Larry Arndt

CS-320

June 14, 2021

Summary and Reflections Report

With the contact class my testing approach was aligned to the software requirements. I tested each of the attributes that the class assigns, and then went on to test all the exceptions the class can throw. I did this for each of the required cases that can trigger such an exception such as a field being too long or being null. The only thing that was not tested in the contact class were the setter functions which get tested during the contact service tests. The same is true for the task class all the requirements are tested with the exception of the setters which are tested in the task service test. This is also true for the appointment class but it is tested in its entirety because it has no setter functions.

In the services the functions are tested and the JUnit tests are aligned to test the functionality of the different class methods. In each service there is an and add and remove command that is tested that is common to all of the services. The remove method uses a search function to generate the vector location of the item to be removed from the ID given. The search function is also used in the modify functions specific to the contact and task services. The JUnit tests were effective in testing the classes. Based on the coverage data I gathered when I comment out the setters which are not part of the class for testing purposes, I get one hundred percent coverage for the contact and task classes. If I do not comment out the setters it is about sixty percent. All the other coverage percentages are over ninety seven percent. So I feel that the JUnit test were effective based on that coverage percentage.

I ensured my code was technically sound by following good programming practices. For example:

static void AddTask(String TaskId, String tName, String tDescription) //add a task to vector

{for( Task eachTask: taskVector) {

if (eachTask.getId().equalsIgnoreCase(TaskId)){ // catches duplicate id

throw new IllegalArgumentException("Invalid id");

}}

Task task = new Task(TaskId,tName, tDescription);

taskVector.add(task); }

This function adds a task to the task vector. It also checks to see if the incoming ID has already been used before, in which case it will throw an exception. The comments tell other programmers what is going on, and the descriptive variable names guide an understanding to what the code is doing as well.

The code seems pretty efficient, it does what it is intended to do. There is not a lot of repeated code that is doing nothing. For example the Modify contact is a pretty robust function:

static void ModifyContact (String ContactId, String fName, String lName, String pNum, String Address)

{int temp2 = -1;

Contact tempContact;

temp2 = SearchForContact(ContactId);

tempContact = contactVector.get(temp2);

tempContact.setFirstName(fName);

tempContact.setLastName(lName);

tempContact.setPhoneNum(pNum);

tempContact.setAddress(Address);

contactVector.set(temp2, tempContact);

}

It calls another function that returns the index of the location of the contact that is being searched for, gets the contact at that location and puts it in a variable temporarily for modification and then sets it back in the location when done. It is pretty efficient with the way the vector works.

In this project we did JUnit testing that tested the outcome of each decision branch in the program. It tested everywhere where code could be ran so there was no code that was not tested for complete coverage. We did not do integration testing to know if this code will work with the rest of a system. Also some white-box techniques may have not been done. We did not draw up any flow graphs, but we did do coverage testing, so I do believe that it covers the same thing. In a bigger project you would need to work the system pieces together and make sub systems functional together most likely and there would be testing that would go on to ensure the program works as intended.

It was important to employ caution in this project because I had to ensure that I got all the correct tests done to ensure the correct coverage. That meant that I had to choose to test the right aspects of the class I was testing. For the Services this meant testing each function of the service and for that I had to actual create another function that returned an object to the tester. In the base classes the testing had to test for all the exception handling and I may have thrown the exceptions twice each time and over tested now that I think about it but maybe it does not hurt to be over cautious.

I really tried to be objective as possible in testing my code. I am sure I may have some bias in testing my own code. I do know that in the appointment class where I was to test for a null condition on the date function it would not work. It is because it is parsing the date and when a null goes into the parser it throws a different exception. So, I modified the test to catch the other type of exception, so it passes the test now. It seemed the way to go. I do not know what else you would do. It could be a bias though.

I feel that it is important to be disciplined when writing software as a professional. I did copy my own code here and modify it a bit, but I think that is different than cutting corners. What I did was just being efficient. Copying code for reuse when you know the code already works as intended is not being undisciplined. It is important to be careful when doing so to make sure to update all the variable names and such correctly. I think I had one test name that I left miss named that I caught later. I plan to avoid technical debt as a practitioner in the field by striving to be the type of coder and tester that puts a good amount of time and effort into my work. Coding in a manner of good practice and procedure, as I have been taught and not being afraid to seek help in online references, forums and other resources that are available for aid. Like for this project I had to look up many of the commands I used with the vector. I found some in some old code I had wrote in a past class and some online. With that research I was able to make it work.