[Creating A Calculator Using JFrame](http://www.dreamincode.net/forums/topic/321933-creating-a-calculator-using-jframe/?s=dca80c658c90cfefc5cb0eaa24d9f6cf)

In this tutorial you will learn the basics to creating a calculator with a graphical user interface. I use NetBeans for this, I highly suggest you use NetBeans or a similar IDE(Integrated Development Enviornment). You will also need Java 7 for the Nimbus look and feel. In this tutorial you will learn about using the ActionListener and the building of a GUI.  
  
To begin we will need to add a few imports.  
import java.awt.\*; - essential for visual components.  
import javax.swing.\*; - again, it's essential for the components and to set our visual feel.  
import java.awt.event.\*; - essential for handling events such as our ActionListener.  
  
I will be naming my class *Calculator*. So you'll start this class off like you would any class, public class Calculator {. Only now we need to make use of JFrame and ActionListener. You want to extend JFrame and implement ActionListener like so,  
  
public class Calculator extends JFrame implements ActionListener { }  
  
Since every program needs a main method, we can add it in right now.

public static void main(String[] arguments) {

Calculator c = new Calculator();

}

**Let's begin on our declarations now.**  
For our frame we will have five rows of components, so I make 5 panels, one for each row.  
  
JPanel[] row = new JPanel[5];  
  
Decide how many buttons you will have on your calculator. I use 19.  
  
JButton[] button = new JButton[19];  
  
With these buttons we have only declared them, we are going to need to Initialize them shortly. Each button will need a string, but instead of typing out 19 lines to initialize each one we are going to create a loop for that. So we are going to put the string values for each button in an array to use for our loop later. Note, be careful on the order, you want it setup for button[0] to equal buttonString[0].

String[] buttonString = {"7", "8", "9", "+",

"4", "5", "6", "-",

"1", "2", "3", "\*",

".", "/", "C", "√",

"+/-", "=", "0"};

Alright, now let's create arrays for the dimension widths and heights of the buttons we will create.  
int[] dimW = {300, 45, 100, 90}; - our widths will be 300, 45, 100, and 90 for the different types of components.  
int[] dimH = {35, 40}; - our heights will be 35 for the display and 40 for the buttons.  
  
Let's declare and initialize our dimensions here.  
  
Dimension displayDimension = new Dimension(dimW[0], dimH[0]); - Uses the first integer in the width, and the first integer in height.  
Dimension regularDimension = new Dimension(dimW[1], dimH[1]); - Uses the second integer in the width, and the second integer in height.  
Dimension rColumnDimension = new Dimension(dimW[2], dimH[1]); - Uses the third integer in the width, and the second integer in height.  
Dimension zeroButDimension = new Dimension(dimW[3], dimH[1]); - Uses the fourth integer in width, and the second integer in height.  
  
Now we need to declare some booleans for our functions - adding, subtracting, multiplying, and dividing. Let's use an array for that.  
  
boolean[] function = new boolean[4]; - not initialized yet  
  
We'll need some temporary doubles later for our calculations so,  
  
double[] temporary = {0, 0}; - these are now initialized.  
  
Let's create a display using JTextArea.  
  
JTextArea display = new JTextArea(1,20);  
  
and the last of our declarations, just for a more appealing effect, we will use a different font, style, and pt.  
  
Font font = new Font("Times new Roman", Font.BOLD, 14);  
  
  
Alright, let's move on to making the constructor!  
  
This is how we begin with it,

Calculator() { //Same as class name

}

Inside our constructor we start off with super("Title"); - this is required, you will recieve errors if you don't begin with this. Replace "Title" with the title of your program.  
  
Now call the method setDesign(); - you will recieve an error for this as we have not yet created that method.  
  
To set the size of the frame we will use setSize(w, h); replace the width with **380** and the height with **250**  
  
We will make it so the frame cannot be resized, which can sometimes cause the layout to get messed up.  
setResizable(false);  
  
add in setDefaultCloseOperation(EXIT\_ON\_CLOSE); to exit the frame when closed.  
  
Okay, now we think for a second, how to layout the frame. I want to use 5 rows, and have up to 5 components for each row. So I'm going to use the GridLayout manager for this. GridLayout grid = new GridLayout(5,5);with the first 5 in number of rows, and the second 5 for up to how many components we will allow in the row. to set the layout we will use *setLayout*, setLayout(grid);  
  
Moving on(we are still in the constructor), let's initialize our booleans. Instead of using 4 lines of code to do this, we can use a for loop.

for(int i = 0; i < 4; i++)

function[i] = false;

So let's examine how the loop works. We are creating an integer, i. so while *i* is less than 4, we have it set function*to false, with it incrementing one each time. Since we want function[0], function[1], function[2], and function[3] to be set equal to false, nothing higher and nothing lower. Now make sure you understand the concept of a loop, because we will be using a lot of them.  
  
Okay, so if we were to use the FlowLayout manager, the components are dropped in an area the same way words are organized on a page in English. From left to right, and from top to bottom.  
  
So I think we'll use FlowLayout for each row now. This will be how we set up row 1(which we'll call as row[0] later),  
FlowLayout f1 = new FlowLayout(FlowLayout.CENTER); - We are only going to use this for row1.  
  
For the remainder of the rows we only need to set up one new FlowLayout.  
FlowLayout f2 = new FlowLayout(FlowLayout.CENTER,1,1); - the 1's are integers for horizontal gap and vertical gap.  
  
Let's now initialize our JPanel row's so we can use them. we will create a loop for this.*

*for(int i = 0; i < 5; i++)*

*row[i] = new JPanel();*

*Now we can use our rows!  
  
So let's set the layouts we made to the rows. to do this it will be component.setLayout(layout). what we will do is,  
row[0].setLayout(f1); - which will make our first row the first flowlayout.  
  
Now since we are using the second flowlayout for the remainder of the rows we can use a loop.*

*for(int i = 1; i < 5; i++)*

*row[i].setLayout(f2);*

*Our rows now have the layout we need them to.  
  
Since there's a few of the same things we have to do for every button, let's create another loop.*

*for(int i = 0; i < 19; i++) {*

*button[i] = new JButton();*

*button[i].setText(buttonString[i]);*

*button[i].setFont(font);*

*button[i].addActionListener(this);*

*}*

*Okay now let's take a look at what this code does. We are setting our text in the buttons with the same text from our buttonString. so button[0] will have the text from buttonString[0], button[1] will have the text from buttonString[1] and so on. Here we are setting the font we declared to every button. the.addActionListener(this); will be important for later when we need to make the buttons actually work.  
  
Well that's for the buttons, but now what about the display? We have a few things to do with the display too. We have no need for a loop since we are only dealing with one component now.*

*display.setFont(font);*

*display.setEditable(false);*

*display.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);*

*So here, we are setting the font, making it so the input is not allowed by keyboard, and setting it so the input appears from right to left on the display.  
  
Now we can work on setting sizes for all of our components. If you remember correctly we created a dimension for the display, the regular buttons, the buttons on the right column, and the zero button. To set the size we will use [i]component.setPreferredSize*  
  
For our display - display.setPreferredSize(displayDimension);  
  
For the regular buttons and the right column buttons we will use loops.

for(int i = 0; i < 14; i++)

button[i].setPreferredSize(regularDimension);

for(int i = 14; i < 18; i++)

button[i].setPreferredSize(rColumnDimension);

and for zero, we just need to use - button[18].setPreferredSize(zeroButDimension);  
  
Here is where we add components to the panel, and the panels to the frame.  
  
To add the component to the panel use panel.add(component);  
I'll show you what we need,   
row[0].add(display); this adds our display to row 1.  
  
and adding the panel, add(row[0]); which adds row1.  
  
The other rows will be different because we will be using loops, Follow along for row2.

for(int i = 0; i < 4; i++)

row[1].add(button[i]);

row[1].add(button[14]);

add(row[1]);

We loop through the first 4 buttons and add them, then we add in button 15. Here are the remaining rows:

for(int i = 4; i < 8; i++)

row[2].add(button[i]);

row[2].add(button[15]);

add(row[2]);

for(int i = 8; i < 12; i++)

row[3].add(button[i]);

row[3].add(button[16]);

add(row[3]);

row[4].add(button[18]);

for(int i = 12; i < 14; i++)

row[4].add(button[i]);

row[4].add(button[17]);

add(row[4]);

Now all we need to do with our constructor is add on setVisible(true); to it and we're done  
  
Okay, so right now you are recieving 2 errors and you cannot compile yet, so let's fix those errors so you can see how your calculator looks. Outside of our constructor we can create a method called setDesign().

public final void setDesign() {

try {

UIManager.setLookAndFeel(

"com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

} catch(Exception e) {

}

}

This is the setDesign method, for more info on the look and feel look into Java LookAndFeel. - Because using the NimbusLookAndFeel we require Java 7.  
  
Let's fix this last error so we can see how it looks!  
  
below your setDesign method you can create a new one called actionPerformed, this is required when using the ActionListener.

public void actionPerformed(ActionEvent ae) {

}

You can leave it blank now, we will be adding into it later.  
  
Now if you run the program you should be able to see what our calculator will look like. The buttons do not work yet, we will get to that last.  
  
We have some methods to create now. One method for our clear button, one method for our plus minus button, one method for square root, and one method for getting our results from the functions. I will be posting the full methods and explaining them in comments.  
  
Let's start with clear().

public void clear() {

try {

display.setText(""); // Sets the display blank

for(int i = 0; i < 4; i++)

function[i] = false; // Sets the functions back to false

for(int i = 0; i < 2; i++)

temporary[i] = 0; // Sets our temporary variables back to 0

} catch(NullPointerException e) {

}

}

Now our square root method.

public void getSqrt() {

try {

double value = Math.sqrt(Double.parseDouble(display.getText())); // Create a variable for value, and use Math's square root to find value

display.setText(Double.toString(value)); // Sets display to new value

} catch(NumberFormatException e) {

}

}

Now our positive negative method.

public void getPosNeg() {

try {

double value = Double.parseDouble(display.getText()); // again we create a variable for our current value

if(value != 0) { // if the value isn't 0

value = value \* (-1); // we multiply it by -1 to get it's opposite value

display.setText(Double.toString(value)); // set the text to the new value.

}

else {

}

} catch(NumberFormatException e) {

}

}

and now our result method which is a little more complex.

public void getResult() {

double result = 0; // variable for result

temporary[1] = Double.parseDouble(display.getText()); //our second temporary number from display

String temp0 = Double.toString(temporary[0]); //necessary string for text of first temp

String temp1 = Double.toString(temporary[1]); //necessary string for text of second temp

try {

if(temp0.contains("-")) { //if first string contains -

String[] temp00 = temp0.split("-", 2); //split into two strings at -

temporary[0] = (Double.parseDouble(temp00[1]) \* -1); //puts string back in double with the real value.

}

if(temp1.contains("-")) { // same as above with second temporary

String[] temp11 = temp1.split("-", 2);

temporary[1] = (Double.parseDouble(temp11[1]) \* -1);

}

} catch(ArrayIndexOutOfBoundsException e) {

}

try {

if(function[2] == true) //we start off with multiplication obviously

result = temporary[0] \* temporary[1]; //sets result to multiplication of function

else if(function[3] == true) //now division

result = temporary[0] / temporary[1];

else if(function[0] == true) //now addition

result = temporary[0] + temporary[1];

else if(function[1] == true) //now subtraction

result = temporary[0] - temporary[1];

display.setText(Double.toString(result)); //display now has result

for(int i = 0; i < 4; i++)

function[i] = false; //set all the functions back to false

} catch(NumberFormatException e) {

}

}

Now that we have all the methods we need, all we have to do is make our buttons work. This is the working method for actionPerformed, try not to copy and paste, try to type it out on your own. it's all explained in the paragraph below.

public void actionPerformed(ActionEvent ae) {

if(ae.getSource() == button[0])

display.append("7");

if(ae.getSource() == button[1])

display.append("8");

if(ae.getSource() == button[2])

display.append("9");

if(ae.getSource() == button[3]) {

//add function[0]

temporary[0] = Double.parseDouble(display.getText());

function[0] = true;

display.setText("");

}

if(ae.getSource() == button[4])

display.append("4");

if(ae.getSource() == button[5])

display.append("5");

if(ae.getSource() == button[6])

display.append("6");

if(ae.getSource() == button[7]) {

//subtract function[1]

temporary[0] = Double.parseDouble(display.getText());

function[1] = true;

display.setText("");

}

if(ae.getSource() == button[8])

display.append("1");

if(ae.getSource() == button[9])

display.append("2");

if(ae.getSource() == button[10])

display.append("3");

if(ae.getSource() == button[11]) {

//multiply function[2]

temporary[0] = Double.parseDouble(display.getText());

function[2] = true;

display.setText("");

}

if(ae.getSource() == button[12])

display.append(".");

if(ae.getSource() == button[13]) {

//divide function[3]

temporary[0] = Double.parseDouble(display.getText());

function[3] = true;

display.setText("");

}

if(ae.getSource() == button[14])

clear();

if(ae.getSource() == button[15])

getSqrt();

if(ae.getSource() == button[16])

getPosNeg();

if(ae.getSource() == button[17])

getResult();

if(ae.getSource() == button[18])

display.append("0");

}

To find the source of the button that was pressed we use ActionEvent.getSource() == button because we called ActionEvent ae we can just use ae.getSource() instead. So our button[0] is 7. with JTextArea we use append(string) to add text in. So if our button source is button[0] (which is our 7 button)display.append("7"); This gets done with all the buttons that produce text in the display.   
  
Now moving on to our function buttons (multiply, divide, add, subtract). for the source of that button being pressed we need to set our first temporary number from the string on the display. i.e. temporary[0] = Double.parseDouble(display.getText()); now for the function we are doing we need to set that function to true. since button[3] is our add function we set function[0] equal to true, as function[0] is our adding. now we have to reset the text in the display to get our value for our second temporary before equals is hit, with a simple display.setText(""); So now do this with the other buttons that have functions, remembering that (function[0] is adding, function[1] is subtracting, function[2] is multiplying, and function[3] is dividing).  
  
Now simply enough we just need to use those methods we created for those buttons.  
The clear button will have clear(); called, the plus minus button will have getPosNeg(); called, the square root button will have getSqrt(); called, and the equals button will have getResult(); called.  
  
Your final source code should look similar to this:

import java.awt.\*;

import javax.swing.\*;

import java.awt.event.\*;

public class Calculator extends JFrame implements ActionListener {

JPanel[] row = new JPanel[5];

JButton[] button = new JButton[19];

String[] buttonString = {"7", "8", "9", "+",

"4", "5", "6", "-",

"1", "2", "3", "\*",

".", "/", "C", "√",

"+/-", "=", "0"};

int[] dimW = {300,45,100,90};

int[] dimH = {35, 40};

Dimension displayDimension = new Dimension(dimW[0], dimH[0]);

Dimension regularDimension = new Dimension(dimW[1], dimH[1]);

Dimension rColumnDimension = new Dimension(dimW[2], dimH[1]);

Dimension zeroButDimension = new Dimension(dimW[3], dimH[1]);

boolean[] function = new boolean[4];

double[] temporary = {0, 0};

JTextArea display = new JTextArea(1,20);

Font font = new Font("Times new Roman", Font.BOLD, 14);

Calculator() {

super("Calculator");

setDesign();

setSize(380, 250);

setResizable(false);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

GridLayout grid = new GridLayout(5,5);

setLayout(grid);

for(int i = 0; i < 4; i++)

function[i] = false;

FlowLayout f1 = new FlowLayout(FlowLayout.CENTER);

FlowLayout f2 = new FlowLayout(FlowLayout.CENTER,1,1);

for(int i = 0; i < 5; i++)

row[i] = new JPanel();

row[0].setLayout(f1);

for(int i = 1; i < 5; i++)

row[i].setLayout(f2);

for(int i = 0; i < 19; i++) {

button[i] = new JButton();

button[i].setText(buttonString[i]);

button[i].setFont(font);

button[i].addActionListener(this);

}

display.setFont(font);

display.setEditable(false);

display.setComponentOrientation(ComponentOrientation.RIGHT\_TO\_LEFT);

display.setPreferredSize(displayDimension);

for(int i = 0; i < 14; i++)

button[i].setPreferredSize(regularDimension);

for(int i = 14; i < 18; i++)

button[i].setPreferredSize(rColumnDimension);

button[18].setPreferredSize(zeroButDimension);

row[0].add(display);

add(row[0]);

for(int i = 0; i < 4; i++)

row[1].add(button[i]);

row[1].add(button[14]);

add(row[1]);

for(int i = 4; i < 8; i++)

row[2].add(button[i]);

row[2].add(button[15]);

add(row[2]);

for(int i = 8; i < 12; i++)

row[3].add(button[i]);

row[3].add(button[16]);

add(row[3]);

row[4].add(button[18]);

for(int i = 12; i < 14; i++)

row[4].add(button[i]);

row[4].add(button[17]);

add(row[4]);

setVisible(true);

}

public void clear() {

try {

display.setText("");

for(int i = 0; i < 4; i++)

function[i] = false;

for(int i = 0; i < 2; i++)

temporary[i] = 0;

} catch(NullPointerException e) {

}

}

public void getSqrt() {

try {

double value = Math.sqrt(Double.parseDouble(display.getText()));

display.setText(Double.toString(value));

} catch(NumberFormatException e) {

}

}

public void getPosNeg() {

try {

double value = Double.parseDouble(display.getText());

if(value != 0) {

value = value \* (-1);

display.setText(Double.toString(value));

}

else {

}

} catch(NumberFormatException e) {

}

}

public void getResult() {

double result = 0;

temporary[1] = Double.parseDouble(display.getText());

String temp0 = Double.toString(temporary[0]);

String temp1 = Double.toString(temporary[1]);

try {

if(temp0.contains("-")) {

String[] temp00 = temp0.split("-", 2);

temporary[0] = (Double.parseDouble(temp00[1]) \* -1);

}

if(temp1.contains("-")) {

String[] temp11 = temp1.split("-", 2);

temporary[1] = (Double.parseDouble(temp11[1]) \* -1);

}

} catch(ArrayIndexOutOfBoundsException e) {

}

try {

if(function[2] == true)

result = temporary[0] \* temporary[1];

else if(function[3] == true)

result = temporary[0] / temporary[1];

else if(function[0] == true)

result = temporary[0] + temporary[1];

else if(function[1] == true)

result = temporary[0] - temporary[1];

display.setText(Double.toString(result));

for(int i = 0; i < 4; i++)

function[i] = false;

} catch(NumberFormatException e) {

}

}

public final void setDesign() {

try {

UIManager.setLookAndFeel(

"com.sun.java.swing.plaf.nimbus.NimbusLookAndFeel");

} catch(Exception e) {

}

}

@Override

public void actionPerformed(ActionEvent ae) {

if(ae.getSource() == button[0])

display.append("7");

if(ae.getSource() == button[1])

display.append("8");

if(ae.getSource() == button[2])

display.append("9");

if(ae.getSource() == button[3]) {

//add function[0]

temporary[0] = Double.parseDouble(display.getText());

function[0] = true;

display.setText("");

}

if(ae.getSource() == button[4])

display.append("4");

if(ae.getSource() == button[5])

display.append("5");

if(ae.getSource() == button[6])

display.append("6");

if(ae.getSource() == button[7]) {

//subtract function[1]

temporary[0] = Double.parseDouble(display.getText());

function[1] = true;

display.setText("");

}

if(ae.getSource() == button[8])

display.append("1");

if(ae.getSource() == button[9])

display.append("2");

if(ae.getSource() == button[10])

display.append("3");

if(ae.getSource() == button[11]) {

//multiply function[2]

temporary[0] = Double.parseDouble(display.getText());

function[2] = true;

display.setText("");

}

if(ae.getSource() == button[12])

display.append(".");

if(ae.getSource() == button[13]) {

//divide function[3]

temporary[0] = Double.parseDouble(display.getText());

function[3] = true;

display.setText("");

}

if(ae.getSource() == button[14])

clear();

if(ae.getSource() == button[15])

getSqrt();

if(ae.getSource() == button[16])

getPosNeg();

if(ae.getSource() == button[17])

getResult();

if(ae.getSource() == button[18])

display.append("0");

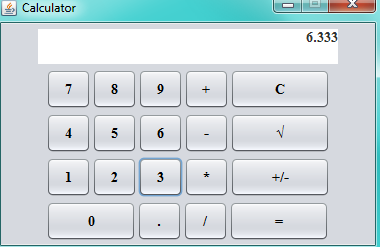
}

public static void main(String[] arguments) {

Calculator c = new Calculator();

}

}

//You should now have a fully working calculator, that should look like this  
 THE END...!