DebuggingExercise.java:

The issue in this situation was that the first loop was parsing outside of the array's index. This was happening because "i" in the for loop was assigned the value 1 which was also being used as an index for the array. The issue comes from the fact that indices start at 0 so the program would get an error saying that the index was out of the array's range when "i" became 4.

After running the program once to see what kind of error I was getting, to further identify this issue, I toggled the break points around the first for loop I stepped into this loop to see how the array was being filled out. This was when I saw that the index 1 was being assigned the first value instead of the index 0.

In order to fix this issue, I subtracted 1 from i that was being used as the array's index in order to get the first value to be assigned to the index 0. Initially it was numbers[i] = i which was changed to numbers[i-1] = i to fix the "index out of range" error.

DebugHash.java:

2a84296c6a45c4734bbe39beebb670ea

Value was obtained by setting a hit count breakpoint set at 49791 on the getHashString method.

FibDebug.java:

The first issue that was apparent was that the parenthesis defining main was not closed. The next issue was that the program was not returning the right value in the fibonacci sequence when given a number. Specifically, inputting 15 into the method initially was outputting the wrong number. The numbers outputted should have been the output if the input had two added to it.

In order to identify this problem, I placed a breakpoint on the fib method and stepped into the while loop to see how the values outputted by the method were assessed. When doing so, it became apparent that the initial two values (f0, f1) were not taken into account as actual values in the fibonacci sequence when in reality, they should. For the error regarding the parenthesis, this error became apparent when trying to run the file.

The first fix was to close the parenthesis in the main method. The next fix was to take into account the first two values which were done by subtracting the value "n" by two in the while loop. Initially the while loop read: while(n>1){n--; ...}. My fix was to subtract two from "n" making the while loop read: while(n-2>1){n--; ...}.

Marker.java:

The issue in this situation was that the program's method would not terminate when a value was passed through a previous if statement. For instance, the initial value 90 when inside the method should have only returned "High Distinction" however, the bug here was that the value 90 was also going through the other statement returning all the other outputs.

To identify this issue, I set a breakpoint at the printGrade method and while stepping through, saw that the value was being passed not only through the appropriate one but through every single one.

There were a couple ways to resolve this issue, however, I just added an "else" in front of each of the other if statements following the initial one that returned "High Distinction." Previously, by having each if statement as an "if" it essentially made this entire block of if statements multiple individual statements whereas by defining them as else if statements, there is just one individual if statement block making it so that the value 90 would return the output that we wanted without printing the rest of the if statements out.

Account.java & AccountDebug.java:

When running AccountDebug.java through the debugger, a message saying main() was throwing comes up. In other words, an exception was getting thrown. The cause for this issue was that the methods being called under main were not static methods.

The issues were found by hovering over every instance where a method was called in main where issues were identified as well as ways to fix them. By clicking on the solutions that Eclipse suggests, it would go through the Account.java source file and list out how each method should be called.

In order to fix the issue, all methods that were called in main became static along with the instance variables. By doing this, the main method was no longer throwing an exception and was outputting the wanted output.

Person.java & PersonDebug.java:

When attempting to run the file, the main method is shown to be throwing. First issue that was apparent was that static in the main method was misspelled. Next issue was that the constructor Person(String, int) was unidentified. When looking at the Person.java source file, the constructor was defined incorrectly suggesting the reason why the constructor was unidentified.

Eclipse underlined the lines of code that were having issues. By hovering over these lines, the issue could be identified along with possible solutions. In this instance, it becomes clear that the issues were surrounding the fact that the constructor was not created correctly along with its contents.

To resolve these issues, first, the typo in main was fixed so that it was properly static. Next, the constructor in the Person.java was fixed: previously called Student() but in order to actually be a constructor, it must have the same name as the class it resides in. The remaining issues were due to values being defined but not actually being used. When stepping through with the debugger, it was suggesting that the values had to be in reference to the instance variables as they were not initially, where as they were being referenced in the other methods. To fix this, instead of defining these values as new variables, “this.” was put in front of the corresponding variables in order to establish the reference needed. After doing this, the program outputted the desired output.