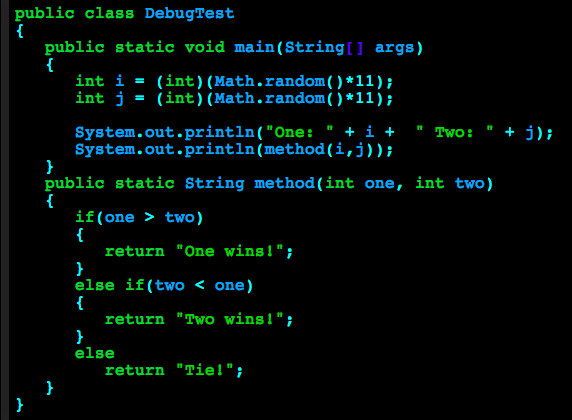
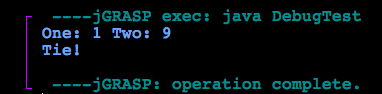
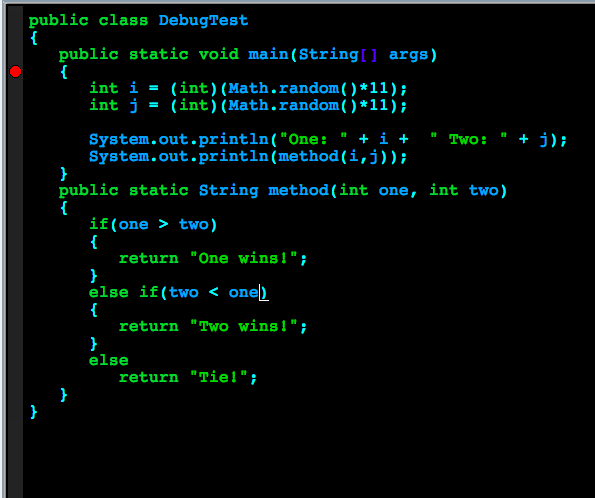
The Case of the Messed Up Program:



I made a simple program to see which player wins based on generating a random number. However, when I run it, player two never wins, even when two is greater than one:

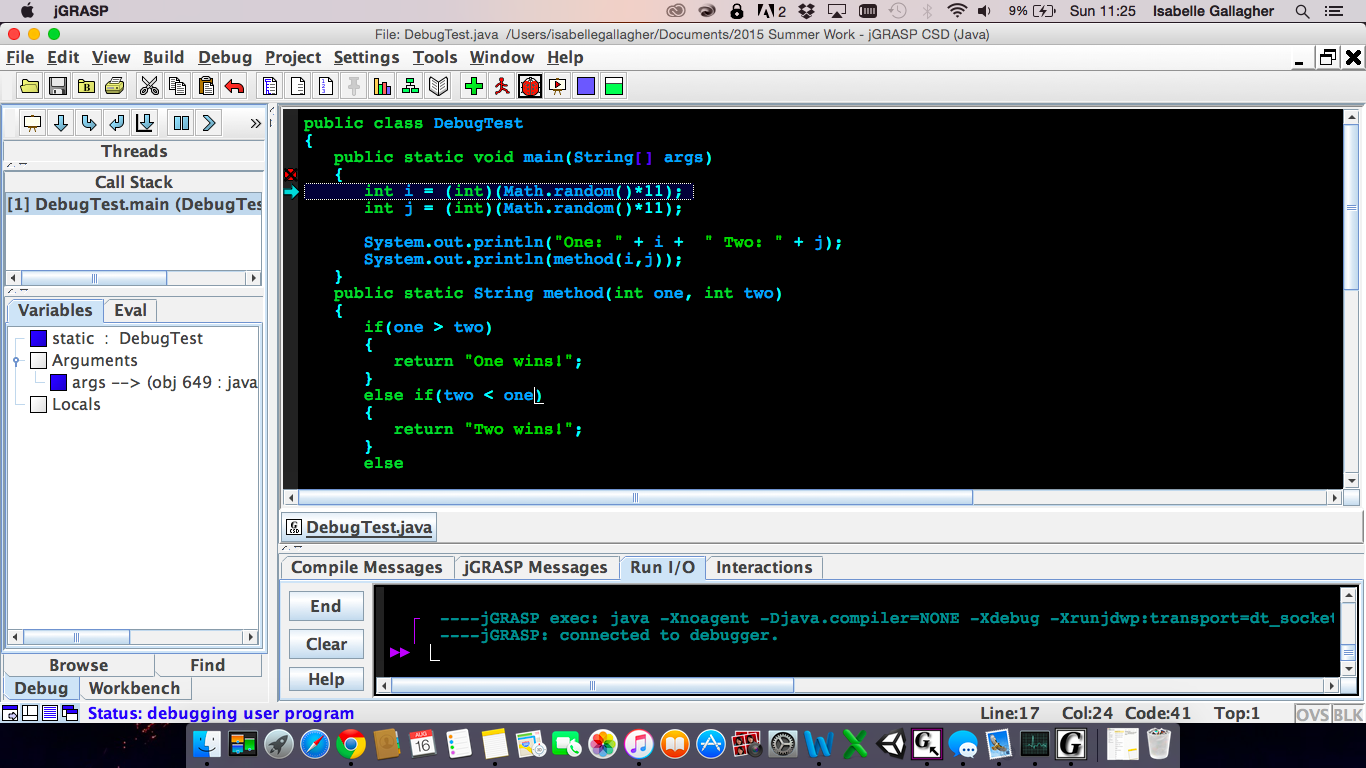


I can’t figure out what’s wrong, so I decide to debug. First, I have to place a breakpoint. This tells the computer where to stop the program. I have no idea what’s wrong, so I place the breakpoint at the top of the program by clicking on the left side:

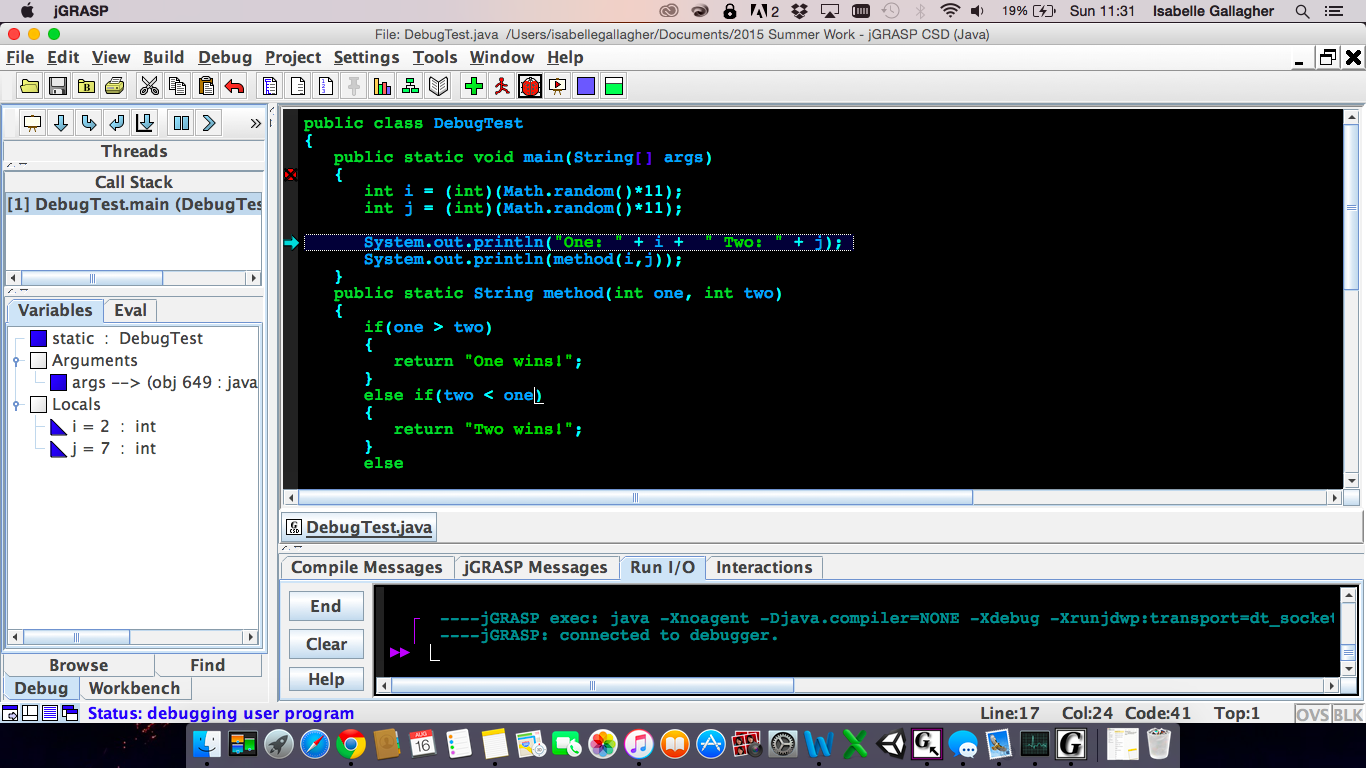


Now I can finally use the debugger. Before I start, I have to prepare myself. I know that it’s critical to our success to know exactly what I want my program to do before writing it. I need to know what every single line of code should be doing, so that way I can figure out which line is the culprit – the line that’s causing the problem.

Instead of clicking run at the top, I click the ladybug next to it to begin debugging:

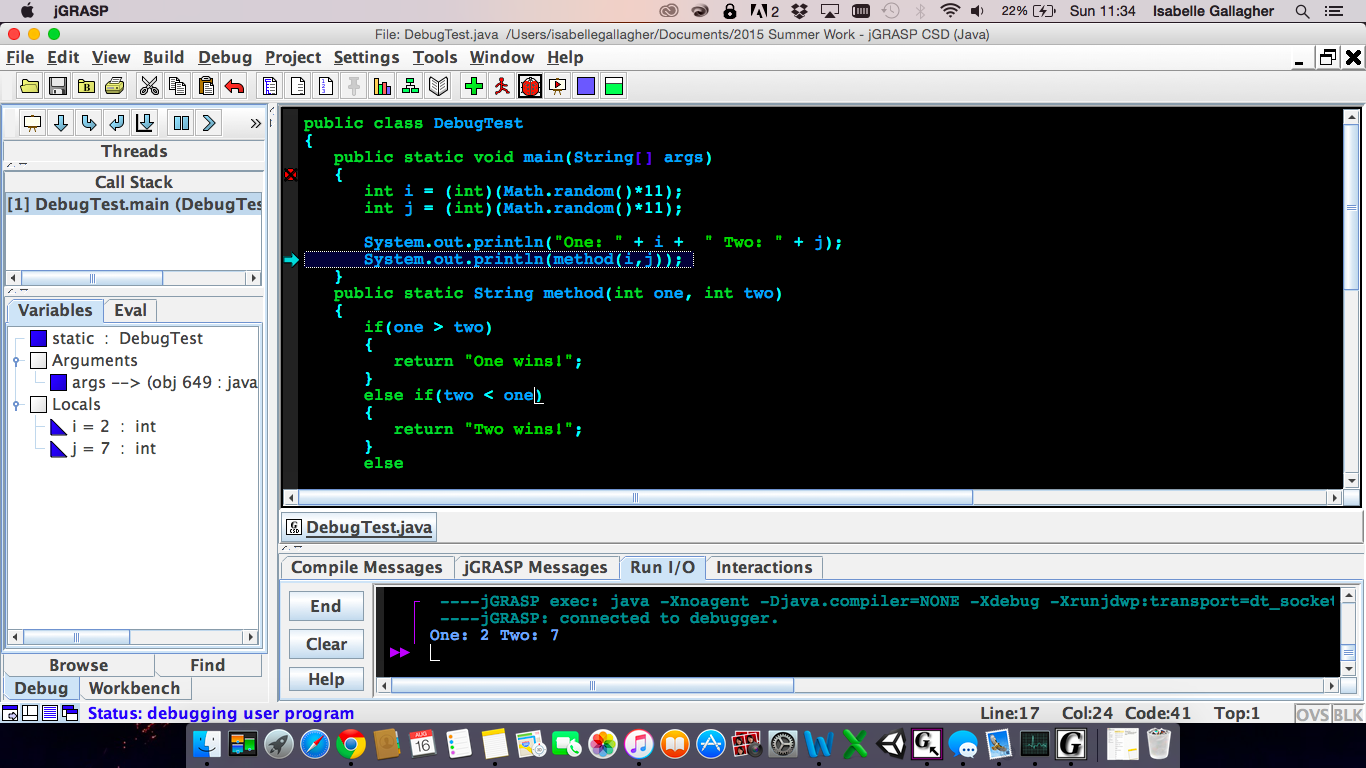


The panel on the side is where I will look as I go through the program to find the problem. Starting at the beginning, I know that the first two lines should create two new ints with values between zero and ten. Let’s make sure that’s what it does. The downward arrow is the step button. We use it to go through the program line by line. Pressing it twice will get me through the first two lines:



The sideways arrow shows where you are in the program. It moved to the next line, so we know the first two lines executed. We can see in the side panel that there are two new ints named i and j with values of 2 and 7, which are definitely within the range. The first two lines are correct.

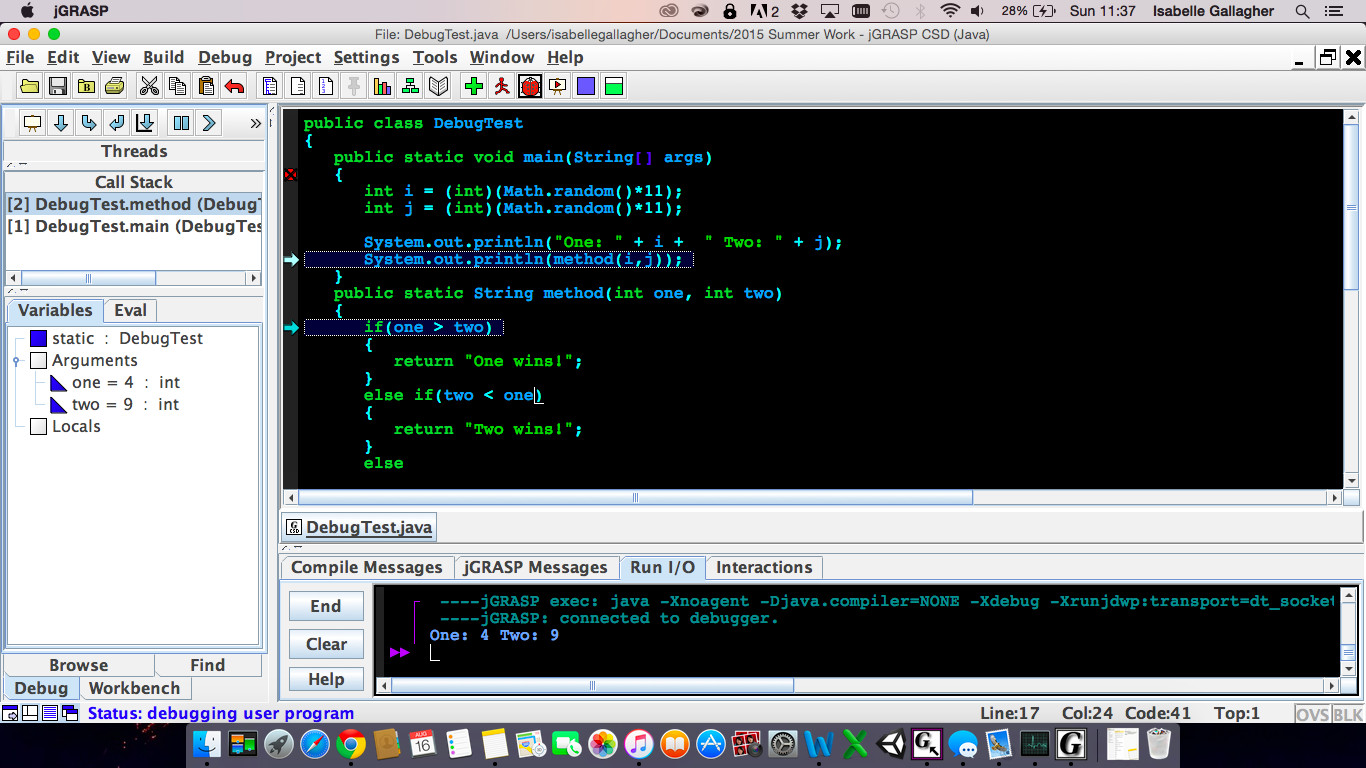
The next line should print out the values of One and Two. Let’s make sure it does that.



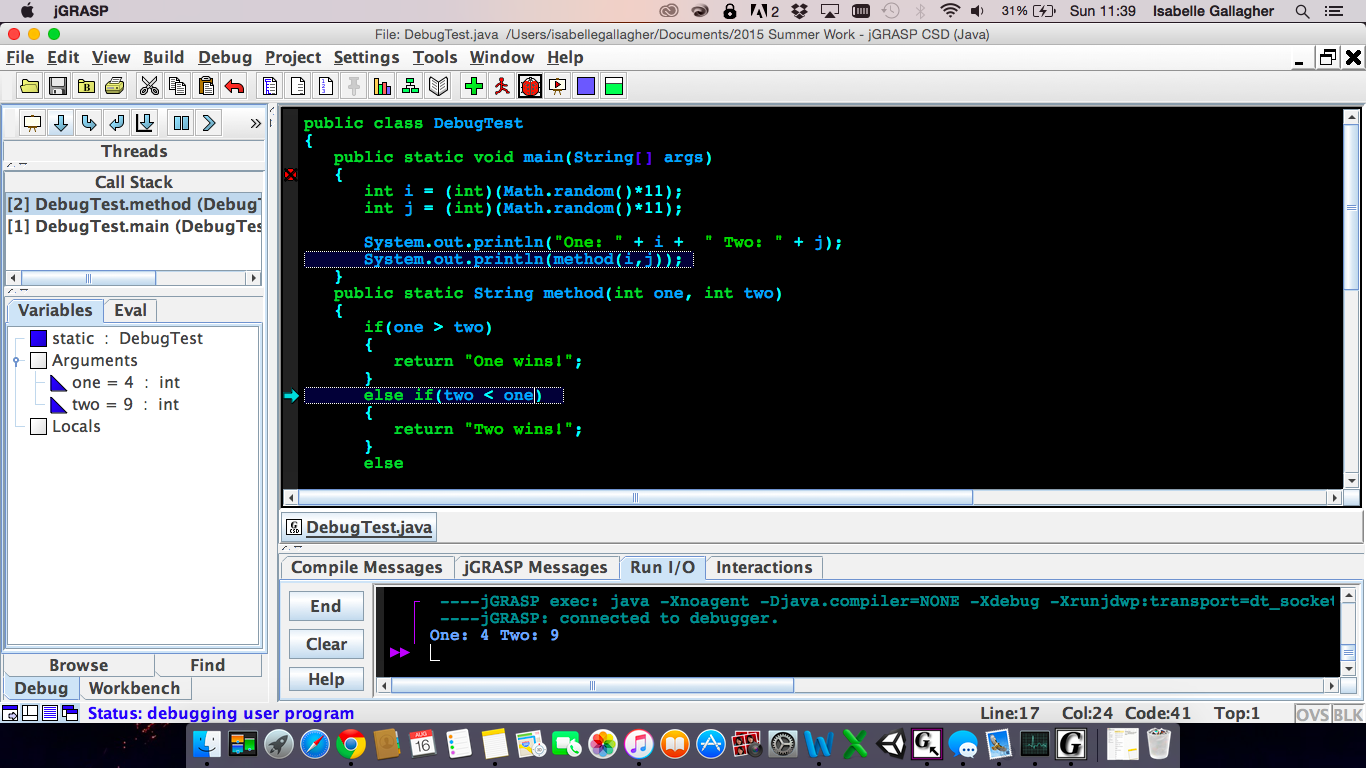
It definitely does. Let’s move on to the next line, which is the last line in the main method of the program. If we don’t find the error here, things might get complicated.

To spare you the pain and suffering, I’ll just tell you what happened and what went wrong here:

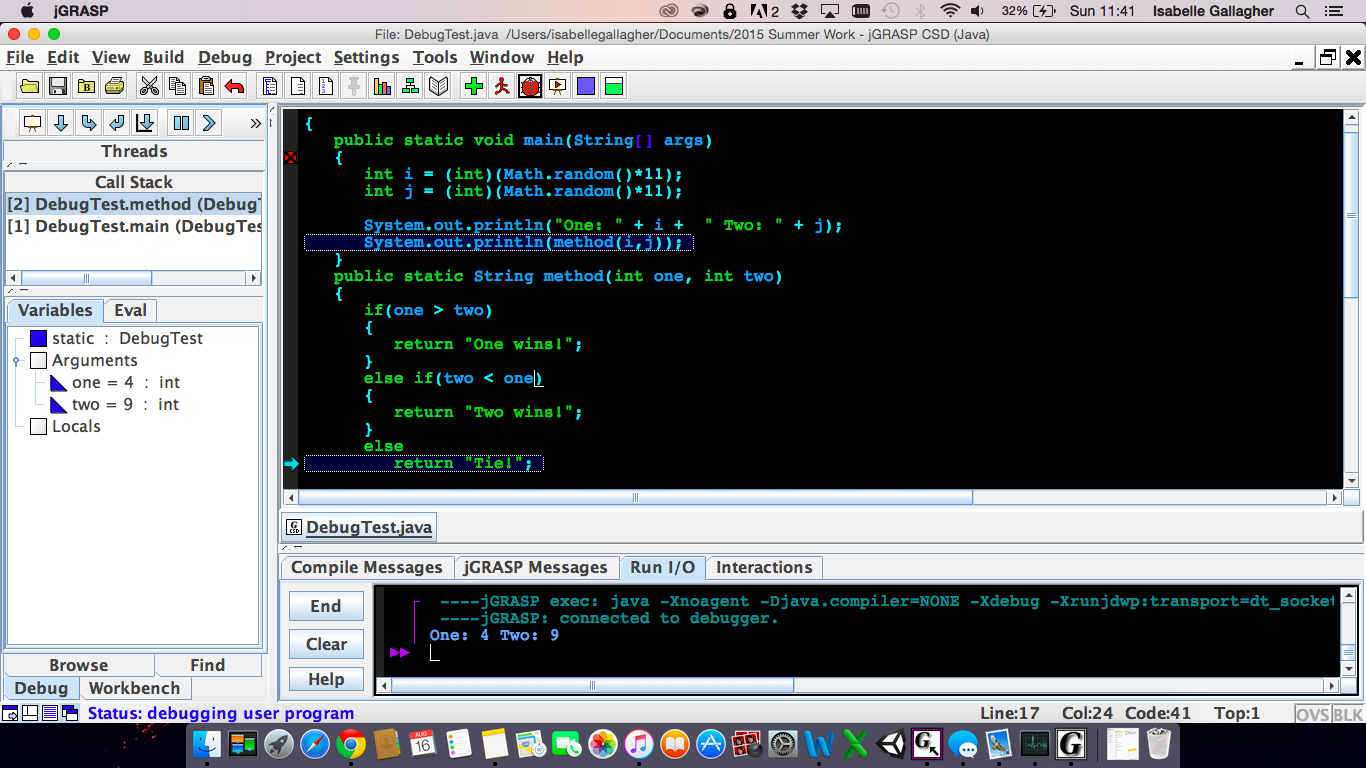
When I stepped through the last line, it printed out tie in the bottom. We know this wasn’t supposed to happen, but we are no closer to figuring out why. I now have a hunch that the problem is in method(). This time, instead of stepping through the last line, let’s step into it. This will get us to method(). The step into button is right next to the step button. If you click it once, you will get into method.



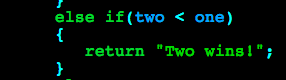
As you can see, the program is stopped at two points: the last line, and the first line of method(). Let’s continue to step through method. We know that one is not greater than two, so it should skip the first if-statement:



It does! The arrow has now moved to the else-if section. This next one is the part that the program shouldn’t skip over because two is supposed to win. Let’s see what happens.



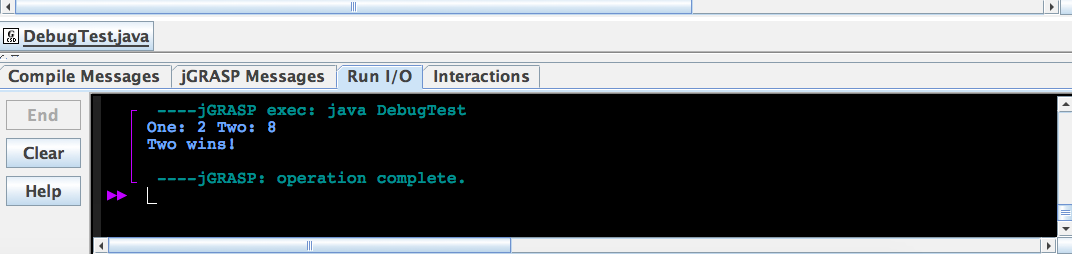
It skips right over that section and goes right to returning “Tie”. That’s a problem. We are very close now to figuring out the problem. Let’s look a little closer at the section it wasn’t supposed to skip.



Because it skipped this section, we know that the if-statement didn’t evaluate to true, even though it should have.

Aha! Can you see what the problem is?

I put a < sign when it should have been >. Often times, the most troublesome errors turn out to be something silly like this, but using the debugger helped me figure it out. Let’s fix the error and make sure the program runs successfully. Press the “End” button in the console to stop debugging. Get rid of the breakpoint by clicking on it again:



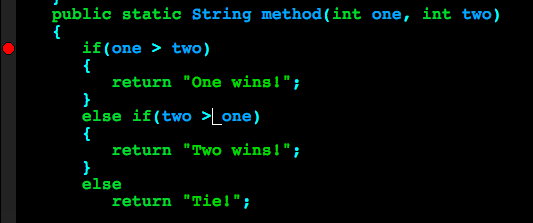
The program runs successfully! It says when two wins this time instead of saying tie.

I’ve successfully used the debugger to figure out where the problem was in my program. Even though this is a very simple program, the debugger gets increasingly useful the more complex your programs are.

See you next time!

P.S. If you were confused about using Step Into, continue reading below. There’s another way to do this.

Instead of clicking the step into button, you can end the debugging session. This time, get rid of the first breakpoint we put at the top of the program by clicking on it. Because we now know that the problem is method(), we can put a breakpoint at the beginning of method instead. Then we can run the debugger, and it will start debugging at method().



It will have the same effect as doing step into.