CS162 Summer 2012
Lab 1: Debugging

I. The Buggy ExtremeValueFinder

Download the BuggyExtremeValueFinder.java file. This is a buggy version of the ExtremeValueFinder class used in Lab1. The following steps will help you get familiar with the different operations in the debugger. Answer the questions that are in boldface below. Remember that if you make a mistake, you can always restart the debugger.

a. Put a breakpoint at the first line of the main function. This is the line that looks like:
```
double vals[] = {-4.0, -1.0, …};
```

b. Start the debugger

c. Hit the **Step Over** button (Hotkey = F6) once.
Question 1: What variables show up in the Variables Window?

d. Hit the **Step Over** button again. You should now be on the line that calls *findExtremeValues*. **Step Into** (Hotkey = F5) this function call, which will bring you into the *findExtremeValues* function. In the Variables Window, you should see "this". To the left of "this", there will be a symbol that looks like a plus sign inside a box. Click on this symbol, which expands "this".
Question 2: What does expanding "this" do?

e. Use the Step Over button to execute the code line-by-line in the debugger. Inspect the Variables window to help you answer the following questions:
Question 3: for each iteration of the for loop, what are the values of *minSoFar* and *maxSoFar* ?
Question 4: How many times did the loop execute?
Question 5: Now try to fix the BuggyExtremeValueFinder class. What bugs did you have to fix to get the code working correctly?

II. The Message Scrambler

This is a more complex Java program that you will debug. You probably won't understand everything that is going on, but using the debugger can help you understand the program a little better. As before, follow the instructions below and answer the questions.

a. Put a breakpoint on the first line in the main function. This is the line that declares the String message.

b. Start the debugger. Hit **Step Over** 3 times. This puts you on the call to *encrypt()*. Hit **Step Into** to go inside the *encrypt()* function.

c. Once you are inside the *encrypt()* function, take a look at the **Call Stack**.
Question 1: What is on the call stack below the line Thread[main] (Suspended)?

d. Hit *Step Into* once more. This puts you inside the *initTempBuffer()* function.
Question 2: What changed on the call stack? Why?

e. Hit **Step Over** once.
Question 3: What is the value of tempBuffer?

f. Put Breakpoint #2 on the line inside *initTempBuffer()* which returns tempBuffer. It might take a while to go through the for loop. To skip executing each line in the for loop, we'll hit the **Resume** button (this is a yellow rectangle and a green triangle) which stops us when we hit the next breakpoint (which is the one you just set).
Question 4: What is the value of tempBuffer now?

g. You will now delete Breakpoint #2. Right click in the margin and select "Toggle Breakpoint". Then hit **Step Over**.
Question 5: What changed on the call stack? Why?

h. We're going to skip the rest of the encryption. Put your mouse cursor on the line in the main function that looks like:
`String decryptedMessage = ms.decrypt(encryptedMessage);`
We could set a breakpoint here and hit the resume button. However, we could use a different technique. Click on the left mouse button while your cursor is on this line of code. Then click with the right mouse button and select **Run to Line**. This will advance the debugger to the line you just clicked on.
Question 6: What is the value of the encryptedMessage variable?

i. Now hit Step Over once.
Question 7: What is the value of the decryptedMessage variable?