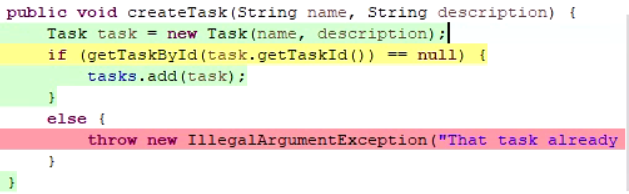
Over the past t

wo modules, I have created and tested two applications, each with several methods to test. To ensure that I was aligned with the requirements laid out by the rubric, I created a separate testing class for each of my Java classes. I also ensured that I had a unit test for each function or type of function. For example, if I needed a setter for a private field in my class, I ensured that all branches of that logic worked as intended.

For last week’s test cases, I didn’t run any coverage checks. I did this week however and was surprised at how many branches I had missed. This week, I was able to reach 92.2% coverage. This tells me that for each method, I was able to test almost all of the possible logic the code could have reached.

To ensure that my code was technically sound, I used those percentages. On first coverage run, I believe I was a little bit over 50% coverage. I was able to use the syntax highlighting in Eclipse to go back and make corrections in my code where needed, or write test cases that included logic that had been previously excluded. This was a cycle that I remained in until I was happy with my code. Below a snippet can be found where we can see the syntax highlighting showing covered, partially covered, or not covered at all.



I found that I was struggling a little bit these modules with efficiency vs readability when writing my test cases. I relied heavily on the “assertAll” method that is built into JUnit, which helped me with both cases. Below is an example of such a case where I wanted to test that the object was successfully added by checking the name and description of an obtained element from my data structure.

