CISC 3120 Fall 2012 Homework 2

Instructions

- This is the first homework assignment for CISC 3120.
- **It is due by midnight on Wednesday September 12th** and you will submit this online — instructions below.
- **Follow these submission instructions:**
  1. Create a ZIP archive from the `.java` files that make up your homework.
  2. Go to:

     `http://agents.sci.brooklyn.cuny.edu/parsons/cisc3120/`
  3. Login and submit your ZIP archive.
     You should have received your username and password from Prof Parsons.
  4. You can submit as many times as you like before the deadline—only the final version of the homework will be graded.
  5. Failure to follow these instructions will result in points being taken away from your grade. The number of points will be in proportion to the extent to which you did not follow instructions. . . (since it can make it a lot harder for me to grade your work — grrrr!)

1 Source

You should start from versions of the two Java objects:

`SimpleCalc3.java`

`CalcComponent3.java`

which you can find on the homework page. These give you an initial version of the calculator which is a bit more sophisticated than the version you were working on in class, but any work you did in class can just be transferred across.

2 First build your project

Once again you you need to use your Eclispe skills to set up a project:

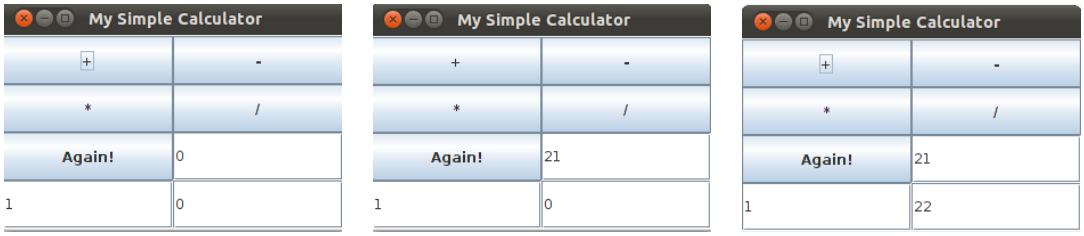
1. Create a new Java project:
   File > New > Java Project
2. Add the files:
   `SimpleCalc3.java`
   `CalcComponent3.java`
   to the project.
   A not-very-elegant way to do this is to create new file:
   File > New > File
   and then paste in the code.

3. Run it:

   Run > Run

   will try to run the project that contains the file that is displayed in the main part of the Eclipse window.

4. You should see a window that looks like the lefthand image in:



   The lower left text field has a randomly generated value.

5. You can enter a value in the uppermost of the text fields, and in the middle image above, and hitting the + button adds the two numbers, putting the result in the final text field, as in the rightmost image above.

3 Extending `addUp`

The first thing to do is to extend the functionality of the `addUp` function:

1. Make `addUp` work with `double` arguments.

   Note that this will involve quite extensive changes, not just to the function, but also to some of the attributes of the `CalcComponent3` class.

2. Change the display to allow the reading and writing of `double` values.

   You will need to use the class `Double` rather than `Integer` and increase the number of columns in the text fields (and I'm sure do some other things that I have not listed).

3. Now extend `addUp` so that it works with any number of arguments.

4 Add more functions

1. At the moment, the calculator only adds up numbers,providing functionality for the + key. Extend it so that it provides functionality for the other keys,* / - (that is multiplies, divides and subtracts).

   The functions you write should, like the modified `addUp`, handle `double` arguments (you don't need to make them handle any number of arguments).

2. Now add some additional functions (again with `double` arguments):

   - sqrt (square root)
   - sin (sine)
   - max (maximum)
   - min (minimum)

   Note that square root and sine only take one argument.

5 Alter the interface

At this point the interface starts looking a bit unsatisfactory. (It doesn't look like either my old high school calculator, or the one on my computer). Those calculators just have one field. The calculators use that one field to display input and show output. So, modify your calculator in the following way:

1. Reduce the number of textfields to 1.
2. Add an = ("equals") button.
3. Alter your code for the + button so that when the user hits +, the textfield is cleared, but the number that was in it is stored.
4. The relevant calculation is only carried out once you have hit the = button.
5. For this simple version you can assume that you will have to deal with two possible sequences of operations of the following form:

   24 + 3 =

   So, you expect a number, then, another number and then equals. When you get to the = you print the answer in the text field.
6. Now extend the calculator to work like this for all the binary operators.
7. Now consider the sqrt operation. For this you have a sequence like:

   25 sqrt

   So you expect a number and then the operator, and when you get the operator you print the answer in the text field.
8. Now extend the calculator to work like this for sine as well.

6 Challenge problem

The challenge problem is to modify the calculator further.

1. My computer has a calculator that behaves as follows.
2. The user can type in numbers into the text field as before, and pressing the operation keys (+, -, etc) just adds those symbols to the text field.
3. When the user hits =, the computation in the text field is carried out.
4. The challenge is to make your calculator work like this.
5. Writing the various symbols to the text field is easy.
6. To do the computation, you will have to parse the string you get from the text field.

   Probably the easiest way to do this is to use the `String` method `split`.

   This takes a string and from it generates an array of strings broken up in a way you specify.

If you complete the challenge you will get extra credit beyond what you would get for completing the previous section (you only need to do one).

If you complete the challenge, you should (a) document in the comments exactly what kinds of sequence of operation the calculator will handle and (b) change the text at the top of the window to read "Challenge Calculator", so I know what I am dealing with.

7 Document it and hand it in

1. Change the comments in the files to reflect the modifications you made.
2. In particular, you should have a comment that explains how to use your program.
3. This comment should describe any limitations on your code, and any places where your program deviates from the instructions above.
4. Submit your work to me, following the instructions above.

8 How I will mark your work

1. You will get credit for each of the things listed above that you have implemented in your program and which work as I have described them.
2. However, your program MUST COMPILE AND RUN for you to get any credit at all.

   (At this point in your computer science career it is unacceptable to submit programs that don't compile and run).
3. Thus it is better for you to submit a program that works and contains less functionality than it is to submit a program with more functionality that does not work.
4. You can get full credit for this homework **without** answering the challenge problem.