Summary and Reflection

For this specific project the testing approach had to ensure that all the requirements of the software were met. For example, many of the individual variables have limiting factors in place such as length that needed to be taken into account. The way I addressed these limitations was to implement a throw exception if any of the values were not what was expected.

throws InvalidInputException{

if(isValidInput(contactID)){

this.contactID=contactID;

}else{

throw new InvalidInputException(contactID);

Meaning if the value is a unique contactID there will be no issues however if it is not then the exception would be thrown.

if(address!=null&&address.length()<=30){

this.address=address;

}else{

throw new InvalidInputException(address);

}

Then the test code was developed for the contact class using various names and data points to ensure that the contact class runs with no issues and throws the exceptions when needed. Each individual variable had to be set with individual parameters to ensure the success of the application. The unit test that I developed for both classes provided excepted inputs and outputs to ensure the application encountered no issues when running.

Contact contact1 =new Contact("9898","ABC","ZZZ1","1234567891","Hyd");

When developing the code, I ensured it was in the most efficient way possible. I did this by declaring all variables and defining their types specifically to eliminate any confusion. One thing I could have done to ensure even more efficiency is incorporate break statements to evaluate various when clauses.

By successfully running the unit tests on both classes it shows that the code is working as intended at the individual level. Unit tests are the way to ensure the developed code is technically sound. Assuming of course that the test class is testing all aspects of a given application.

There are many different testing techniques that developers use to ensure the code is successful. These techniques can be broken down into two different categories being functional and non-functional. Throughout the milestones of this course, I have used many of these techniques to ensure my code is functioning properly. First and foremost, unit testing, this was done by implementing a test class to ensure the code accept inputs and runs according to expectations. If the code does not, then an exception would be thrown indicating there is a problem of some sort. This was then followed by an integration test which ensures that all the code components work fluid together. Such as a person setting entering information and setting an appointment. Another technique that would become very important as the number of users increases would be performance testing. Implementing code to test load capacity, spikes in usage as well as an endurance test to ensure the program can run under various conditions. Another testing technique that was not used as it does not really apply to these milestones would be security testing. If the program were to contain personal or private data from its users there would need to be a way to test to make sure that the data cannot be breached. Usability and compatibility testing would also play a major role in the development of this application to ensure that the end user experience is an enjoyable as possible and there are no inadequacies in the UI.

Overall testing is performed at all stages of development for Mutiple reasons. The true end goal of running these tests is to establish that the program or application can run and perform under real world situations and circumstances. These tests are used to catch and isolate failures hopefully before they can make their way to and influence the end user in a negative fashion.

As a software tester it is important to adopt a solution orientated mindset. Being able to not only isolate a problem but have the capability to discover why the problem is occurring. When developing any code, issues will occur but it’s the approach of problem solving that changes for each individual. Having code and objects that interact with one another at multiple levels can be fairly complex. Editing code when working on an object can create ripples effecting all other objects/classes in the system. Therefore, extreme caution must be used when altering any existing code. As unforeseen consequences can develop elsewhere in the code.

When reviewing your own developed code, it's difficult to escape your own, created mental biases and roadblocks. It’s crucial to approach review with an open mindset, by doing so the goal is to create code that can be understood and easily developed by others if necessary. This becomes of utmost importance in a team environment or project. Other individuals in the team should be able to critique developed code and provide constructive feedback to ensure that all coders/testers are on equal ground.

For projects of any size, it is important to thoroughly write and test all code being developed. To state the obvious this is the only true way to ensure success of a project. If corners are cut or shortcut are taken in any stage of development, the consequences could be anywhere from minimal, to catastrophic. For example, the Heathrow terminal 5 opening in London was an utter disaster, code wise which led to millions of dollars being lost in the process. Simply because tests were run and cut short due to the extended construction time cutting down the total numbers of tests that could have been performed to meet the deadline. In almost all cases it’s better to push back deadlines then to release a complete dud of a project. Take for example the video game Cyberpunk 2077 which was just released this month. The game had been pushed back multiple times prior and was finally set to release. The game itself contains an absurd number of bugs and glitches that are essentially game breaking. So much so that all the major providers, being Sony, Microsoft and Steam are all issuing refunds for the purchase of the game. The developer had a chose to either delay the project again or release it in the current state and that decision had a huge impact on the overall value of stock for the developer.