Project 2

For each of the three features (contact, task, and appointment) I have used the used the same method, mindset, and frame in which I was testing. I approached each feature by making a new Java class. After this, I got started coding by first making notes to myself, in the text, of the requirements. Once I had the requirements notated out, I then concentrated on setting the stage for the strings (*public, private*) that were being used, pertaining to my requirements. After all the Strings were set, then I proceeded to put my conditions in place (*if then statements*). When this portion of code was completed, I then proceeded to set my “this” statements and directly after that, my return statements. My approach was aligned with the software requirements in great detail; I implemented all of the software requirements directly from the directions into the code so that the codewill align perfectly with the requirements. For example, in Module 5, the directions stated: *The appointment object shall have a required unique appointment ID string that cannot be longer than 10 characters. The appointment ID shall not be null and shall not be updatable.* The code that aligned along with this set of directions were:

public String ID;

…

public Appointment(String ID, String Data, String Description) {

if(ID == null || ID.length()>10) {

throw new IllegalArgumentException("Invalid ID");

}

You can tell that the JUnit test is working by the test displaying no errors and a green bar in the left corner of the screen. I ensured that the code was technically sound by aligning the code requirements with the JUnit test. (Please refer to previous lines of code from my test). I ensured that the code was efficient by watching the test time within the JUnit test results.

The software testing techniques that I have employed in this project are the JUnit tests and the hints that eclipse would give; I used the hints to correct the issues that I had within my code. I did not use vectors to test my code. Vectors in general help make it possible for there to be multiple threads. The JUnit tests, in general, gives you the ability to test the code line by line to see if each condition works.

While working on this project, I worked with a tester mindset. I tried to prevent a bug before it had the chance to fully appear. I used caution during ever step of the coding process by trying to prevent a bug before it appeared. It was important to appreciate the complexity of the code because you were able to see how the JUnit section directly intertwind with the regular code classes. For example:

public Contact(String firstName,String lastName, String ID, String phone, String address) {

if(firstName == null || firstName.length()>10) {

throw new IllegalArgumentException("Invalid first name");

&

@Test

void testContact() {

Contact contact = new Contact("Bill", "Bob", "1234", "6784047785", "102 May Lane, Car Ga 30579");

assertTrue(contact.getfirstName().equals("Bill"));

I was not bias in my code because I chose to have an open mind while I code to be successful within the code. Not for me, but I can see how being bias would be a concern if a developer, who was biased, were to test their own code. A bias developer would not be able to fully see their own problem in the code 100% because it would be hard for them to where they went wrong within the code, for example.

It is important to be disciplined as a software engineer and not to cut corners. Depending on what you are developing, as a developer, if you cut corners with a code or testing a code, it could potentially cost someone their life. For example, if you are testing a code for the new android phone on the market and you receive an error message for the code, and then ignore it, that could potentially make the phone not fully functional. For example, this could make the phone injure someone if the phone reaches a certain temperature, because initially you as the tester ignored the error messages.

I plan to avoid technical debts a practitioner in the field by making sur that I do not ignore the error messages, making sure my code is neat, and concise. I will make sure my coding is neat by making sure the code is as legible as possible; This in return will reduce the number of bugs and errors that may appear in the code.

Works Cited:

* Project 1
* Milestone 3
* Milestone 4
* Milestone 5