Module 7 Testing Assignment

Explanation on Testing Design

For this assignment, I mirrored a similar strategy which was identified in one of the supporting learning modules. With IntelliJ, I defined a new directory called “tests” and associated it to the root folder of the project directory. I then established the project structure’s modules to know of the new testing specific directory. The creation of this directory and mapping allowed for a clean separation of application resources and test scenarios. This is key as you need to be careful not to accidentally develop a testing class as a run-time/dependent class of the application.

Once the directory and mappings were complete, I created the test class with three test scenarios, one for each method for the class. I also created a global private object which can be used for all the testing scenarios, which was preloaded with a word and a simulated number count.

For the getCount testing method, I had the method validate that for the testing object that it can successfully read the value stored in the count variable. This test was performed by the assertEquals method. If the values matched, then it would pass.

The getWord testing method is similar to the getCount, however for this one, I decided to leverage a Boolean test to make sure the values didn’t match the expected result. Normally, you would do this test to either make sure the statement was true or use the assertEquals method, but since there is only a few tests for this assignment, I decided to do some reverse logic.

The final method test, compareTo, compares the word count values between two objects. Since it needed a second object to compare it against, I declared a new object in this method. The challenge with this test scenario is the expected test value is a wildcard based on the object you use. Thus you can’t leverage this test with objects outside of this specific testing scenario.