Project 2

This project overall was an interesting one. It was curious to see how the testing works with the code and the feedback it gives. It also all is determined by how it is written and what they focus is on. Making sure the requirements are met is the biggest key. Testing comes in different forms.

For this project, I decided to use a simpler form. I developed my testing to test the basic requirements and functionality of the classes. For both the contact, task, and appointment classes, they had specific fields that had constraints on them. I put those fields in their respective classes. To take care of the constraints on each field, I used the constructor to check and determine if the inputs were valid before creating the task or contact. If a field has an invalid input, an exception is thrown to be picked up later and the code fails. If everything is valid, it proceeds to create the task or contact for any future use. To test these items, I created a test code that created new objects. I tested whether or not an object could be created, and then proceed to check the length of each item to be sure that it was less than the requirement and was not null. These tests ensure that the requirements for the task and contact classes are met.

As far as the service classes, they had their own requirements to complete and test for. In the same way I developed the requirements for the task, contact and appointment classes, I developed the service classes for each object. I knew there needed to be a list of the objects, so I stored them in an array. From there, I created the add, delete and edit/update functions. The functions are self-explanatory of what they do with the exception of the delete and edit, that searches by the IDs of the objects to make adjustments. I believe these things lead to the overall quality of my code is solid. I know that my code covers every requirement because of the previous stated process that took into account the requirements given.

I am confident this is true and that my code is technically sound because I was struggling to get my test to all pass. To accomplish this, I went to tutoring to get another set of eyes on it. In doing this, they pointed out that I put my if statement to add an object inside the for loop checking to see if that object already exists. Moving this outside the for loop allowed for all my tests to pass. This lets me know that my code is technically sound. On top of making sure my code was technically sound to make it efficient, I made it simple. This not only to make it efficient but to make it easy to read. Instead of adding a bunch of extra wording and assignments, I used this.variable to make assignments in the constructor. This streamlines the code and allows for ease of reading. I also created get functions for the items that needed to be retrieved and set functions for the items that were able to be edited. In doing so, I made it easier to access the items in the objects and therefore making the code more efficient.

For each milestone I performed the same test. These tests worked as intended and I believe were the most fit for the assignment given. The test I performed was a functional or requirement test. This tests whether or not the code performs as intended. In turn, I tested to see if I met the requirements given to me. This type of testing is beneficial because you know for sure whether you achieved what was asked of you. There is no gray area, you either passed or did not. Another name for the type of test I created is decision testing. For this, I created a test that would execute a set of instructions and return true or false. Depending on what was returned, determines whether the test passes. I structed my code to produce a true or false depending on the action performed in the function. My test takes that Boolean and compares it to the expected value. If those values match, then the test passes. This makes testing requirements easy because it tests the functionality of the coded requirements.

I did not test reliability, usability, or performance. In this scope, these things are not necessary to know. In a bigger project or one that requires a good and fast performance, these things would be necessary. These things would be something I possibly test for when I put the project together rather than the individual milestones. Since the milestones are smaller and only need to make sure they function properly, I focused on the functional testing.

Even though I was focused on the functionality, when I put all of the classes together, I checked the coverage of my testing. This is to confirm that I am testing my code thoroughly and that it all passes. This is important not only to have proof that you have solid code, but it shows you are performing good practices to have all or most of your code covered by your tests.

As mentioned before, the mindset I adopted was purely functional. I knew if I completed the requirements and made the code function properly, then the test would pass. Therefore, it tells me that I achieved my goal. Testing can be complex, but because I tried to keep it simple and focus on functionality the test was straightforward. To limit bias, I crafted the tests based off the requirements given. This allowed me to put my own ideas for testing aside and simplify the process. Discipline is one of the most important characteristics for a developer. This is because creating solid code is the foundation of what we do. Cutting corners just leads to bad code or leaves spots open for hackers. I strive to create solid code; I check and double check my work for any mistakes. I also create thorough test to check the integrity of my code. With all of these things considered, this project went pretty smoothly and I feel I created solid code that will hold up.