**Junit Testing**

In order to implement a Junit testing there are several things that I had to change one of them being adding a new method that would return the value of the HashMap that we populated with values in our word occurrence file. The method was called returningMap and it was a public HashMap that would throw an exception because of the try catch loop and error handling we are dealing with.

**A screenshot of a social media post

Description automatically generatedHashMap method returningMap()**

I ended up hardcoding the test URL input for the GUI just to avoid having to input the URL over and over again. This will be change to user input in the final version of the code. Only for testing purposes was this done. The reason why for implementing another method was due to the action method for performing the button action was a void method. Since this function is void and we cannot return anything the only way around this was making another method (returningMap) that will be called upon instead of having all the logic in the actionPerformed(actionEvent e) method like I had before in Module 6. The HashMap was declare at the top of our public class to utilize across the entirety of our code.

In URL\_test Junit test file I made a HashMap object called expected that would hold values to be test against our own HashMap that we will return using the returingMap method our word occurrence java file. I specifically made a simple index.html file with a h1 tag and one p tag with a sorted arrangement of characters that included special character and spaces to check if our word occurrence ability to only retract alphanumeric values. I made and ran this HTML file using Visual Studio Code since it has a option to run HTML file live using the live server addon. I inserted a total of 16 words into our test HashMap along with values assigned to each word.

**Junit Test**

A screenshot of a computer screen

Description automatically generated**A screenshot of a social media post

Description automatically generatedSimple Index HTML**

**Index.html file running live with visual studio code via the Internet Explorer browser**

A screenshot of a cell phone

Description automatically generated

I then made another HashMap called output that would be compared to the expected HashMap to see if the values are all the same. Using the assert Equals we pass both the output HashMap and expected HashMap. If both HashMap are the same, then assertion is true and test case passes. If there is another difference, then test case fails. I implemented both an example of the test case passing and not passing by changing one of the test cases values to intentionally fail.

**Testcase passed successfully**

A screenshot of a computer

Description automatically generated

**Test case failing (done intentionally by altering value in test case)**

**A screenshot of a social media post

Description automatically generated**

In the future I would like to add a functionality for more robust testing for both passing URL as parameters and string.