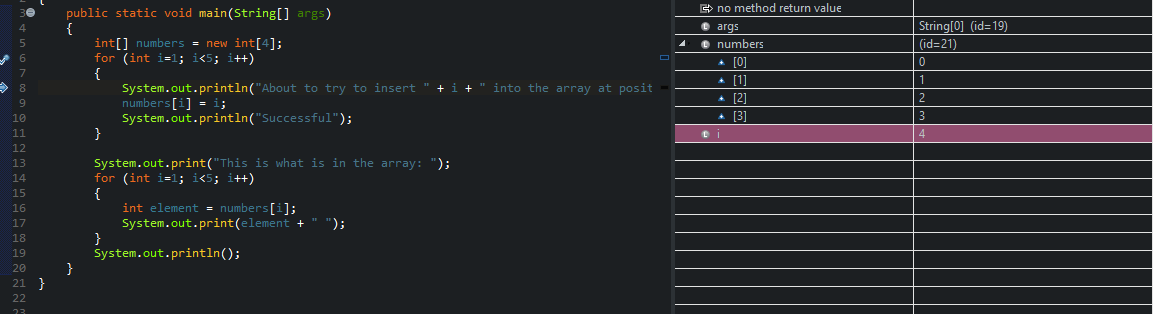
Stephen Hirschel

Lab 4 - Debugging

For DebuggingExercise.java, I set a break point at the beginning of the first for loop to help diagnose the problem. The problem became instantly clear, the array is only capable of holding 4 values, which are array positions 0-3, and the loop is set to start at 1, immediately ignoring the first position. The loop is also set to run until i < 5, which will cause it to go out of bounds. Between expanding the array and lowering the loop threshold, I opted to lower the loop to i < 4 to satisfy the requirements of the array. The second for loop also had this problem, so I modified it to match the first loop and the code runs without errors, and outputs everything it is supposed to.

A screenshot of a computer screen

Description automatically generated



A screenshot of a cell phone

Description automatically generated

For DebugHash.java, I first started by creating a hit count breakpoint on lines 60 and 62, inside the main method. This did not work, and the code just went on infinitely. Upon closer inspection of the code, I place a hit count breakpoint on line 33, inside the printArbitraryHashes method. I set the hit point to suspend at 49,791 “hits” and stepped into each line after. When I stepped into line 37, the hash string on deck was “2a84296c6a45c4734bbe39beebb670ea”.A screenshot of a cell phone

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

For FibDebug.java, Eclipse immediately notified me of an error on line 3, defining the main method. There was a close parenthesis missing after args, so I remedied that before debugging. The code worked fine after that, but upon research of what the 15th Fibonacci number actually is, I discovered the code is actually returning the 16th number in the sequence. I modified the while loop on line 12 to (n-1 > 1) and now the code returns 610, the 15th Fibonacci number.

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

For Marker.java, comments state that only one message should be output, yet at first run 5 messages are printed. Putting a breakpoint at the beginning of the printGrade method and stepping into each line reveals that the if statements aren’t written in a way that would stop this from happening, as the given mark 90 satisfies too many of them.

A screenshot of a cell phone

Description automatically generated

For AccountDebug.java, there’s an instant flag on line 6 that is verified when trying to run the code. I set a NullPointerException breakpoint at the beginning of main, and the debugger catches it on line 6, as it should. Looking at the Account.java, you see that the Account method should have a string argument, which means it should have its parameters defined as a string. After fixing this, the output worked, but the format left something to be desired. “$-100.0” isn’t the kind of output a user would expect from their bank, so I dug a little deeper which took the debugger deep into the FloatingDecimal.class as I stepped into it. Setting the default balance to 400, the math worked out fine in the output, as far as the algorithm goes, but it is stated that balance should never be negative on line 11 in Accounts. I created an if statement in AccountDebug.java that catches a negative balance and returns a string denying the withdrawal. I then added DecimalFormat to the debug code to format the output in a manner that is appropriate for banking. A screenshot of a computer screen

Description automatically generatedA screenshot of a computer screen

Description automatically generated

A screenshot of a cell phone

Description automatically generated

For the PersonDebug.java file, there are errors that pop up before running anything. Static is misspelled “Statuc” on the main method. Easy fix, but this highlights a new error on line 4. Person, while being the name of the class we want to call, is undefined, and you can see in Person.java, there are some issues as well. The Student constructor has no reference, which keeps throwing an error on the call in the debug on that line 4, so I changed the name from Student to Person. Now running reveals that the name and age variables are never called, because they are initialized in the Person constructor after they are already initialized as private class variables, so I removed the String and int declarations inside of the Person (formerly Student) method. Finally able to get an output, the console proudly declares that Fred Flintstone is Fred Flinstone, which while existentially true, is not the goal of the output. Another step into debug session reveals that the print statement on the debug file calls the getName() method twice. An easy fix to make the second method call getAge() and everything is working as intended. A screenshot of a computer screen

Description automatically generated Yabba Dabba Doo! A screenshot of a cell phone

Description automatically generated