Adam Roy

CSCI 280

Project 2

1. Choosing a GUI library
   1. **Java Libraries available**

In Java there are many different libraries to choose from that act as a base to develop GUI applications in Java. Some of the libraries include AWT, Swift, SWT, SwingX, JGoodies, JavaFX and Apache Pivot.

* 1. **Assignment Scope**

In the scope of the programming assignment, we were given the option to use AWT and Swing. Swing is the path that I chose for the project as it felt much cleaner and more organized to program in.

1. Setting up the display window
   1. **Setting up the frame**

When the program first begins a frame must be created to hold all the elements that will later be added to the calculator to make it functional and professional looking. The process of creating and configuring the frame are as follows.

* 1. **Pseudocode of adding a frame**

Frame variable is initialized with a new JFrame() command.

Frame title is set with setTitle(“ ”); command (the text appears on the top of the window).

Frame size is set using setSize();

Frame resizable parameter is set to false to ensure the window size cannot be changed.

Frame visibility parameter is set to true to the window is visible to the end user.

Frame closeOperation(); is set to exit on close so the program fully exits when the end user clicks on the x button to close the program.

1. Adding elements to the frame
   1. **Initializing elements**

Each button and every text window are elements that need to be added to the frame before they can be used. All the buttons are initialized using the library function JButton. The procedure for initialization includes defining the element followed by creating defining the element as a new JButton which can be done with the JButton b0 and b0 = new JButton(“0”); commands.

* 1. **Adding the elements to the frame**

The next step is to define the bounds and position of each element that will be added to the frame. This can be achieved using the following code b0.setBounds(x, y, w, h); In this case x and y are the coordinates for the relative position of the button in the frame and w and h are the width and height of the button element. After the bounds are set additional parameters, such as setting the font or color can be added to modify the look of the buttons. The final step is to use X.add(Y); ((frame.add;(b0) in this case) where X is the name of the frame element and Y is the name of the element to be added.

1. Adding functionality to buttons
   1. **Adding action listeners to button elements**

To add functionality to each of the buttons on the calculator a way of registering button presses must be implemented. Fortunately, there is a function from the Swift library and it can be done using the addActionListener function. The structure of the function is b0.addActionListener(this); with this line the computer knows to watch for button presses on the specific button and utilize the function later on.

* 1. **Checking for button presses on numbers**

To enter the button presses to the screen and later to calculations the now active action listener translates button presses to source inputs which are utilized to determine what function should be executed. This is done with a series of if then else if statements. For example, else if the source from a button press is equal to b0 or button zero then set the textField or the number screen to the number pressed and keep listening and adding extra button presses until a function button is pressed.

* 1. **Checking for button presses on functions**

The button presses of functions is very similar to that of numbers the only difference is it sets the calculation function to a certain case 1-4 for addition, subtraction, multiplication, and division. There is a special function to implement the use of a negative toggle button which multiplies the number being calculated by -1 to toggle negative or positive numbers. Finally, the other function different from the rest is the equals function. The equals function takes the case from the previous function and uses that the determine the calculation to be performed and then returns the new value to the screen.

1. Bug testing and project finalization
   1. **Bugs that needed addressing**

One of the main bugs was the lack of a method to enter negative numbers, this was later addressed using a +/- button and now works well and has been extensively tested. Another bug was the disappearance of buttons when transferring the code from one machine to the next, as Java is a portable language it was surprising it was having this issue but was later determined to be outdated libraries on one machine.

* 1. **Finishing the final project**

Making the GUI look clean and professional required making sure the spacing between all of the buttons was exactly the same, one of the issues that is still present is the off-center nature of the number window. Another issue that could be addressed is the history window, when doing large decimal place calculations, the numbers overrun the bounds of the window and the text is clipped.