where our calculator will sit

The main() function is where our window is first initialized.

The initialize() function is where we will design and code the functionality of our calculator.



Funded by the  
Erasmus+ Programme  
of the European Union

{4}

Python & Java 4 Teachers



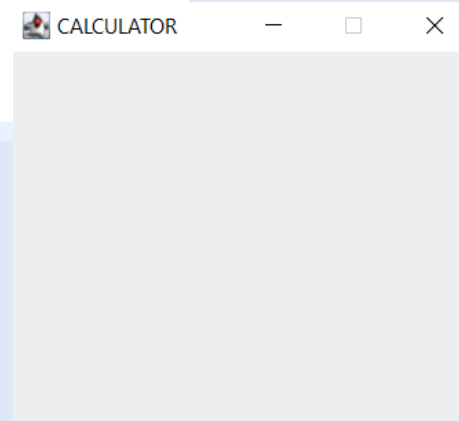
# Step 2

## Establishing our window

```
31
32 private void initialize() {
33     windowCalculator = new JFrame();
34     windowCalculator.setResizable(false);
35     windowCalculator.setTitle("CALCULATOR");
36     windowCalculator.setBounds(100, 100, 285, 258);
37     windowCalculator.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
38     windowCalculator.getContentPane().setLayout(null);
39 }
40
41 }
```

Inside here, we establish the main properties of our window, such as the size, the layout and the title.

So far all we have is a plain window - lets add some features!



# Step 3

## Adding components to our window

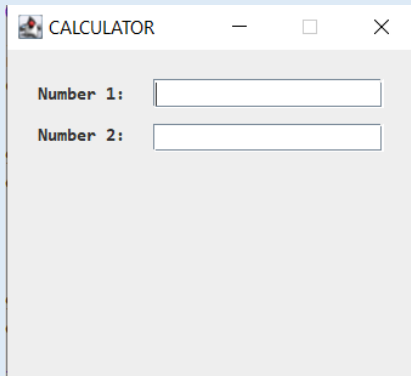
```
13 private JFrame windowCalculator;  
14 private JTextField textField1;  
15 private JTextField textField2;  
16 private JButton addition;  
17 private JButton subtraction;  
18 private JButton division;  
19 private JButton multiplication;  
20 private JButton clear;
```

Near the top of the code, we can set up the components we are going to have in our calculator.

(Don't worry about the error messages! These will be fixed as we go along)

Add the following code inside of our initialize() function.

It should look something like this!



```
46  
47 textField1 = new JTextField();  
48 textField1.setBounds(96, 20, 152, 19);  
49 windowCalculator.getContentPane().add(textField1);  
50 textField1.setColumns(10);  
51  
52 textField2 = new JTextField();  
53 textField2.setColumns(10);  
54 textField2.setBounds(96, 49, 152, 19);  
55 windowCalculator.getContentPane().add(textField2);  
56  
57 JLabel label1 = new JLabel("Number 1:");  
58 label1.setFont(new Font("Consolas", Font.BOLD, 12));  
59 label1.setBounds(20, 25, 76, 13);  
60 windowCalculator.getContentPane().add(label1);  
61  
62 JLabel label2 = new JLabel("Number 2:");  
63 label2.setFont(new Font("Consolas", Font.BOLD, 12));  
64 label2.setBounds(20, 52, 66, 13);  
65 windowCalculator.getContentPane().add(label2);  
66
```



# Step 4

## Adding more components to our window

Add these into your initialize() function! The result label will be invisible but you should see 4 operator! buttons

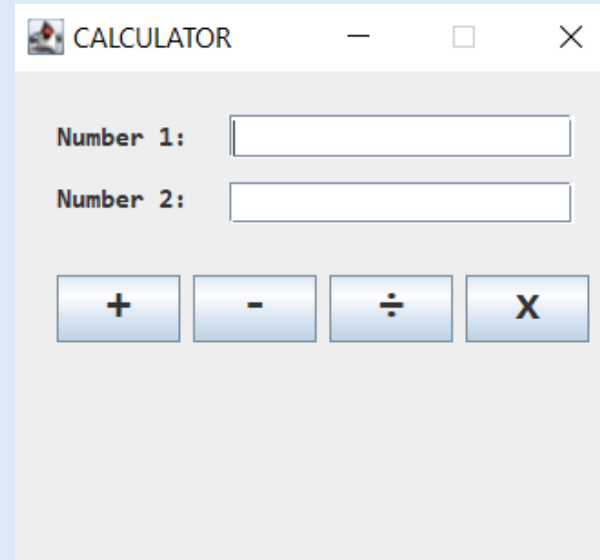
```
JLabel result = new JLabel("");
result.setFont(new Font("Consolas", Font.BOLD, 18));
result.setBounds(29, 126, 219, 85);
windowCalculator.getContentPane().add(result);

addition = new JButton("+");
addition.setFont(new Font("Consolas", Font.BOLD, 21));
addition.setBounds(20, 90, 55, 30);
windowCalculator.getContentPane().add(addition);

subtraction = new JButton("-");
subtraction.setFont(new Font("Consolas", Font.BOLD, 21));
subtraction.setBounds(80, 90, 55, 30);
windowCalculator.getContentPane().add(subtraction);

division = new JButton("\u00F7");
division.setFont(new Font("Consolas", Font.BOLD, 21));
division.setBounds(140, 90, 55, 30);
windowCalculator.getContentPane().add(division);

multiplication = new JButton("x");
multiplication.setFont(new Font("Consolas", Font.BOLD, 21));
multiplication.setBounds(200, 90, 55, 30);
windowCalculator.getContentPane().add(multiplication);
```



# Action Listeners

Action Listeners in Java are sections of code that are run in response to an event, such as a button being pressed or a window loading up.

1) A class that implements a listener such as ActionListener.

2) Register our listener with the component we want event handling used on.

3) Implements the listener method which for ActionListener is actionPerformed(ActionEvent e).

4) Do something with the component in question or its event.

```
Class OurProg implements ActionListener {  
    ...  
    public void aMethod() {  
        JButton btn = new JButton("btn");  
        btn.addActionListener(this);  
        ...  
    }  
    public void actionPerformed(ActionEvent e) {  
        btn.setText("You clicked me!");  
        ...  
    }  
}
```

When the button is pressed this method gets invoked and passed the ActionEvent object.

# Step 5

## Adding functionality to our buttons

For each button, we want an ActionListener that will get the numbers from the text fields, perform an operation on them and then display the result to the user

```
addition = new JButton("+");

addition.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        float sum = Float.parseFloat(textField1.getText()) + Float.parseFloat(textField2.getText());
        result.setText( Float.toString(sum) );
    }
});

addition.setFont(new Font("Consolas", Font.BOLD, 21));
addition.setBounds(20, 90, 55, 30);
windowCalculator.getContentPane().add(addition);
```

Inside, we create a sum variable, which stores the result of adding the number in textField1 to the number in textField2 – we change the data type to a Float so we can handle decimal numbers.

Challenge: Can you add an ActionListener to the other buttons?



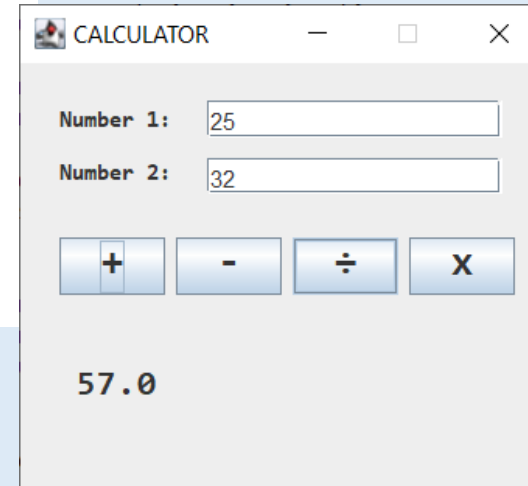
# Something like this...

```
subtraction = new JButton("-");
subtraction.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        float sum = Float.parseFloat(textField1.getText()) - Float.parseFloat(textField2.getText());
        result.setText( Float.toString(sum) );
    }
});
subtraction.setFont(new Font("Consolas", Font.BOLD, 21));
subtraction.setBounds(80, 90, 55, 30);
windowCalculator.getContentPane().add(subtraction);

division = new JButton("\u00F7");
division.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        float sum = Float.parseFloat(textField1.getText()) / Float.parseFloat(textField2.getText());
        result.setText( Float.toString(sum) );
    }
});
division.setFont(new Font("Consolas", Font.BOLD, 21));
division.setBounds(140, 90, 55, 30);
windowCalculator.getContentPane().add(division);

multiplication = new JButton("x");
multiplication.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        float sum = Float.parseFloat(textField1.getText()) * Float.parseFloat(textField2.getText());
        result.setText( Float.toString(sum) );
    }
});
multiplication.setFont(new Font("Consolas", Font.BOLD, 21));
multiplication.setBounds(200, 90, 55, 30);
windowCalculator.getContentPane().add(multiplication);
```

Challenge: Can you add a CLEAR button that will clear all information from the screen?



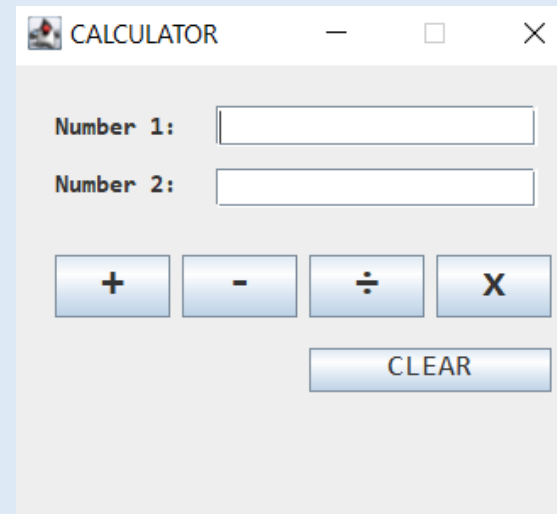


# Step 6

## Adding a clear button

To add a clear button, the ActionListener will set all textFields and labels to 'null' (this will just clear them).

```
117
118     clear = new JButton("CLEAR");
119     clear.addActionListener(new ActionListener() {
120         public void actionPerformed(ActionEvent e) {
121             result.setText(null);
122             textField1.setText(null);
123             textField2.setText(null);
124         }
125     });
126     clear.setFont(new Font("Consolas", Font.PLAIN, 15));
127     clear.setBounds(140, 134, 115, 21);
128     windowCalculator.getContentPane().add(clear);
129
130 }
```

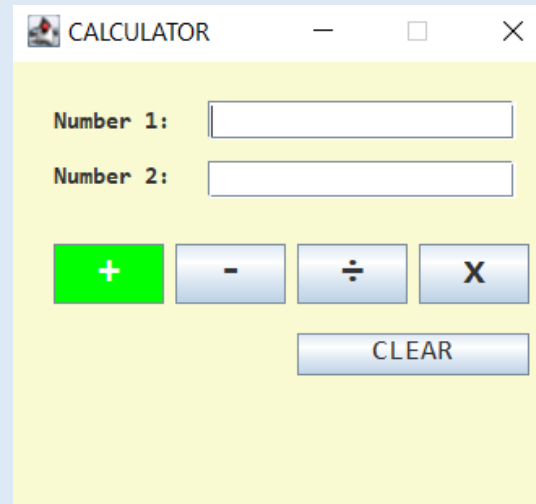


# Extension

Lets add some colour to our buttons! Using the following examples, try and personalize your calculator.

```
addition = new JButton("+");  
addition.setForeground(new Color(255, 255, 255));  
addition.setBackground(Color.GREEN);
```

```
windowCalculator.setTitle("CALCULATOR");  
windowCalculator.setBounds(100, 100, 285, 258);  
windowCalculator.getContentPane().setBackground(new Color(250, 250, 210));
```



Be creative! Feel free to customize the calculator GUI as much as you want – can you move buttons around or change the font?



# Conclusion...

## Learning Outcomes:

- ✓ Learn how to create a Java project
- ✓ Create a window
- ✓ Take user input using a GUI
- ✓ Create functional buttons
- ✓ Output information using a GUI



# Links to Everyday Life...

## At Work

Calculators are everywhere! Not only in Maths class, but on your phone and your computer

GUI-based applications are everywhere! They allow our programs to be more accessible by a wider range of users.



# Congratulations!

You have created your own GUI based calculator application using Java!



Funded by the  
Erasmus+ Programme  
of the European Union

{4}

Python & Java 4 Teachers