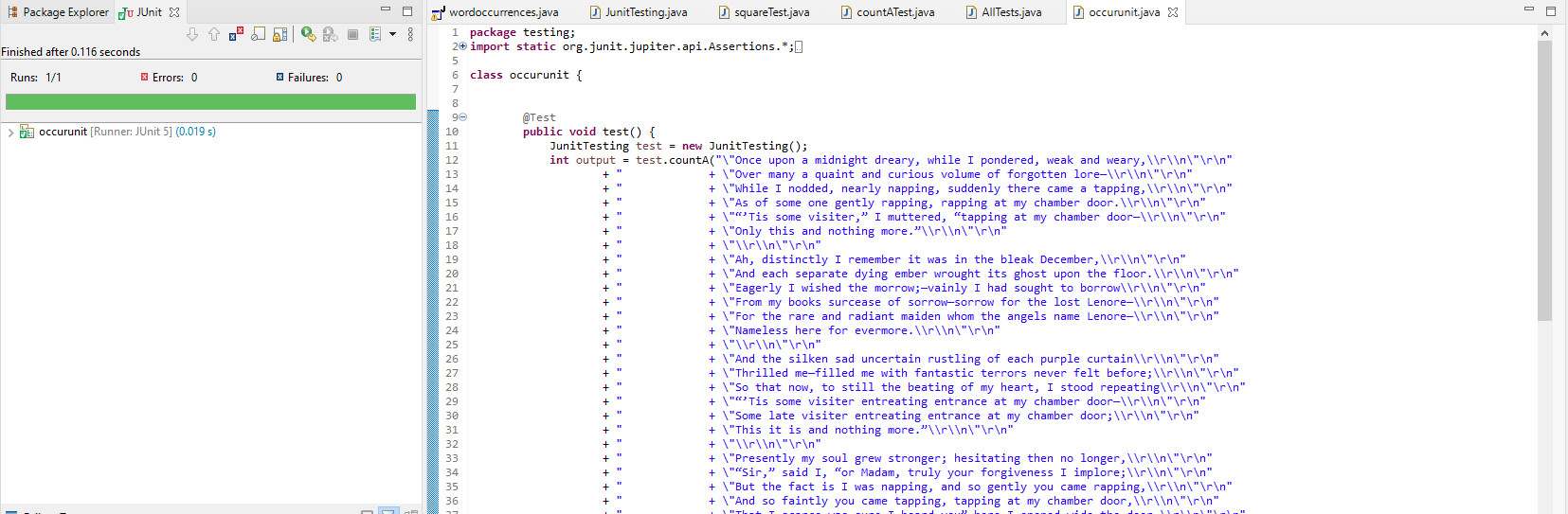
So the process of Unit testing is the observation of my test program under a controlled environment in which a JunitTest exerts stimulus to the application and if the program responds in the expected manner the unit passes, if not then the unit fails and can then be sourced to the individual test in the suite.

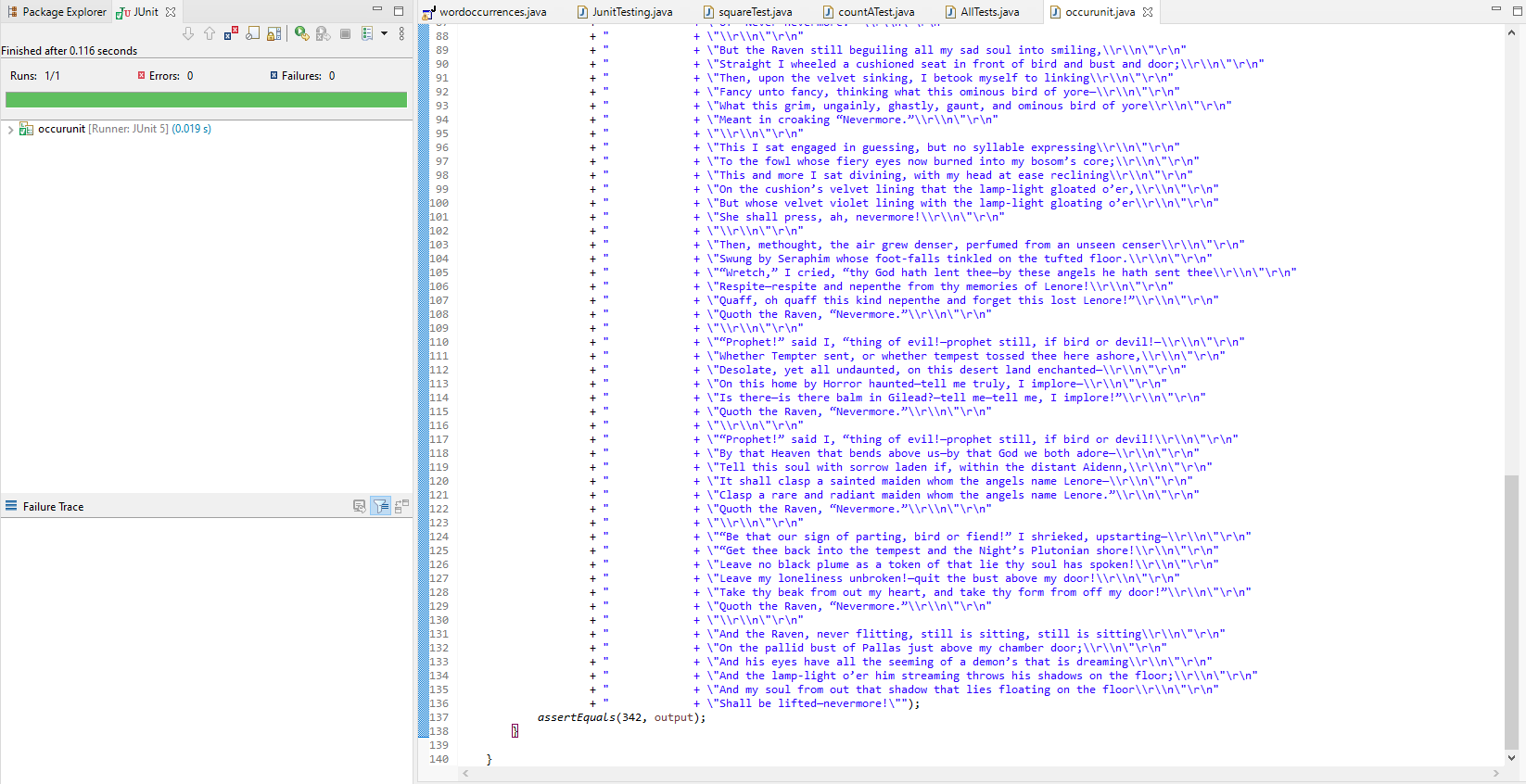
To start, I first followed the modules tutorial(<https://www.youtube.com/watch?v=I8XXfgF9GSc> ) of creating some base tests ( squareTest.java, countATest.java) and then placing them in a Testing Suite to observe several tests at once. This began with: modifying the original “word occurrences program” and adding a new Junit Testcase to the src folder. Once this is created, a basic setup is to place the Junit in a named package , then enter the Junit’s class name.

A default template of a Junit is then given and can now be modified for a specific test.

The test uses an expected conditional modifier that has been performed by an assertEquals() method that “fact checks” the portion of the program as either a pass or fail system. The special annotation often used to identify tests is simply @test which helps define those sections being tested. This annotation helps mark all test cases used in troubleshooting, which on small or large projects can include multiple test cases using multiple methods .

Then after that in order to tailor my tests to the specific “word occurrences” program i began to experiment with what specific elements of the program i would want to count or test the logic of, from my current understanding JUnit testing is mostly a pass or fail observation of classes in a program , This would have to include analyzing the text for specific words or number of words and report if the program counted the specific number correctly . I Then placed The text poem of “The Raven “ by Edgar Allen Poe in a similar Junit Test Case to (countATest.java) and found that their were 342 uses of the letter “A” , once this was changed in the expected outcome the program gave a pass rating and showed the application detected the correct elements of the poem.





While my Junit testing may not have all the features or capabilities that can be expressed by such a powerful tool i can speculate a wide variety of benefits of this process in coding:

* Can be used to “Failure trace” reasons why test cases may have failed or where the problems originated from.
* Junit testing has the benefit of timestamps to further assess when potential problems occurred and at what stages of development.
* You can save Junit results on the Failure trace tab by copying the failure list; this can be saved as a full history of tests and potential failures.