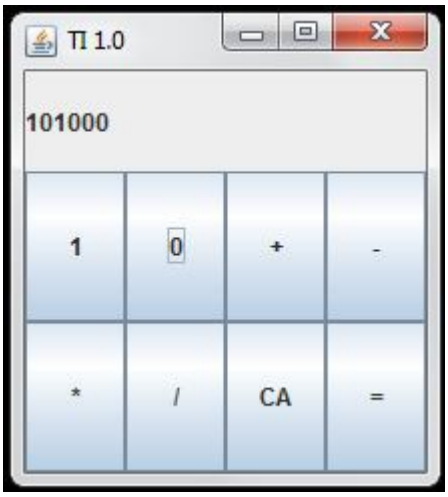
CS 1331 Homework 6

Due Friday October 19th, 2012 8:00PM

Binary Calculator



Task: Make a calculator application that works with binary numbers!

Deliverables: You must submit all of the following source files for credit:

1. **BinaryCalculator.java**
2. **CalculatorScreenPanel.java**
3. **HW6Driver.java**

If you write any additional classes for the assignment, include them. If a source file you submit does not compile, you will receive no credit for that file (so make sure to practice safe submission).

Prior to starting this homework, we recommend you read up on binary (base 2) numbers. If you need help, be sure to ask questions on Piazza!

HW6Driver.java:

This is just a driver class. It contains a main method that generates your GUI. It should create a JFrame, and add a CalculatorScreenPanel to it. Be sure you set up your frame's default close operation properly.

CalculatorScreenPanel.java:

This class represents the user interface for a calculator. It must have the following features:

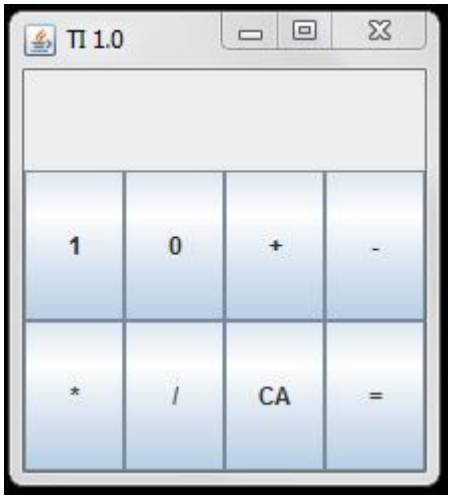
- A display for the number being inputted and for the output of the operation. (This could be a label or a drawing)

- You will have 8 buttons on your calculator face, arranged in a (4 x 2) grid as shown above. The operations work just as you'd expect. First, the user inputs a string of 1's and 0's. Then, the user selects what operation to perform. Finally, the user inputs another string, presses "=", and the result of the operation is displayed on-screen. If the user begins typing again, the screen is cleared and the process begins anew. If the user presses "=" when no operation has been selected yet, then nothing happens.
  - 1 : A "1" input (add a 1 to the end of the current input)
  - 0 : A "0" input (add a 0 to the end of the current input)
  - + : Addition (add the next input to the current)
  - - : Subtraction (subtract the next input from the current)
  - * : Multiplication (multiply the current input by the next input)
  - / : Division (divide the current input by the next input)
  - = : Equals (signals the last input of an operation is complete, displays result)
  - CA : Clears the current input

- How you handle the controller logic is up to you. It would be helpful to think about the kind of *states* your calculator can be in (such as waiting for input, etc), and think of ways to keep track of which state your calculator is *currently* in. The calculator must be able to work properly when the user keeps adding operations. For example, the sequence:
  - "0101 + 0111 * 010 - 110 =" should result in 10010 being displayed. Remember, the user is inputting these operations one at a time. It should work just like a normal calculator. This seems daunting at first, but here's a big hint: if the user requests a new operation, try to store the results of the operation you just finished, and use it as the first input of the next operation.

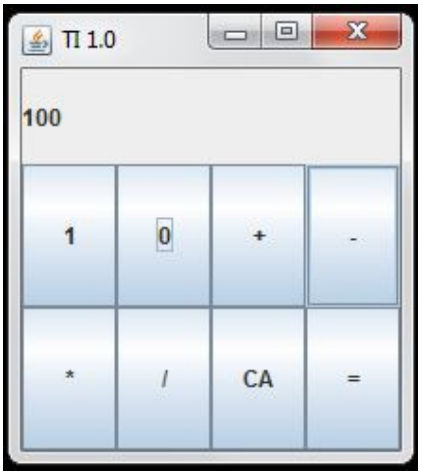
Example GUI:

Our calculator starts off with a blank input field.

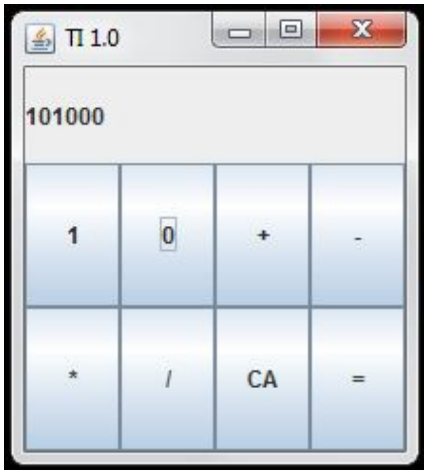


Pressing various combinations of zeros and ones sets bits in the first input string.

Pressing the desired operator clears the display and allows us to put in the second input string. In this case we're multiplying 10 by 4.



Upon pressing equals, the output of the desired operation is displayed.



BinaryCalculator.java:

This class represents a binary calculator (the brains of your calculator application). It takes in input as unsigned 31-bit binary numbers (this means that you won't have to worry about negative numbers). It must have the following features:

- It should be able to handle at least addition, subtraction, multiplication and division. Each operation it can perform should be available as a static method in the class that takes in two Strings (the two inputs as unsigned binary strings), and returns a String (the result, in binary).

Remember that binary numbers are just sums of powers of two, where the power is the index of the bit that is turned on (the zero index is on the right). For example:

10010 = 2^4 + 2^1 = 18
01101 = 2^3 + 2^2 + 2^0 = 13

One strategy for handling the operations is to convert the binary numbers into decimal ints and perform standard math on them. If you choose to do this, keep in mind that you will not receive credit for using **Integer.parseInt(String, 2)** to convert the binary strings to integers (you'll have to do it yourself). You are allowed to use **Math.pow(int, int)**. To convert integers to binary strings (if you choose to do it this way), you are allowed to use **Integer.toBinaryString(int)**.

Additionally: Proper encapsulation and documentation are required. Use class and method javadocs with the appropriate tags.

Extra Credit:

We are offering three possible extra credits here. You may only choose 2. Please remember to mention which ones you have done in your javadocs!

- (+5) Use a JTabbedPane to switch between views of the of the output as either binary, decimal, or hex.
- (+5) If invalid input is given to the calculator, a dialog box should appear saying so, for

example in the case of division by zero or of a syntactic error.

- (+5) Implement these 4 bit-wise operations: AND, OR, NOT, XOR. There must be a corresponding button in the GUI for each one. [info]