Obada Hamdan

Professor Jeff

CSE 4321 Software Maintenance and Testing

6 December 2022

Summary and Discussion

In this project, we were given a program called Printtokens.java which had a few bugs in it to be found. The whole idea if this code was to accept different symbol inputs to be provided by the user, by either providing the symbol from console or from a file. The program would the classify each of these tokens by its type which could be a special symbol, character and string constant, numbers, keywords, special characters and comments. Basically, it is determined from the beginning which route the program will take whether there was a valid file passed or not.

During this project, I would have doubts that were always clarified from the teacher assistant, which I really appreciated. However, being familiar with Java really gave me a boost, especially in the beginning when I was trying to understand the code. For this project, I used IntelliJ IDE for its ease of use, and ability to implement Junit easily. I set up Maven for my project and included all the required implementations to be able to use Junit. The Junit portion was the part I learned the most from.

I started this project with the goal of being aware of the bugs in the program, and not letting that affect my ability to comprehend the code. When creating the CFG's, I made sure to

include that the basic blocks were correctly identified. I also made sure to be aware that there were a few statements in the code that included two separate blocks on the same line. I did my CFG's by hand using good notes on iPad. It was a very stressful process, especially when I was using the graph as a reference to make my test paths. At that point if I found I made a big error that affects my test paths, when creating all my CFG's, I made sure to use the false branch to take the desired route, and true branch to take the undesired route. In this program, I've found 17 methods that needed unit tests done to it. I made sure to achieve edge coverage on each of my individual methods, where according to my reports, shows that respectively. Each method had different input that would execute the given path, which helped me a lot when generating my test cases to input into Junit. I have labeled my Test Paths as TestPaths in PDF format. I made sure to include each test case above each of the individual test method, to guide me through implementing it. When forming my tests, I made sure also to use the best Value Source annotation for that given method. I did not implement an end-to-end test for this project, due to time constraint.

For my reports, I have included them in a separate folder with all the reports exported. This part I was checking the coverage really gave me the chance to find more errors in the code, that were fixed and mentioned in the Faults and Corrections document. For this submission, I will make sure to include all required files. In total, I had 73 test methods for the 16 that were tested except main.