Lab Exercise 9
CS 2334

October 22, 2015

Introduction

In this lab, you will begin experimenting with Graphical User Interfaces (GUIs) in Java. GUIs are defined as a hierarchical set of graphical components (hierarchy, here, is in a *has-a* sense). At the top of hierarchy is a component that is a container of other components – in our case, this component is a **JFrame**. Other components, such as **JButton**, **JLabel**, **JTextField**, and **JRadioButton**, provide the individual pieces of the GUI with which the user interacts. **JPanel** objects are components that also act as containers for multiple components, giving us a convenient, logical way of grouping different parts of the GUI together.

Your task for the lab is to create a GUI for a simple calculator. The GUI will give the user two text boxes in which to type operands (values) and a set of radio buttons for selecting the type of operator. After performing the calculation, the GUI will display the result in a text box.

Learning Objectives

By the end of this laboratory exercise, you should be able to implement a simple GUI by:

1. Creating a window
2. Adding various graphical components to the window
3. Responding to window events in a meaningful way

Proper Academic Conduct

This lab is to be done individually. Do not look at or discuss solutions with anyone other than the instructor or the TAs. Do not copy or look at specific solutions from the net.

Preparation

1. Import the existing lab9 implementation into your eclipse workspace.

   (a) Download the lab9 implementation: `http://www.cs.ou.edu/~fagg/classes/cs2334/labs/lab9/lab9-initial.zip`

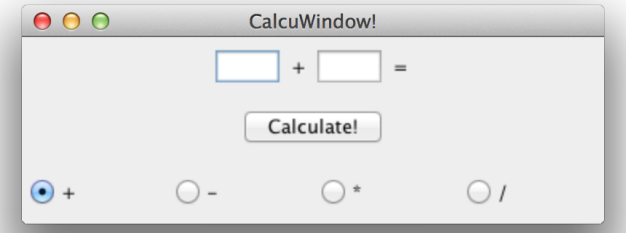
   (b) In Eclipse, select *File/Import*

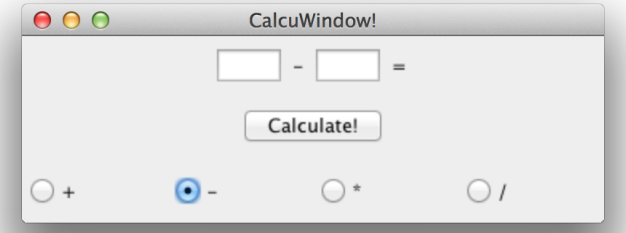
   (c) Select *General/Existing projects into workspace*. Click *Next*

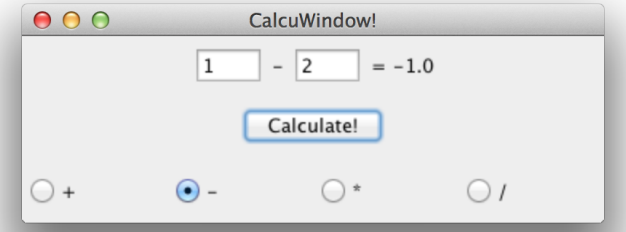
   (d) Select *Select archive file*. Browse to the lab9-initial.zip file. Click *Finish*

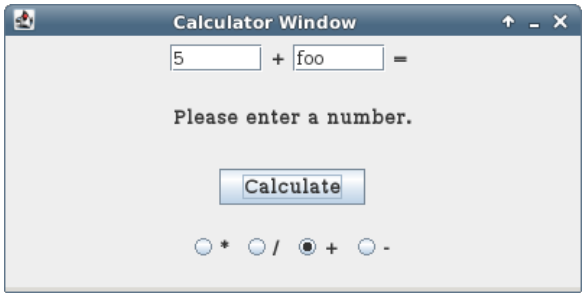
Simple Calculator

Below is an illustration of the graphical user interface for the calculator that we are creating.



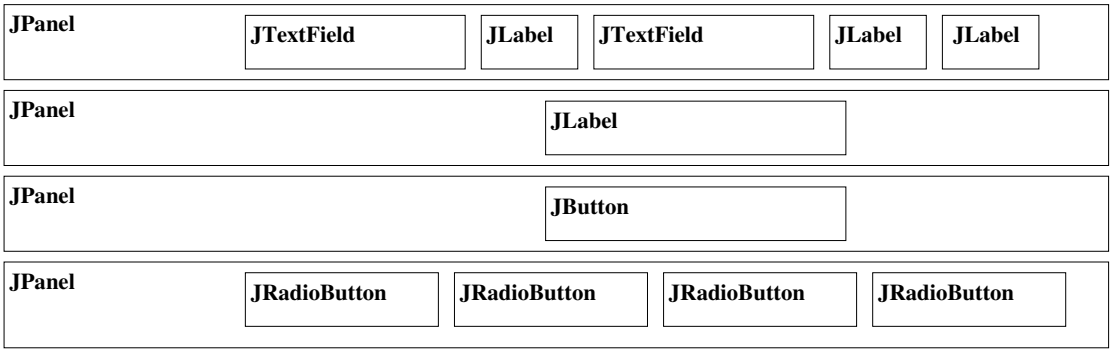






The user will enter two numbers into the two text fields. There are four operations provided to user: add(+), subtract(-), multiplication(*) and division(/). When the user clicks on the button "Calculate," based on the operation selected by the user, the answer should be displayed at the right of "=" sign. By default, the "+" sign should be selected. If the user enters an invalid number or leaves it blank, then an error message should be displayed, just below the operand boxes. When the user closes the window, the program should terminate.

Our calculator is organized using a set of 4 horizontal rows, as shown below:



The rows, each encapsulated by a **JPanel**, contain the following information:

1. The operands, operator and result
2. An error message (which can be blank)
3. A *Calculate* button
4. A set of radio buttons for operator selection

Here, we have used a **GridLayout** of dimension 4×0 to organize the four **JPanel** objects within the **JFrame**.

Use the following components in the design of your GUI:

- **JFrame** for the window
- **JPanel** for the sub-panels
- **JButton** for the button
- **JTextField** instances for providing editable input text boxes
- **JLabel** instances for displaying the result and the error message
- **ButtonGroup** to tell the Java windowing system that only one operator button can be selected at any one time
- **JRadioButton** instances for the radio buttons

Lab 9: Specific Instructions

There is only one class file, CalculatorFrame.java, that is provided in lab9-initial.zip.

1. Modify the class according to the instructions given in the comments
    - Be sure that the class name is exactly as given in the initial zip file
    - You must use the default package, meaning that the package field must be left blank
2. Implement the methods for the CalculatorFrame class
    - Do not change the name for instance variables and method name if provided in initial zip file
    - Do not add functionality to the classes beyond what has been specified
    - Don't forget to document as you go!

Hints

- Remember that your class is-a **JFrame**. This means that it will provide all of the methods that are made available by **JFrame**. Make sure to call *super()* in your constructor.
- Your constructor should create all of the GUI components, hook them together, and attach the appropriate **ActionListener**s
- Your main function must create a single instance of **CalculatorFrame** and then do nothing else
- References to the different GUI components should be included in your instance variables
- All of your inner, anonymous classes can "see" the instance variables
- See the **JRadioButton** API documentation for information about how to ask the radio button about whether it has been selected
- See the **JLabel** API documentation for information about how to change the text contained within the label

Final Steps

1. Generate Javadoc using Eclipse.
    - Select *Project/Generate Javadoc...*
    - Make sure that your project is selected, as is the CalculatorFrame
    - Select *Private* visibility
    - Use the default destination folder
    - Click *Finish*
2. Open the *lab9/doc/index.html* file using your favorite web browser or Eclipse (double clicking in the package explorer will open the web page). Check to make sure that that all of your classes are listed and that all of your documented methods have the necessary documentation.
3. If you complete the above instructions during lab, you may have your implementation checked by one of the TAs.

Submission Instructions

- All required components (source code and compiled documentation) are due at 11:59pm on Friday, October 23rd.
- Prepare your submission file:
  1. Select the project in the *Package Explorer* window.
  2. Right-click. Select *Export*
  3. Select *General/Archive File*
  4. Expand your project and verify that both the *src* and *doc* folders are selected, as well as all of their contents
  5. Enter the archive file name: *lab9.zip* (note that you may want to browse to a different destination folder)
  6. Select *Save in zip format*
  7. Click *Finish*
- Submit your zip file to the lab6 folder on D2L.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 10 points

(10) The program is properly formatted (including indentation, curly brace and semicolon locations).

(5) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 15 points

(15) The program is using proper data types and method calls.

(10) There is one error in data type or method call selection.

(5) There are two errors in data type or method call selection.

(0) There are multiple errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented.

(7) A component of one required method is missing.

(4) A method is not implemented or components are missing from two methods.

(0) Two or more required methods are not implemented or components from three or more methods are missing.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(10) The program fails one test.

(5) The program fails two tests.

(0) The program fails three or more tests.

Execution: 15 points

(15) The program executes with no errors.

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements.

(0) The method documentation is missing more than two of the required documentation elements.

Inline Documentation: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used and has the correct contents.

(5) The correct zip file name is used, but one required component is missing.

(0) An incorrect zip file name is used or more than one required component is missing.