Project Two

Summary:

The testing I conducted aligned very well with the software requirements. The objects for the contact class have a unique ID, first name, last name, phone number, and address. The class also meets all the special requirements for the different fields such as the ID not being null and not being longer than 10 characters. The objects for the contact service can add contacts, delete contacts, as well as update the first name, last name, phone number, and address. The object for the task has a unique ID, name, and description. The objects for the task service can add tasks, delete tasks, and update the name and description.

The Junit testing helped to make sure every aspect of the code was being tested with the results for accuracy. The Junit tests all passed and had a coverage of 100% for the Contact.java and Task.java files. The coverage for the ContactService.java and the TaskService.java had a high coverage but I was not able to get it to 100%. I think this had to do with some of the code not executing the way I had thought it would. I will have to go back and edit it to make sure I have everything covered.

To ensure that my code is technically sound I make sure that the code is written and spaced out correctly. I also make sure to comment throughout the code to specify what is being done. This can be seen throughout all 8 of the files that were created. I also make sure that my naming of variables and functions are correct and not something that could be misidentified as something else. This goes along with the functions that were created and named to show the action it was doing.

To ensure that my code was efficient I made sure to create functions to do certain things rather than rewriting code over again. I also was able to use breaks in my loops in the ContactService.java and TaskService.java files. I also was efficient with my if, else if, else statements within the Contact.java and Task.java files. I used these statements to help make sure that the different parts of the object met the requirements. Such as the task name not being more that 20 characters and not being null.

Reflection:

There were different testing techniques that were employed for each of the milestones. One technique that was used was using JUnit tests to test our main files. In these JUnit files we can run our main program and make sure that what is supposed to happen does in fact happen. This testing is a form of “Static Testing techniques which are testing techniques that do not require the execution of a code base.” (GeeksforGeeks, 2021) Another technique was looking at the coverage of the main file was the JUnit test was be conducted. This coverage helped to show you the percentage of how well covered your main file was. If the coverage percentage was in the 95% to 100% then you covered all the aspects of the file being tested. This testing is a form of dynamic testing which is “testing techniques that are used to test the dynamic behavior of the application under test, that is by the execution of the code base.” (GeeksforGeeks, 2021)

There are other software techniques that were not used so far but could be beneficial with testing our code. One technique that could be used is use case testing. “Use Case represents the different ways in which the system can be used by the users. The use case divides the system behavior into scenarios, such that each scenario performs some useful action from the user’s point of view.” (SoftwareTestingHelp, 2022) This could be beneficial for seeing the user trying to do specify tasks within the system and how the work or do not work.

The static testing technique of running a JUnit test is good for testing things that you know will work and make the code run. The dynamic testing is better for seeing the coverage of you code and where you might need to add things to make it more complete. Use case testing could be good for seeing how the user would interact with the software and see were things could be improved.

When working on this project it is important to use caution to make sure that the code is what the client wants. You must make sure that the code is going to work as it is intended to and you try and not go against or add anything that is not wanted. You must make sure that the code is tested thoroughly. You also must try and limit your personal bias and not do something that seems right to you when it is not what the customer wants. This is important for making sure that all of you code is tested and you have not cut any corners.

References

Software Testing Help. *Popular software testing techniques with examples*. (2022, March 3). Retrieved April 3, 2022, from https://www.softwaretestinghelp.com/software-testing-techniques-2/

GeeksforGeeks. *Software testing techniques*. (2021, March 1). Retrieved April 3, 2022, from https://www.geeksforgeeks.org/software-testing-techniques/