Ninefeldt, Chloe J

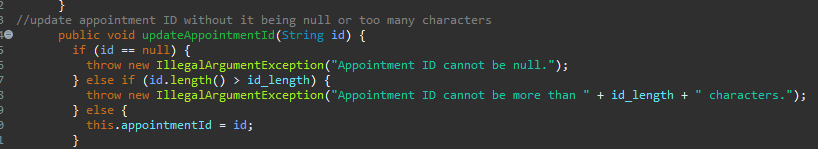
08/15/2021

CS 320 Software Test Automation & QA

Project Two

Grand Strand systems

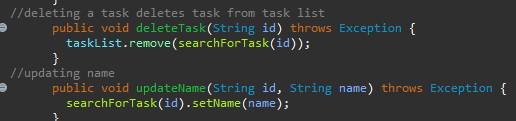
My testing approach aligned with the software requirements as I tested the methods that were required in the main code. For example, in the appointment service, one of the requirements was that the length of the appointment ID could not exceed 10 characters or be null. With this requirement, it was written in the base code that it would throw an illegal argument if the ID entered was longer than 10 characters or was null. I then proceeded to test this in the appointment test and made sure that the code written in the base code did as it was intended. After a couple failures, it went through finally and I was able to be confident that it worked. Please see image below, this is the code I wrote for the base to follow the software requirements for this particular assignment. It was similar throughout as most things needed to be an x number of characters.



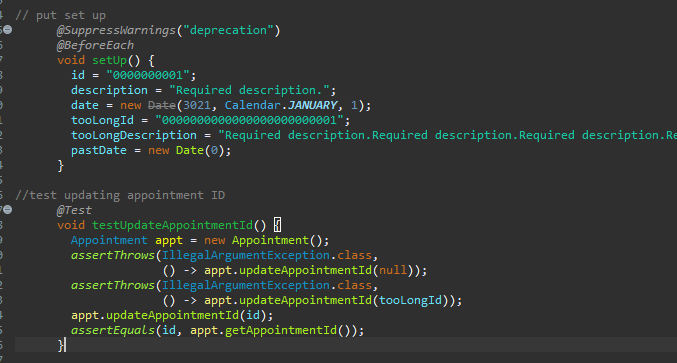
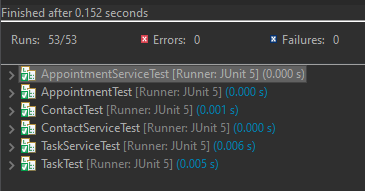
In the beginning, I didn’t know how to write proper code and test cases in general. My first assignment had a ~60% coverage base which I was happy with then. Coming to realize, that’s not that great. Towards then end, I ended up having a 90.7% coverage which is great compared to what it was. I knew my testing was effective based off that as this showed my code was used and used properly. I checked the coverage more and more as this project went on as it gave me insight if I was writing code correctly. This was helpful and I was more confident in my work towards the end with the higher coverage percentage.



Making sure my code was technically sound was making sure I annotated my code correctly and properly named my classes and methods. As with the image below, when writing a test case to update a task, I named the test based on what element the task was being updated to keep everything easy to read and follow if anyone were to read my code. This test was to see if the task was updated properly. When testing a to delete a task, I named the test ‘deleteTask’ and attempted to delete a task and made sure it was deleted properly. Using @Test to declare the method was used in most of the testing and I made sure to keep it organized.



To ensure my code was efficient, I had test cases that made sure that certain strings were not null or didn’t exceed x amount of characters. Running the Junit testing, they passed and did not kick back any errors/failures as seen in the image below. I followed the rubric requirements and tested whether or not certain tasks could be added, deleted, or updated and whether contacts could be added, deleted, or updated. I ensured to the best of my capabilities that my code would meet the requirements of the customer. Same with testing appointments and whether adding an appointment was done correctly. See examples of some code below. I tested if things were entered incorrectly what would happen and what would happen if the inputs were correct just to make sure the program functioned as it was supposed to.



I believe that I used decision table testing for all three milestones along with an approach of writing the test cases before writing the other parts of the code. For each milestone, I started off writing the test cases beforehand to make sure that all requirements were met. An example of this is writing a test case that makes sure the input is no longer than x number of characters and is also not null. I also wrote a test case that tested a correct input so that I knew it was done properly. I made test cases that had different choices of inputs and made sure they had to proper output in order for the requirements to be met.

I did not use state transition testing since when inputs were entered, there wasn’t an option for changes of state. It was either under a certain number of characters, not null, or not happening. There wasn’t an option where if you enter four characters it did a different thing. The milestones were straightforward and didn’t have anything that would change the state of the system.

Decision table testing is when there are lists of all the inputs that could be entered and what happens when those inputs are put it. You can build it as a table and display what happens and then transport that information into code, or test cases. This can be used when there are certain rules that have to be followed for the system to function properly. However, you don’t have to enter in all combinations and possibilities, but only follow inputs that go with the rules that need to be followed. For instance, with the milestones, the ID could only be 10 characters and not null. So, I therefore wrote cases that tested the action of having an input of more than 10 characters or the ID to be null. This in conclusion came back with an error message as the rule is that it cannot be more than 10 characters or null.

State transition testing is where an input can be entered and depending on what that input is can change the state of what happens after. Sometimes it might not even get an output and it just changes the systems internal state. This can be useful when a business/project has inputs that could generate either a whole different output or just change how the system operates.

I employed caution within my code by changing my mindset as if this was an actual job and a company could lose a lot of money and time if I didn’t get this correct. With this mindset, I took more time analyzing my code, making sure test cases tested the correct information, that my code was done properly, and the system worked as it was supposed to. At the end of an assignment, especially the appointment service test, I appreciated everything I put into my code as I threw everything I knew into it. I did research on what methods would be best, how to test properly, and how to organize my code to make sure it was easy to read for myself and others. With this mindset, I think it helped tremendously as I was more careful than ever before to make sure my code worked. In the appointment service test, I struggled the most as I could not get a couple of test cases to work properly. For example, testing to get the appointment ID wasn’t working at all based on the Junit coverage percentage, and it took time to figure out what I did wrong. At the end when I finally found the solution, I realized how easy it was to submit code that almost didn’t work and how important it is to test my own work.

I was absolutely bias when testing my own code. This could happen anywhere whether someone wants to admit it or not. Having someone else test your own code will help catch more errors that are otherwise not seen. Everyone is proud of the work that they do, including myself. There were times where I was confident my code was correct and would work as I intended it to. Then when it came to writing the test cases and running a Junit test and it came back with errors, it is shocking. I’m sure this happens quite a bit as an actual software developer/tester. After the first milestone with so many errors coming back, I started looking at my code from different perspectives. I would write my code first, then leave it alone for a day or two if not more and then relook at my code with fresh eyes. I caught a ton of errors before running the Junit tests which helped me learn more about myself and how I write code. Then after correcting my errors, running the Junit tests I would get back only an error or two rather than multiple.

Cutting corners gets you nowhere in the technology field. You can try all you want but majority of the time it backfires. Writing code and finding ways to make it faster actually just leads to more chaos at the end because you’re more than likely going to have to rewrite your code or fix it. There were times myself in previous classes where I tried to cut corners and that was a lesson I will never forget. I have had to rewrite code a dozen times because I was too impatient or too proud of my work and I will take that lesson with me into my career. Now that I also know how to test my code, I can take that with me too in order to make sure my code actually works as intended. If I would have known how to write test cases before hand with other classes or projects, it would have saved me a couple of grades. In the field though, this could save a lot of money. Doing a project for a client, you want to make sure you deliver something that works and functions, and now I know how to ensure that it does or doesn’t. Whether this is the job I end up with or not, I want to be sure I did my job efficiently and correctly.