Lab 07: Testing And Debugging

Assigned: Thursday, March 28, 2019
Due: Wednesday, April 3 at 11:30pm

1 Objective

In this assignment you will learn how to test your own code and how to use the debugger. You will also see and work with while loops and examples of other concepts from the course.

2 Background

In this assignment, we will be working with a basic calculator program. Unlike all previous assignments, however, you do not need to write the class – I have already done so! Unfortunately, I make no promises that I have done so correctly. In fact, my program is riddled with errors. You are going to need to find the errors and fix them to make the program work correctly.

Experienced programmers know that they are going to make mistakes. (Lots of mistakes!) To make a mistake is human, but to be unaware that you had made a mistake until it causes major problems for your customer, however, is unacceptable. So programmers spend a lot of time writing tests to discover any mistakes they might have made.

In this course, you have been able to count on AutoLab finding your mistakes, because I have written all of the testing code. But this is a very unusual situation. When you are in the "real world" your client does not provide you with tests. They simply start using the program that you deliver in their business, and blame you if anything goes wrong. So we are going to start learning how to write your own tests.

Knowing that a part of your program does not work correctly is half the battle. Figuring out why it does not work correctly is another entire process. Eclipse includes a tool called the debugger to help with this, and I'll be demonstrating how to use it soon after the lab begins.

Because the purpose of this lab is for you to test the program on your own rather than relying on AutoLab to find the problems, you may only submit to Autolab for "free" 10 times. Every submission after the 10th will cost you 1 point.

3 Setup

Create a new Java Project in Eclipse named `Lab07`. Do not create a new class. Instead, download the `Calculator.java` file from D2L, and then move it from your `Downloads` directory to the `Lab07/src` directory within your workspace. Then refresh Eclipse so that it sees the file.

4 Assignment

Read through the class that I have written, and especially read the Javadoc comments for each method, since they will tell you what the method should do. To give you a sense of how the program

is supposed to eventually work, I am including here a sample session:

```
Enter +, -, *, or /: -
Minuend: 10
Subtrahend: 3
10.0 - 3.0 = 7.0
Would you like to do another calculation?: yes
Enter +, -, *, or /: 8
Please try again.
Enter +, -, *, or /: *
Multiplicand: 0.2
Multiplier: 0.04
0.2 * 0.04 = 0.008
Would you like to do another calculation?: nope
That was not an acceptable answer.
Would you like to do another calculation?: no
Goodbye!
```

You have some work to do to get this working:

1. (10 points) [not autograded] Look at the `testCalculate` method. This contains test cases that can be used to observe whether or not the `calculate` method is working correctly. I have written two test cases so far. In each one, I call the method with a certain set of parameters, record what I expect the answer to be, and then print them both. You should write at least three more test cases: one for multiplication, one for division, and finally one that you expect to cause an `IllegalArgumentException` to be thrown. There won't be any expected result for that last one, because if the exception does get thrown, the program immediately exits.

2. (10 points) Run your test cases and see if they work. If so, you should see a line printed for each test case except the last one, and the answers should be either identical or extremely close due to rounding error. Then the very last test case should have caused the exception to be thrown, which means that nothing will get printed for it.

   If your test cases did not work correctly, then you have discovered that there is at least one mistake in the `calculate` method. Find and fix any mistakes that you can, either by using the debugger or just carefully reading and thinking about the body of the method. When you think you have fixed everything, run your test cases again and see if they pass. If not, you have more work to do.

   You should not submit to AutoLab until all of your tests are passing. If AutoLab does not give you full credit for this problem even though all of your tests are passing, that means that your tests are incomplete.

3. (10 points) [not autograded] Create a `testGetOperandName` method similar to `testCalculate`. Read the Javadoc comment for the `getOperandName` method to determine what it should do. You should end up with 9 test cases: 4 for the first operand to an addition / substraction

/ multiplication / division problem, 4 for the second operand to an addition / subtraction / multiplication / division problem, 1 for a bad operator name.

4. (10 points) Change the `main` method to call `testGetOperandName` instead of `testCalculate`. Run your test cases. If they do not all pass, find and fix the errors in the `getOperandName` method. Repeat until all errors have been fixed.

   You should not submit to AutoLab until all of your tests are passing. If AutoLab does not give you full credit for this problem even though all of your tests are passing, that means that your tests are incomplete.

5. (15 points) Because the `getOperation` method (and all the others you haven't gotten to you) involve reading input, you aren't going to be able to write good testing code for it using only what you know so far. Instead, you need to just change the main method to call the method and test it by running it many times, typing in different things to make sure that it responds correctly to all of them.

   Thoroughly test the `getOperation` method and fix any errors that you find in it. **You should not submit to AutoLab until you have done enough testing to be fairly sure that it works correctly.**

6. (15 points) Thoroughly test the `getDouble` method and fix any errors that you find in it. **You should not submit to AutoLab until you have done enough testing to be fairly sure that it works correctly.**

7. (15 points) Thoroughly test the `runProgram` method and fix any errors that you find in it. **You should not submit to AutoLab until you have done enough testing to be fairly sure that it works correctly.**

8. (15 points) AutoLab will now automatically check for Javadoc comments and good style. (It can't find all types of bad style, though, so I will still be checking some things.) Ensure that you are making it happy and that you think you will make me happy as well.

5 Submitting Your Work

Make sure that you have submitted your final version to the `Lab 07: Testing And Debugging` AutoLab page, and that (unless you are satisfied with a lower score) you have 80/80 autograded points.

You do not need to submit your work through D2L – I will look at whatever your last AutoLab submission is.