**\*\*NOTE\*\***

In testing my program on the flip server, I encountered, and was unable to resolve an error *‘****Unsupported major.minor version 52.0****’* I was unable to resolve these issues between finding them, and submitting the assignment.

**DETAILS OF PROGRAM**

The program I have decided to use for this assignment is a Merge Sort that I had created for a class at LBCC. This program has two classes. sortMain(), which just acts as a driver for the other class. It also has sortBody(), which contains all the methods, and data interaction.

In order of usage, sortMain is exclusively used to make a call to the run method in sortBody, run then calls on runSort. runSort is the collector for running all the other methods in their specified sequence to sort the array using merge sort. These are all the methods that do not have more complicated logic than just pointing to other methods.

displayArray() is a method that when called will print out the current array, this is used for confirmation of position every step through the merge sort process, and is just an iterative for loop to move to the next position within the array, and print. displayTempArray() does this same method, but was for testing purposes while creating the project, it is not used and possibly should be removed

The method LowestLevel() gets to the bottom level of the merge sort, and compares values in position 0,2,4,6 with the value that immediately comes after, and sorts them where the smaller number will be on the left, and the larger will be on the right. It uses an if statement to compare the values and determine where the value should reside.

The MiddleLevel() method now has the array in two parts, the first half, and the second half. It uses a for method to go through the two separate halves, and a for method to decide what position a value should have within that segment of the array.

The HighLevel() method is a simple array sort to be used after all the other situations are exhausted, it will just compare and bubble values to their intended positions.

The merge() method is used to take the values in the tempArray, and merge them with the main MyArray to set MyArray to the new standard to be displayed to the user.

I pass an array as a parameter to satisfy that section of the homework’s requirements.

**BUGS IMPLEMENTED**

A total of three bugs were implemented.

One will ask you if you want to rerun the program, it checks the value of the variable, and if you enter no, it will re-prompt you if you want to rerun the program or close the program.

One will overwrite a value of the array with a temporarily used value, causing one value of the array to be overwritten, replacing one of the values initially implemented with a new number.

The third bug has an incorrect comparison when sorting the highest level of the array, making it sort incorrectly, and end with an array that has

**HOW DID EVOSUITE WORK**

Evosuite was the tool I selected for this project, as it has easily accessible plugins for bothe intellij and eclipse. It appears that Evosuite scans the code and methods for expected values and types that the program will be interacting with when it is running, when it runs its tests, it checks these expected values with the end results of the program, and confirms that they are true. Evosuite on their website claims that an important part of testing is code coverage, and while their tool did not often create an improvement in the number of bugs found by developers, it did find that code coverage performed by tests was increased by 300%.

Evosuite reaches these goals by implementing a number of things into their software. They generate JUnit 4 tests for classes, and focus the generated tests on minimizing test cases needed, as well as thoroughly testing branches, outputs, and mutations. In addition to all of these, the tests run in an artificial sandbox to prevent any potential damage the program is capable of causing. The scaffolding created when generating tests is also used to prevent creating flakey, and inefficient test cases, while also creating structure for further test cases to inherit.

**WHAT EVOSUITE DID WELL/POORLY**

Evosuite seems to have some difficulty anticipating what the intended actual output of the program should be, while it understands that for my specific case the output should be an array of int only containing int values is correct, however, it does not go to the level of ‘this value should be 2’. the defects I had created for this program went undetected through the automatic test generation.

**HOW TO IMPROVE**

Evosuite at some point could create some form of UI that allows the user to step-by-step define what values the variables equate to at any time, and then generate tests around that, analyzing certain methods could be much easier with that implemented, and there could always be a fallback if there's not necessarily a consistent value the method will have, evosuite can do it's current test generation.

Evosuite also is a little daunting to learn at first, as I had my bugs in my code, test cases ran, and it reported no failures. I assumed this meant that it did not detect the bugs, but it seems that it did find them, or at least understand that something was going on, but it did not report these as failures when running maven.

**HOW MANY TEST CASES WERE GENERATED**

In total, between my sortMain and sortBody classes it had generated 31 test cases, with many of them varying on what they enter into my y/n option. 28 of the total were dedicated to the class sortBody, which houses most of the components of the program. The other 3 are affiliated with sortMain, and appear to mostly just be running the program.

**DID EVOSUITE FIND BUGS**

The program was unable to find the specific bugs that I had implemented, due to the way it runs. I have included a file named Testlog.txt to display the run where it claims nothing bad was found within the 31 separate tests it had run.

**ABILITY OF TOOL TO FIND BUGS, HOW TO ADDRESS THE PROBLEM IF IT DID NOT FIND THE BUGS**

In my situation, as far as I understand, the tool was unable to find the bugs I had placed into the program, and I understand that due to limitations in knowledge of the code created by an individual using the program it is very difficult to come to a complete conclusion on what values the program should end with, and what the actual intent of the program is. These bugs that I did implement are easily caught by manual testing, as evosuite seems to scan for expected types, and general information, as opposed to the specific interactions within that data it may catch other more subtle errors in a much larger project easier.