DebuggingExercise.java

Specifically WHAT were the issues in each debugging situation?

- The program was attempting to insert a value into **int[] numbers** at an index that did not exist. The for loop that inserted values ran from index 0-5 where the highest index in **numbers** was 4.

- The program was attempting to access a value in **int[] numbers** at an index that did not exist. The for loop that read and printed values ran from index 0-5 where the highest index in **numbers** was 4.

Specifically HOW did you use the Eclipse Debugger to find the issues?

- I set a breakpoint on each of the for loop lines that iterated through **int[] numbers** with a hit-count at **4**. The error being thrown was an **IndexOutOfBounds** exception, so I knew it most likely had something to do with iteration. Close inspection of variables at this point confirmed my suspicions.

Specifically WHAT were the fixes that you applied to correct the actions?

- All that was required was changing the limit on the index iteration from **5** to **4** in the two for loops.

DebugHash.java

Specifically WHAT were the issues in each debugging situation?

- Without a print counter or something along those lines, it is impossible to tell how many hash codes have been generated — identifying # **49791** would be impossible.

Specifically HOW did you use the Eclipse Debugger to find the issues?

- I set a breakpoint on the **while(true)** line that controlled the infinite generation of hash codes and set a hit count at **49791** to identify the value at the 49791st iteration.

Specifically WHAT were the fixes that you applied to correct the actions?

- No actual *fixes* required in this one.

FibDebug.java

Specifically WHAT were the issues in each debugging situation?

- The **fib** method for generating the n-th term in the fibonacci sequence does not return the correct number.

Specifically HOW did you use the Eclipse Debugger to find the issues?

- I stepped through the program and noticed that the **while** loop responsible for iterating through the sequence was starting one term too soon. No hit counts or watchpoints were necessary for this one.

Specifically WHAT were the fixes that you applied to correct the actions?

- Changing the **fib** function to return **f0** if the specified n-term was **1** or to return **f1** if the specified n-term was **2** before looping through **while(n>2)** was enough to correct the error

Marker.java

Specifically WHAT were the issues in each debugging situation?

- The **printGrade** function seems to be printing the incorrect evaluation of a given numerical grade. More specifically, it is printing the correct evaluation, and then every possible subsequent evaluation.

Specifically HOW did you use the Eclipse Debugger to find the issues?

- I stepped through the **printGrade** function and took note of each boolean evaluation of the grade — it looks like *all* of the valid possibilities evaluate instead of just the highest.

Specifically WHAT were the fixes that you applied to correct the actions?

- Change the second through last **if** statements to **else if** statements, that way only one will evaluate as **true**.

Account.java & AccountDebug.java

Specifically WHAT were the issues in each debugging situation?

- A **NullPointerException** was being thrown when the Account **a** was deposited into.

Specifically HOW did you use the Eclipse Debugger to find the issues?

- I stepped through the main method of the driver and noticed that Account **a** was not actually an Account object — it was a **null** object.

Specifically WHAT were the fixes that you applied to correct the actions?

- The **Account** object, **a** was being instantiated as **null** instead of as an Account, so we just had to change that and add an account name to the constructor method.