Lab Exercise 9
CS 2334

October 19, 2017

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 the 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 binary 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 binary operator. When the user presses the *Calculate Result* button, the GUI will compute and display the result in a non-editable text field.

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.zip`

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

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

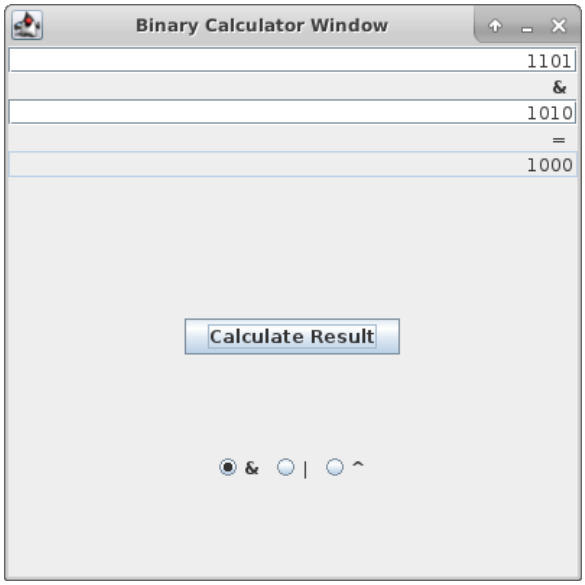
   (d) Select *Select archive file.* Browse to the lab9.zip file. Click *Finish*

Binary Calculator

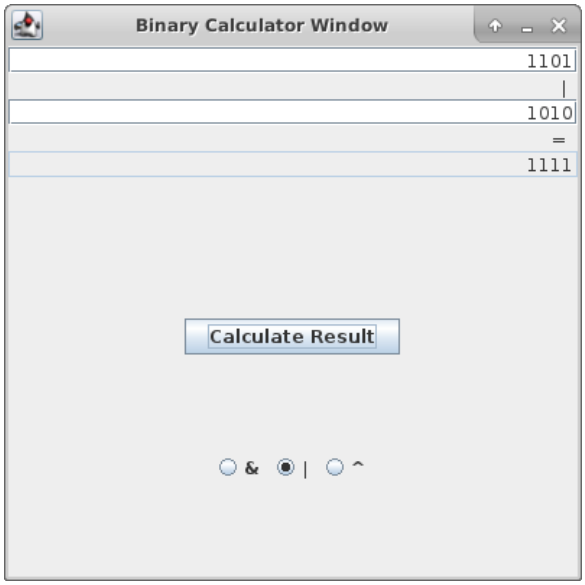
Below is an illustration of the graphical user interface for the calculator that we are creating. The user enters two binary numbers into the two text fields. There are three *bit-wise* operators provided to user: AND, OR and XOR. When the program is first started, the AND button must be selected.

When the user clicks on the button *Calculate Result,* depending on the operator selected by the user, the binary result is calculated and displayed to the non-editable result text field.

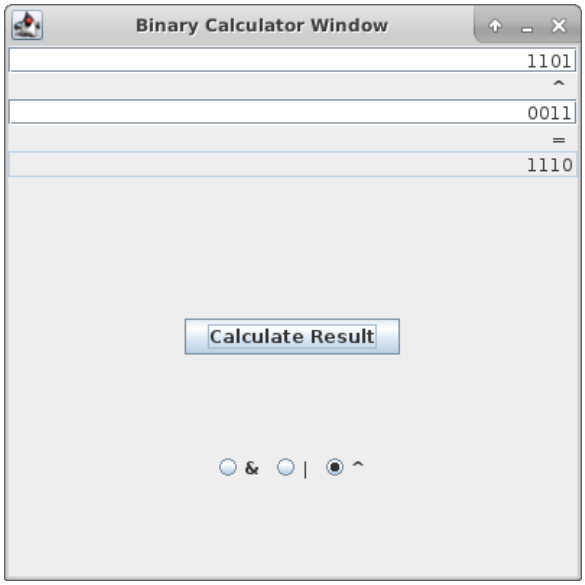
When the user enters the values *1101* and *1010* the input text fields, selects *AND (&)* and clicks *Calculate Result*, the GUI presents the result of "ANDing" the two binary numbers: *1000*. Note that with the *bit-wise* AND operator, the result for bit i is the AND of bit i from each of the operands, and that the different bit positions are independent of one-another.



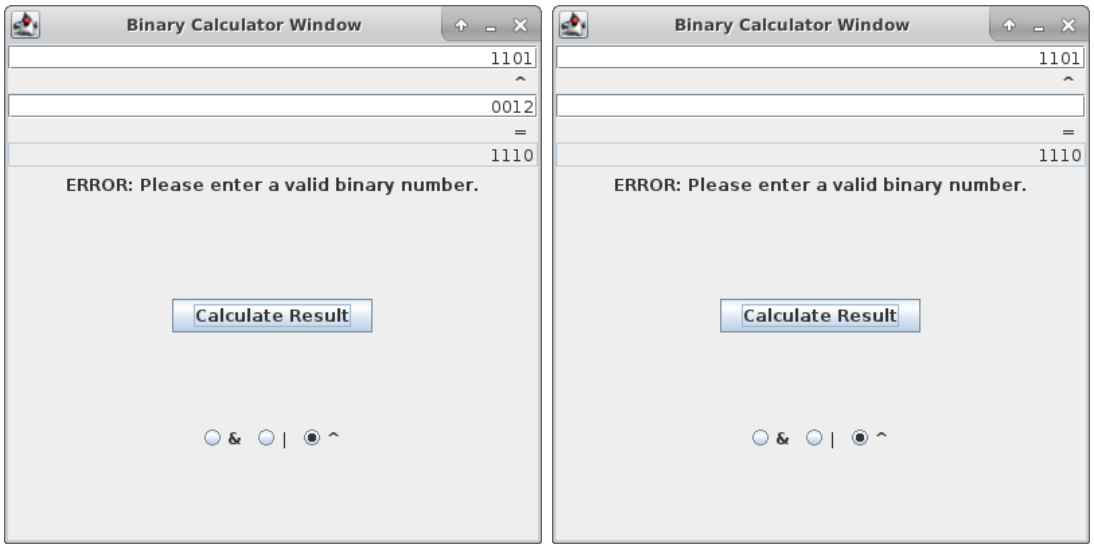
Likewise, when the user supplies the same operands (*1101* and *1010*), selects *OR (|)* and clicks *Calculate Result*, the GUI presents *1111* as the result.



Finally, when the user types the values *1101* and *0011* into the text fields, selects *XOR (^)* and clicks *Calculate Result*, the result of *1110* is displayed.



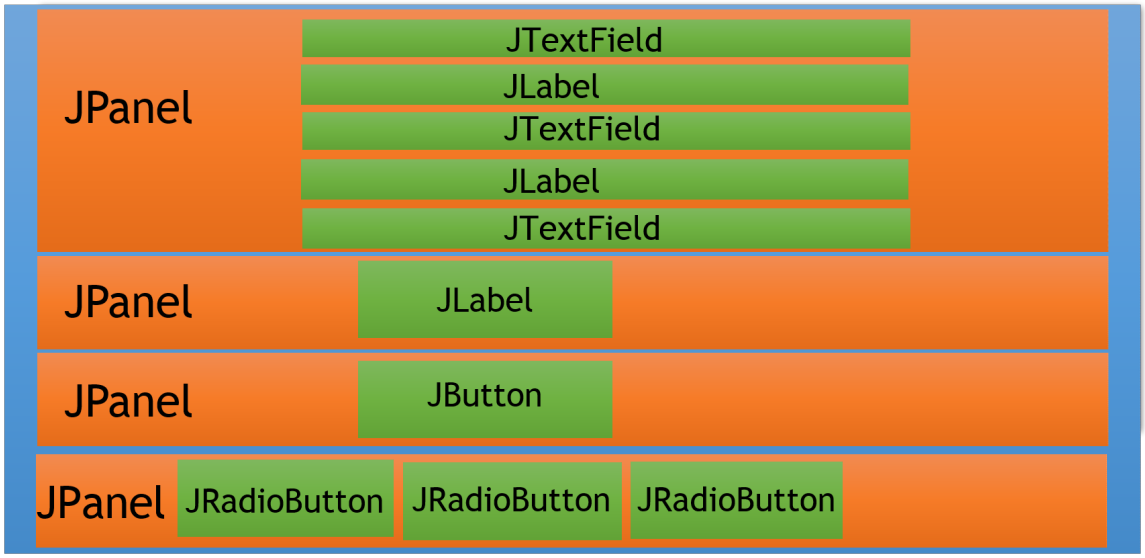
If the user enters an invalid number or leaves a text field blank, and then clicks *Calculate Result*, an error message must be displayed just below the operand box. Note that the result of the last valid computation is still displayed.



When the user closes the window, the program must terminate.

GUI Organization

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



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

1. The first operand (in a JTextField), operator (JLabel), the second operand (JTextField), the equals sign (JLabel) and the result (JTextField)
2. An error message, which can be blank (a JLabel)
3. A *Calculate Result* 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**. Also, we have used a **GridLayout** of dimension 5×0 inside the top JPanel to organize the five text fields.

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 an *editable* input text box and a *non-editable* output text box.
- **JLabel** instance for displaying 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. Note that as soon as a JRadioButton is clicked, the corresponding operator is displayed in the top JPanel.

Lab 9: Specific Instructions

There is only one class file, CalculatorFrame.java, that you must implement. The skeleton is provided in lab9.zip.

1. Modify the class according to the instructions given in the comments
    - Within the **CalculatorFrame** class, there is only the constructor and the *main()* method.
    - 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.
    - Do not change the name for instance variables and method names.
    - Do not add functionality to the classes beyond what has been specified.
    - Don't forget to document as you go!
2. Test your program by compiling and testing all possible inputs and conversions.
    - Be sure that all of the bit-wise computations are correct. Note that it is okay for your result to not have leading zeros.
    - Make sure that the error message appears when the user tries to enter an erroneous input, but disappears when correct inputs are given.
    - You do not have any unit tests to write for this lab. All of your testing will be done by interacting with the GUI. However, we will be testing your code with a set of unit tests.

Hints

- Remember that your class is-a **JFrame**. This means that it will provide all of the methods that are made available by **JFrame**. This is why we call *super()* in Calculator Frame constructor.
- Your constructor should create all of the GUI components, hook them together, and attach the appropriate **ActionListener**s.
- Your main function creates a single instance of **CalculatorFrame** and then does nothing else.
- 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.
- See the **JTextField** API documentation for information about making a text field non-editable.
- The integer bit-wise operators in Java are &, |, and ^. Note that these are different than the logical operators.
- There are many binary calculator tools available on the web that you can use to check your results.
- Look carefully at *Integer.toString()* for something that will help with displaying integers in binary. The *Integer* class also provides methods for converting binary Strings into integers.

Final Steps

1. Generate Javadoc using Eclipse.
    - Select *Project/Generate Javadoc...*
    - Make sure that your project (and all classes within it) is selected
    - 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

Before submission, finish testing your program by executing your unit tests. If your program passes all tests and your classes are covered completely by your test classes, then you are ready to attempt a submission. Here are the details:

- All required components (source code and compiled documentation) are due at 7pm on Saturday, October 21. **Submission must be done through the Web-Cat server.**

- Use the same submission process as you used in lab 1. You must submit your implementation to the *Lab 9: GUI Basics* area on the Web-Cat server.

Rubric

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

Correctness/Testing: 40 points

The Web-Cat server will grade this automatically upon submission. Your code will be compiled against a set of tests (called *Unit Tests*). These unit tests will not be visible to you, but the Web-Cat server will inform you as to which tests your code passed/failed.

Style/Coding: 25 points

The Web-Cat server will grade this automatically upon submission. Every violation of the *Program Formatting* standard described in Lab 1 will result in a subtraction of a small number of points (usually two points). Looking at your submission report on the Web-Cat server, you will be able to see a notation for each violation that describes the nature of the problem and the number of subtracted points.

Design/Readability: 35 points

This element will be assessed by a grader (typically sometime after the lab deadline). Any *errors* in your program will be noted in the code stored on the Web-Cat server, and two points will be deducted for each. Possible errors include:

- Non-descriptive or inappropriate project- or method-level documentation (up to 10 points)
- Missing or inappropriate inline documentation (2 points per violation; up to 10 points)
- Inappropriate choice of variable or method names (2 points per violation; up to 10 points)
- Inefficient implementation of an algorithm (minor errors: 2 points each; up to 10 points)
- Incorrect implementation of an algorithm (minor errors: 2 points each; up to 10 points)

If you do not submit compiled Javadoc for your project, 5 points will be deducted from this part of your score.

Note that the grader may also give *warnings* or other feedback. Although no points will be deducted, the issues should be addressed in future submissions(where points may be deducted).

Bonus: up to 5 points

You will earn one bonus point for every two hours that your assignment is submitted early.

Penalties: up to 100 points

You will lose ten points for every minute that your assignment is submitted late. For a submission to be considered *on time*, it must arrive at the server by the designated minute (and zero seconds). For a deadline of 9:00, a submission that arrives at 9:00:01 is considered late (in this context, it is one minute late).

After 15 submissions to Web-Cat, you will be penalized one point for every additional submission.

For labs, the server will continue to accept submissions for three days after the deadline. In these cases, you will still have the benefit of the automatic feedback. However, beyond ten minutes late, you will receive a score of zero.

The grader will make their best effort to select the submission that yields the highest score.