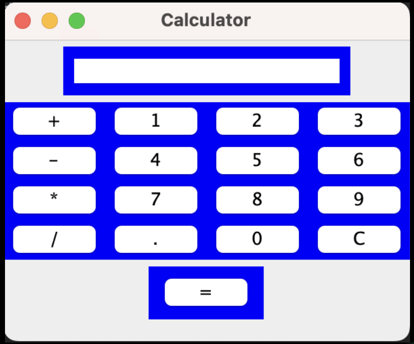
Calculator in Java

WL Hack Club



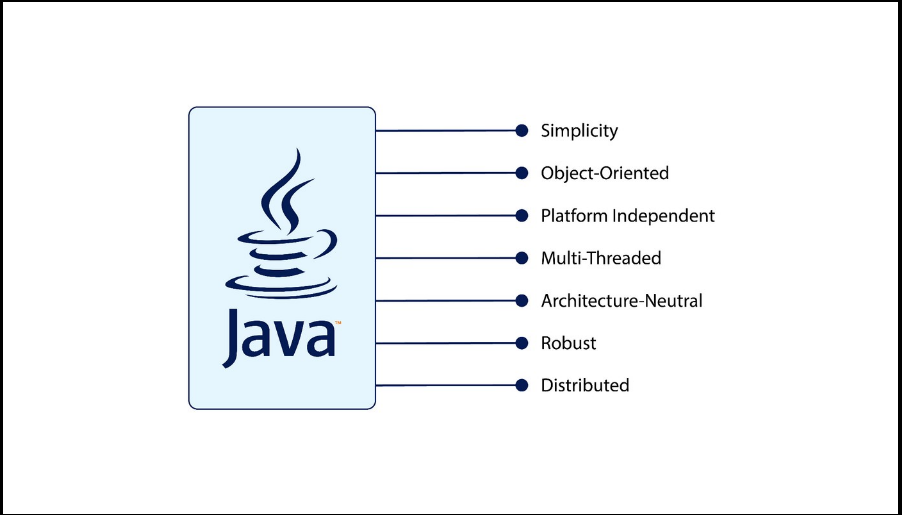
why learn java?

Java is a very popular programming language

Its versatile nature makes it a favored programming language in industry

Easy to learn and use

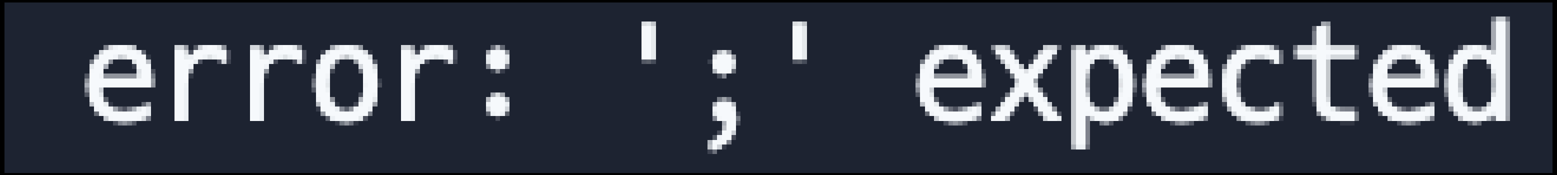
Language of AP-CSA, CS 18000 and 25100 at Purdue



semicolons

In Java you must end every line with a semicolon (unless there is a curly brace at the end)

If you forget one, you will receive an error message

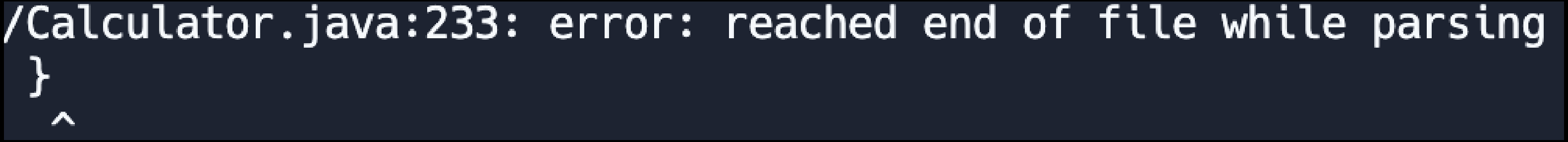


```
error: ';' expected
```

curly braces

Whenever you write and if statement, any kind of loop, or create a class or a function you must enclose the following block of code in curly braces

If you forget one you will get this error

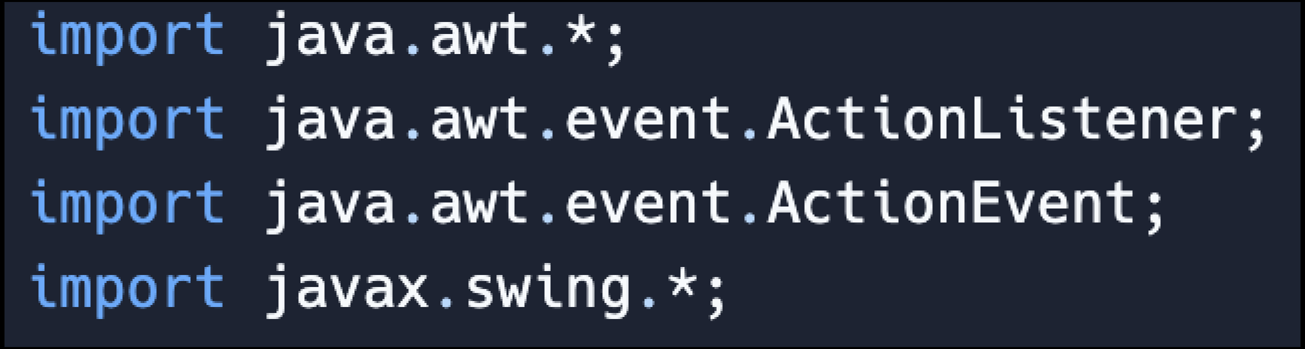


```
/Calculator.java:233: error: reached end of file while parsing
 }
  ^
```

imports

Importing declares a Java class to use in the code below the import statement

Once a Java class is declared, then the class name can be used in the code without specifying the package the class belongs to



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

swing

Swing is a lightweight Java graphical user interface (GUI) that is used to create various applications

A GUI allows users to press buttons and type in text to interact with the program

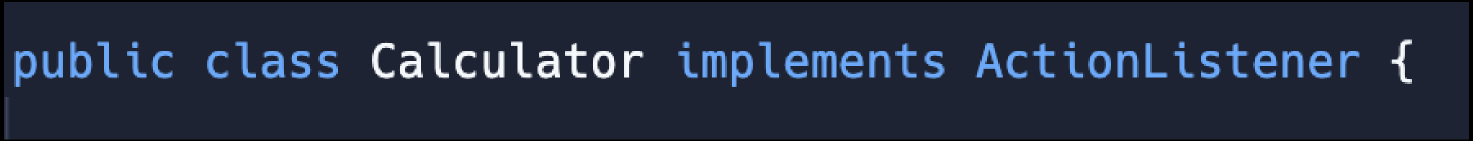
It will allow us to create buttons and text fields for our calculator

classes

A class is a template used to create objects and to define object data types and method

In Java everything is an object, so we must create a Calculator class

The class must have the same name as the file, so rename the file as well



```java
public class Calculator implements ActionListener {
```

Calculator.java

interfaces

Interfaces are a template for a type of class that makes it easier for us to implement it

We know we use an interface with the keyword implements

ActionListener allows the program to know when buttons are pressed

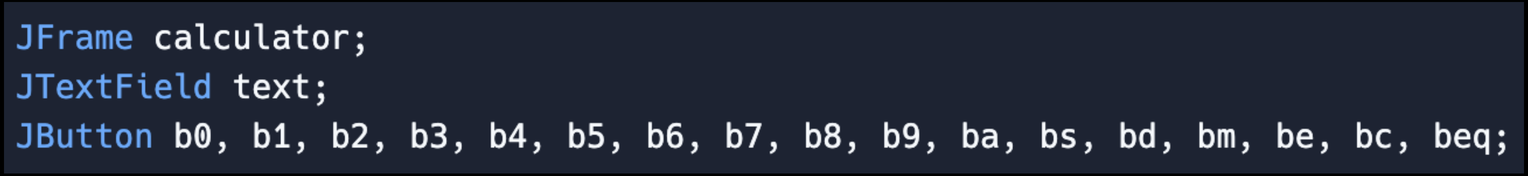
We can use this to “type” numbers when we press buttons on our calculator

We will work on implementing this later

fields

A field is a variable declared as part of a class

We want to declare the frame, text field, and buttons as a field so that we can access them anywhere in the class

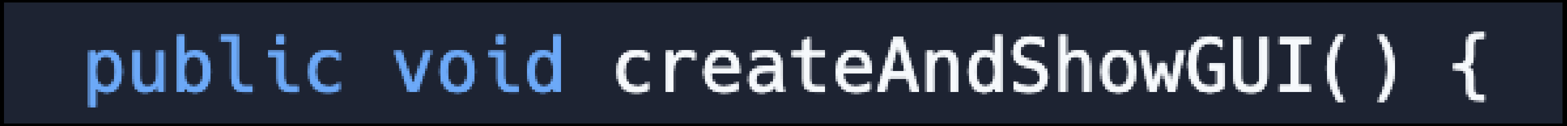


```java
JFrame calculator;
JTextField text;
JButton b0, b1, b2, b3, b4, b5, b6, b7, b8, b9, ba, bs, bd, bm, be, bc, beq;
```

functions

Functions are pieces of code that we write once and can use over and over again

We want to create this function to show our calculator on the screen



```
public void createAndShowGUI() {
```

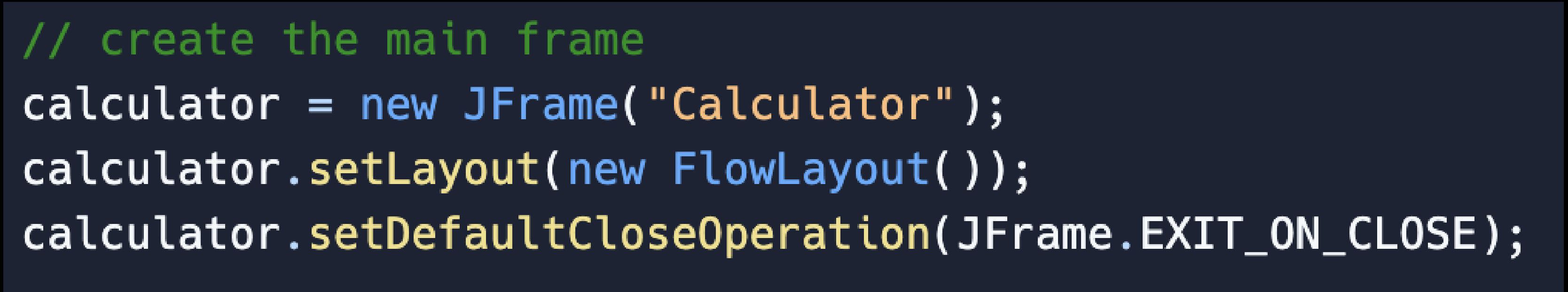
initializing the frame

A JFrame is an invisible a top-level container that provides a window on the screen

We will put other components inside this window

FlowLayout means that the arranges buttons horizontally until no more buttons fit on the same line

JFrame.EXIT_ON_CLOSE makes the program end when the frame is closed



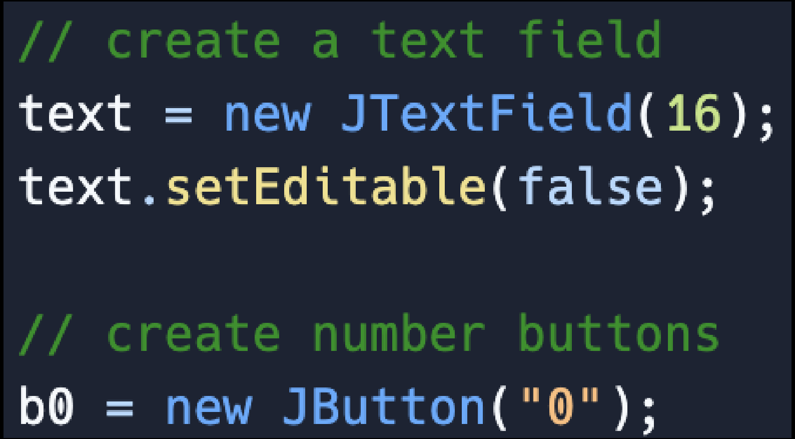
```java
// create the main frame
calculator = new JFrame("Calculator");
calculator.setLayout(new FlowLayout());
calculator.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

text fields and buttons

A JTextField is a place where text can be typed

We do not want the user typing in our text field, the only input should come from the calculator buttons, so we set editable to false

A JButton is a button that we can put text on

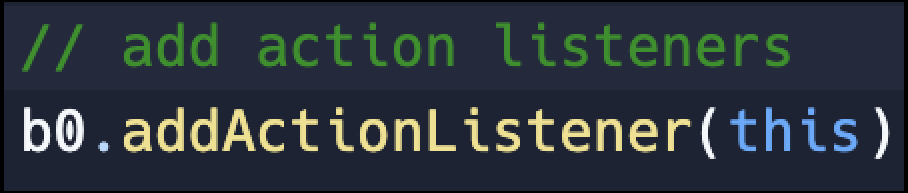


```java
// create a text field
text = new JTextField(16);
text.setEditable(false);

// create number buttons
b0 = new JButton("0");
```

action listener

An action listener allows our button to do something when it is pressed

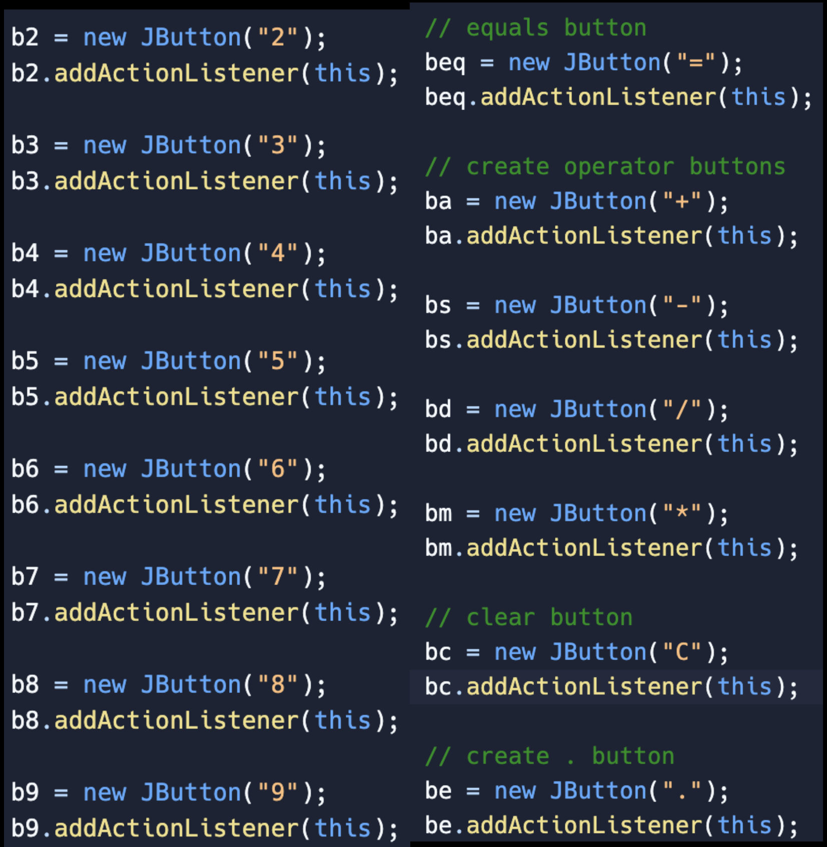
We will implement the code for what it is supposed to do later



```
// add action listeners
b0.addActionListener(this)
```

more buttons!

Now we add the rest of the buttons



```
b2 = new JButton("2");
b2.addActionListener(this);

b3 = new JButton("3");
b3.addActionListener(this);

b4 = new JButton("4");
b4.addActionListener(this);

b5 = new JButton("5");
b5.addActionListener(this);

b6 = new JButton("6");
b6.addActionListener(this);

b7 = new JButton("7");
b7.addActionListener(this);

b8 = new JButton("8");
b8.addActionListener(this);

b9 = new JButton("9");
b9.addActionListener(this);
```

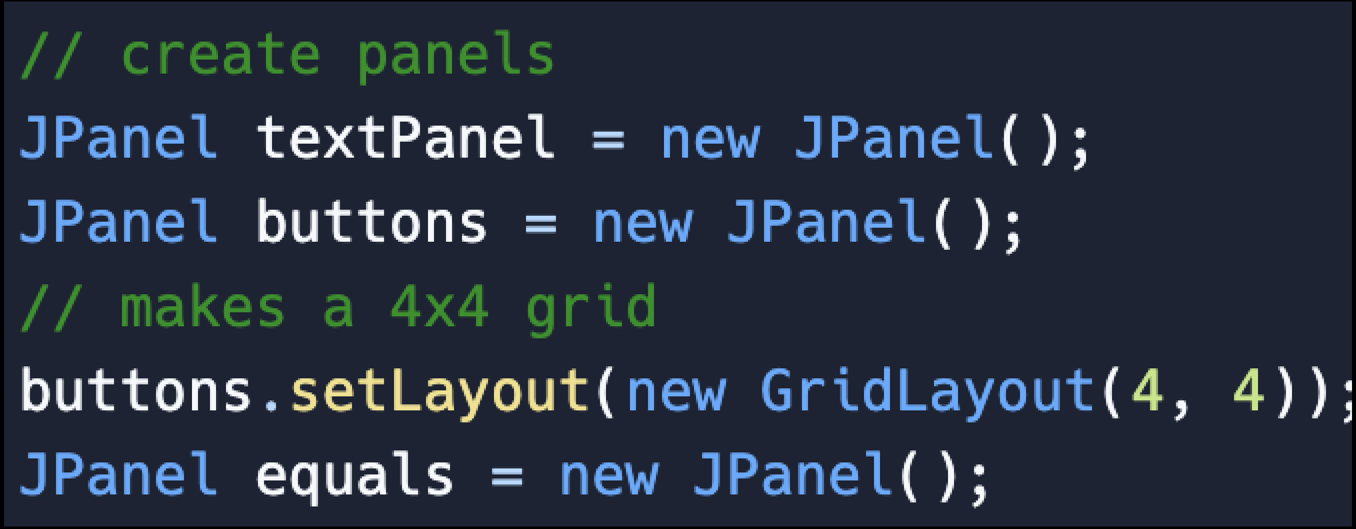
```
// equals button
beq = new JButton("=");
beq.addActionListener(this);

// create operator buttons
ba = new JButton("+");
ba.addActionListener(this);

bs = new JButton("-");
bs.addActionListener(this);

bd = new JButton("/");
bd.addActionListener(this);

bm = new JButton("*");
bm.addActionListener(this);

// clear button
bc = new JButton("C");
bc.addActionListener(this);

// create . button
be = new JButton(".");
be.addActionListener(this);
```

panels

A JPanel is a container that can store a group of components

Its main purpose is to organize our components

The GridLayout creates a 4x4 grid where we can organize 16 of our buttons



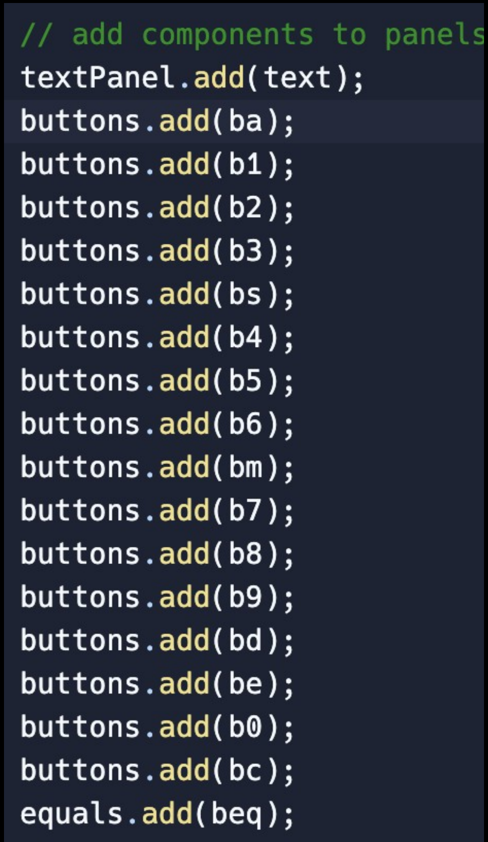
```
// create panels
JPanel textPanel = new JPanel();
JPanel buttons = new JPanel();
// makes a 4x4 grid
buttons.setLayout(new GridLayout(4, 4));
JPanel equals = new JPanel();
```

adding components to panels

Now we add our text field to textPanel,

all our buttons but eq to the buttons 4x4 grid,

and the eq button to equals



```
// add components to panels
textPanel.add(text);
buttons.add(ba);
buttons.add(b1);
buttons.add(b2);
buttons.add(b3);
buttons.add(bs);
buttons.add(b4);
buttons.add(b5);
buttons.add(b6);
buttons.add(bm);
buttons.add(b7);
buttons.add(b8);
buttons.add(b9);
buttons.add(bd);
buttons.add(be);
buttons.add(b0);
buttons.add(bc);
equals.add(beq);
```

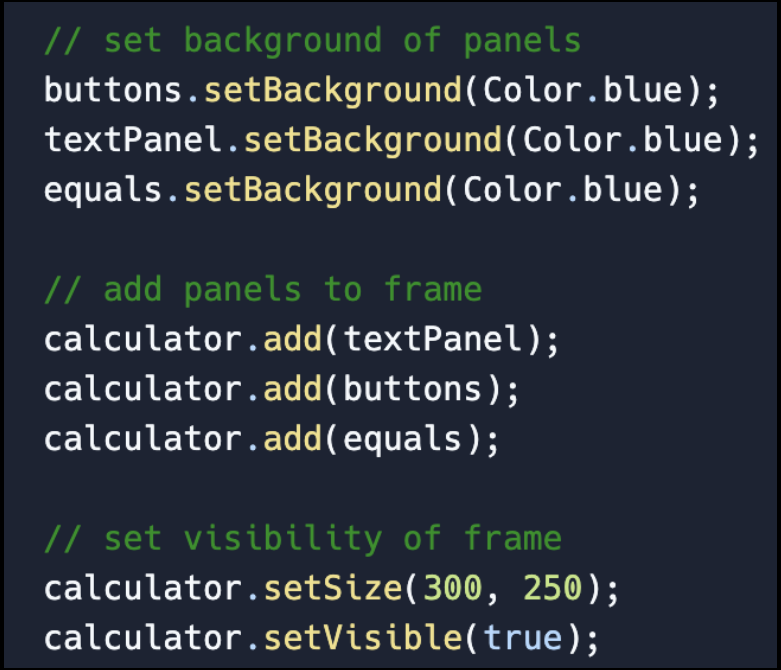
initializing panels

Set the background color of our panels

Add our panels to our original frame

Make our frame visible to the user

We are done with setting up GUI



```
// set background of panels
buttons.setBackground(Color.blue);
textPanel.setBackground(Color.blue);
equals.setBackground(Color.blue);

// add panels to frame
calculator.add(textPanel);
calculator.add(buttons);
calculator.add(equals);

// set visibility of frame
calculator.setSize(300, 250);
calculator.setVisible(true);
```

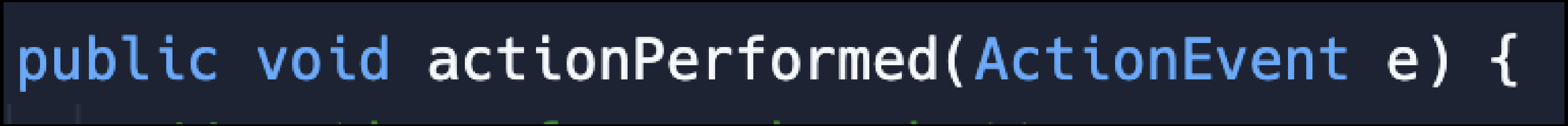
actionPerformed

Our ActionListener interface requires that we have an actionPerformed method

This method is called whenever one of our buttons is pressed

It tells the program what to do when a certain button is pressed

The ActionEvent is what event happened that called the method



```
public void actionPerformed(ActionEvent e) {
```

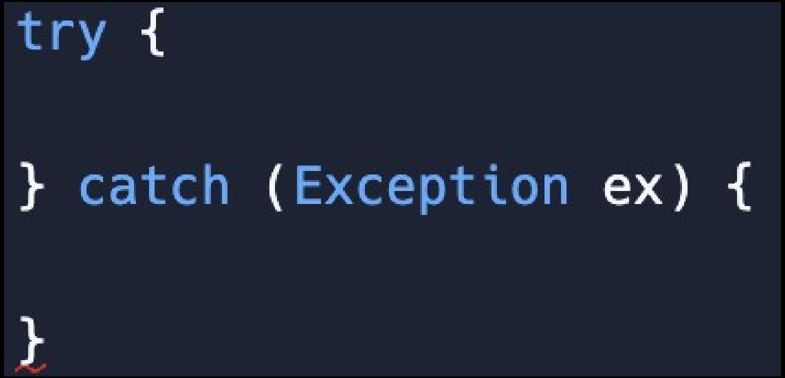
try catch

The try catch statement allows you to define a block of code to be tested for errors while it is being executed

If the code in the try block fails, the program does whatever is in the catch block

Errors will not crash the program

We will use this to prevent unexpected inputs from crashing our calculator



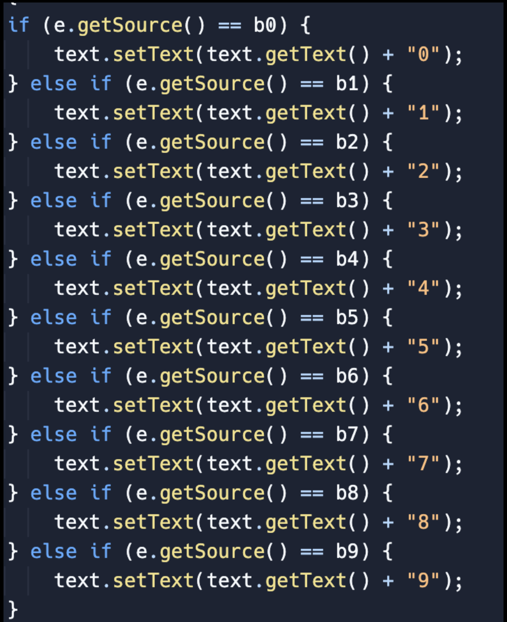
```
try {

} catch (Exception ex) {

}
```

adding actions

Add these actions for the numbers 1-10 in the try block

The if statement checks if a condition is true

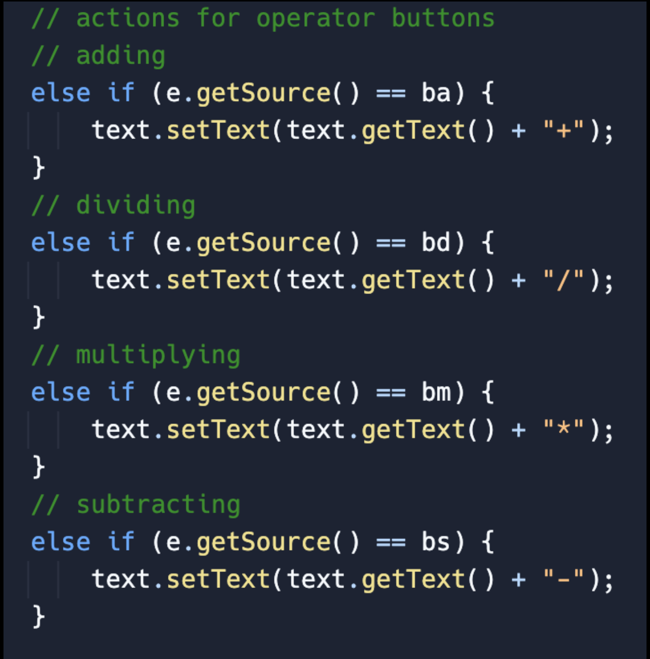
getSource() tells the program which button was pressed



```java
if (e.getSource() == b0) {
    text.setText(text.getText() + "0");
} else if (e.getSource() == b1) {
    text.setText(text.getText() + "1");
} else if (e.getSource() == b2) {
    text.setText(text.getText() + "2");
} else if (e.getSource() == b3) {
    text.setText(text.getText() + "3");
} else if (e.getSource() == b4) {
    text.setText(text.getText() + "4");
} else if (e.getSource() == b5) {
    text.setText(text.getText() + "5");
} else if (e.getSource() == b6) {
    text.setText(text.getText() + "6");
} else if (e.getSource() == b7) {
    text.setText(text.getText() + "7");
} else if (e.getSource() == b8) {
    text.setText(text.getText() + "8");
} else if (e.getSource() == b9) {
    text.setText(text.getText() + "9");
}
```

adding more actions

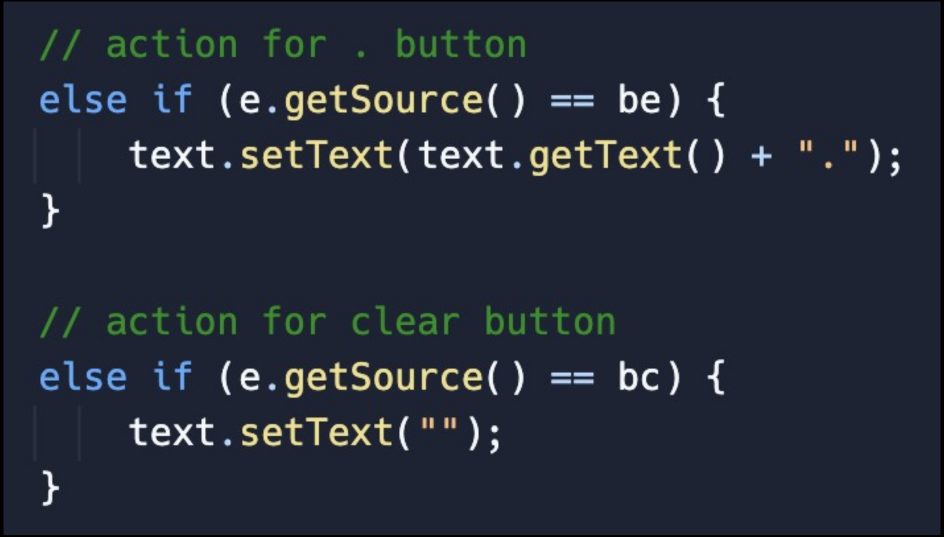
Add these actions for the operator buttons



```
// actions for operator buttons
// adding
else if (e.getSource() == ba) {
    text.setText(text.getText() + "+");
}
// dividing
else if (e.getSource() == bd) {
    text.setText(text.getText() + "/");
}
// multiplying
else if (e.getSource() == bm) {
    text.setText(text.getText() + "*");
}
// subtracting
else if (e.getSource() == bs) {
    text.setText(text.getText() + "-");
}
```

adding even more actions

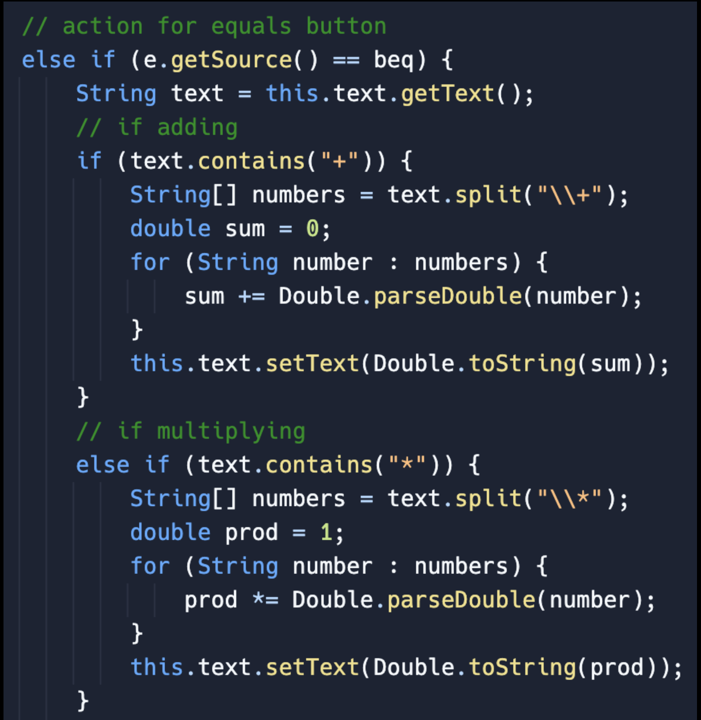
Add these actions for the decimal and clear buttons



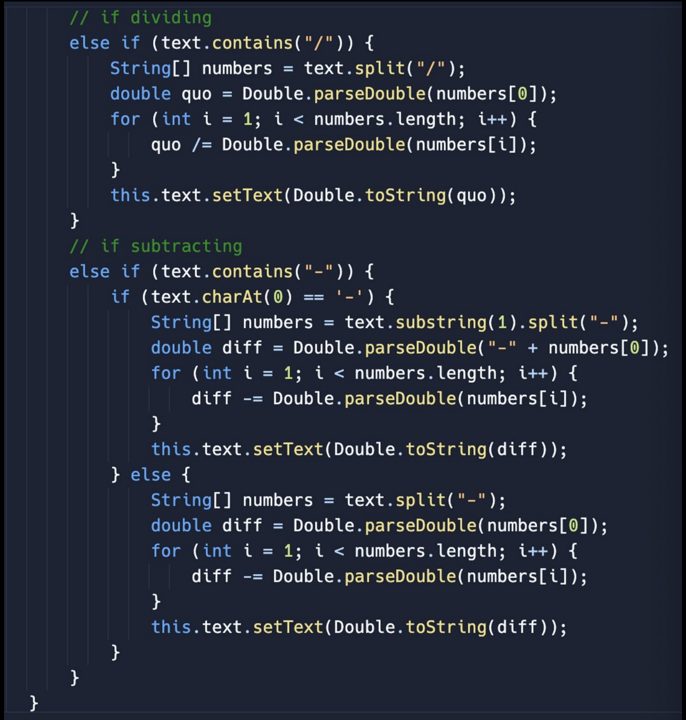
```
// action for . button
else if (e.getSource() == be) {
    text.setText(text.getText() + ".");
}

// action for clear button
else if (e.getSource() == bc) {
    text.setText("");
}
```

adding the final action

Add this action for the equals button



```java
// action for equals button
else if (e.getSource() == beq) {
    String text = this.text.getText();
    // if adding
    if (text.contains("+")) {
        String[] numbers = text.split("\\+");
        double sum = 0;
        for (String number : numbers) {
            sum += Double.parseDouble(number);
        }
        this.text.setText(Double.toString(sum));
    }
    // if multiplying
    else if (text.contains("*")) {
        String[] numbers = text.split("\\*");
        double prod = 1;
        for (String number : numbers) {
            prod *= Double.parseDouble(number);
        }
        this.text.setText(Double.toString(prod));
    }
```



```java
    // if dividing
    else if (text.contains("/")) {
        String[] numbers = text.split("/");
        double quo = Double.parseDouble(numbers[0]);
        for (int i = 1; i < numbers.length; i++) {
            quo /= Double.parseDouble(numbers[i]);
        }
        this.text.setText(Double.toString(quo));
    }
    // if subtracting
    else if (text.contains("-")) {
        if (text.charAt(0) == '-') {
            String[] numbers = text.substring(1).split("-");
            double diff = Double.parseDouble("-" + numbers[0]);
            for (int i = 1; i < numbers.length; i++) {
                diff -= Double.parseDouble(numbers[i]);
            }
            this.text.setText(Double.toString(diff));
        } else {
            String[] numbers = text.split("-");
            double diff = Double.parseDouble(numbers[0]);
            for (int i = 1; i < numbers.length; i++) {
                diff -= Double.parseDouble(numbers[i]);
            }
            this.text.setText(Double.toString(diff));
        }
    }
}
```

understanding the equals action

We first check if the math problem that the user wants to solve is addition, subtraction, multiplication, or division using an if statement

The split() function converts a string into an array

An array is a container that holds fixed number of values of a single type

We then iterate through our numbers using a for loop and do the chosen operation to them

Note: for subtraction and division we start with the first value in the string

Finally we set the text in our text field equal to the result

understanding subtraction

Subtraction is slightly different because - also means “negative”

We first check if our first number is negative, if not we do everything normally

If it is negative, we split the rest of the string

`substring(1)` allows us to get all the characters in the string but the first one, which is a negative sign

We then add the negative back and proceed as normal

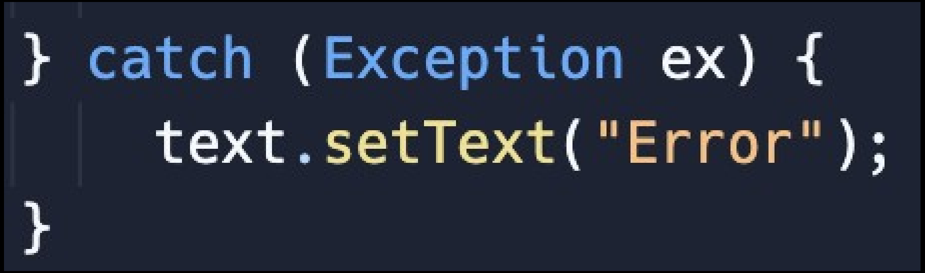
handling exceptions

Let’s assume the user entered an invalid input, such as 38+-/4

This will cause an error, or Exception

In the catch block of our try catch, we update the text field to say “Error”

This lets the user know that their input was invalid, but it does not crash our program

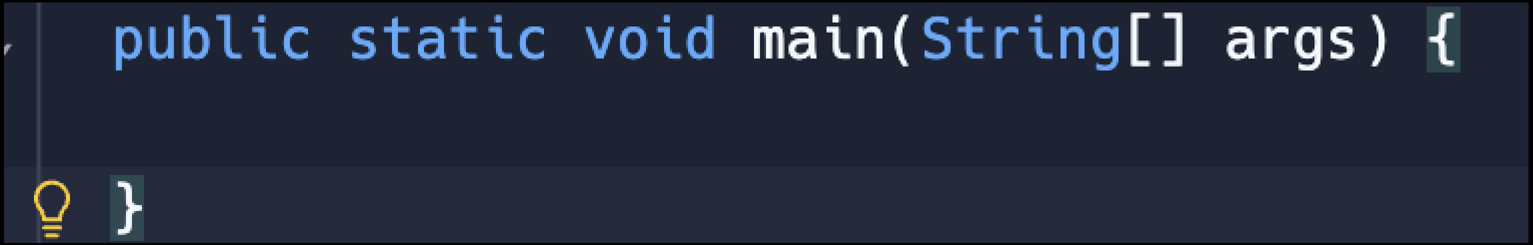


```
} catch (Exception ex) {
    text.setText("Error");
}
```

public static void main(String[] args)

In Java, the point from where the program starts its execution is the main() method

Any code that you want to be run has to go inside this method



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

}
```

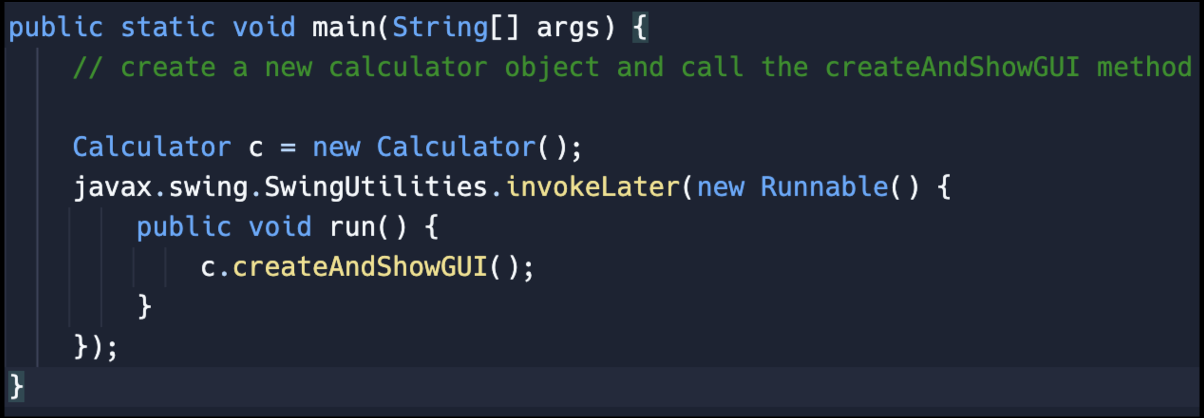
main method

In the main method we create a new calculator object and run our createAndShowGUI() method

We also want to call invokeLater()

It makes sure that only one thing (thread) handles the entire User Interface, which helps avoid many errors

It is good practice to have this



```java
public static void main(String[] args) {
    // create a new calculator object and call the createAndShowGUI method

    Calculator c = new Calculator();
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            c.createAndShowGUI();
        }
    });
}
```

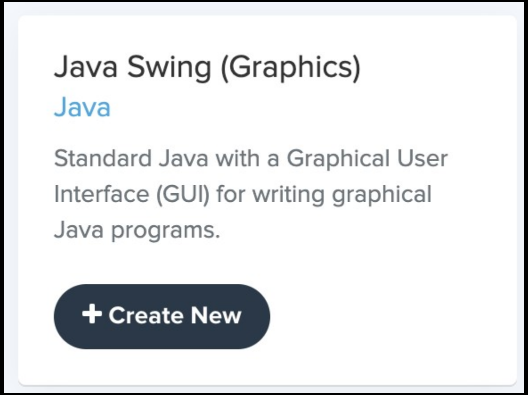
running our calculator

Unfortunately Replit is not good enough to run our calculator

Head over to CodeHS and create a Java Swing project

Copy code from replit using Ctrl-A, Ctrl-C and paste using Ctrl-A, Ctrl-V

https://github.com/WLHackClub/BasicCalculator


Java Swing (Graphics)

Java

Standard Java with a Graphical User Interface (GUI) for writing graphical Java programs.