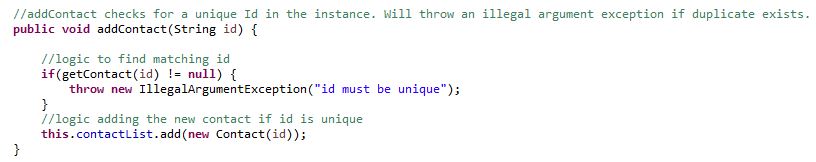
Project Two CS-320

By Dustin Runkel

# Feature Testing Summary

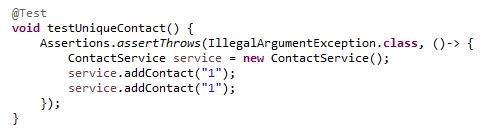
In this section, we will review the testing of the three features of the application. All features were built in Java and tested by utilizing Junit testing. The Junit test cases were derived from the requirements given to me by the requirements document, and tested accordingly. Junit testing is what its name implies, a unit testing method, part of the larger family of dynamic testing methods.

For the first feature, I tested the “contact” and “contactService” features. One of the requirements defines that the contact shall have a unique identifier. The code in question being tested is pictured here:



The get contact method searches the current list of contacts for a match, if a match is made, the contact is returned, which in this case would throw an exception. If the exception isn’t thrown, the contact is created.

Here is how I tested it:

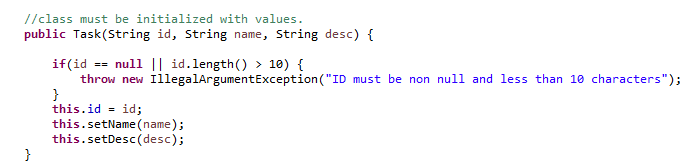


As you can see, the test asserts that if two contacts are made with the same id, an error should be thrown. In this case, there is a dichotomy of outcomes, and this test covers half, and an earlier test covers the other half. So, this is an example of testing aligning with the requirements for the “contactService” module.

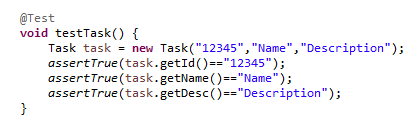
We can also attest to the quality of the testing by looking at the coverage. By looking at the coverage here, we can see a coverage of about 88%, which is good, considering exhaustive testing is impractical. From the coverage testing listed below, we can deduce that most instructions are covered, and the code has been thoroughly tested.



The next module I covered was the “taskService” module. This module manages tasks, requires a unique id, name, and description. For this module, I would say the best showcase of code efficiency is the method of task creation. Shown below:



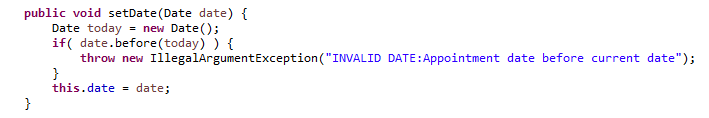
The task cannot be created without a valid id, and the class variables are set through the class mutators only, therefore saving time by not writing redundant code. Also, in testing the validity of the constructor, the validity of the mutator functions are also tested. Here is an example test, that verifies these functions:



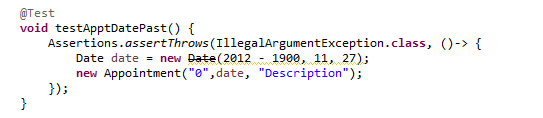
This test will let me know if any accessor or mutator is malfunctioning, as the test will fail if any case is not true. The only drawback of this test is the possibility of a typo, but that can be mitigated with “ctrl+c ctrl+v”. This satisfies the requirements explicitly, as the creation of the task object with valid variables is part of the requirements.

As for the coverage of this module, we can see that “TaskServiceTest” provides 100% test coverage of our “taskService” class, and “TaskTest” provides 100% test coverage of the “Task” class. This is very good coverage, and speaks to the quality of the tests done.

The next module is interesting, as the “appointmentService” requires the storage of a valid date object that is in the future, as well as a valid description and unique id. As per the requirements, a user should not be able to set a date in the future. This is the section of code responsible for tracking this:



This method only allows appointments to be updated and created for a future date. To test this piece of code, we first set a valid appointment, then we set an appointment for some date in the past. Here is the test I made for the code:



We can see, in this test, that the appointment is set 10 years in the past. Therefore, I assert that the code will throw an error, and in this case it does. This is the simplest way to test this piece of code, and is therefore efficient.

As for the test coverage of this module, both appointmentService and appointment have 100% test coverage.

# Developer Reflection

The testing techniques used for this project were briefly discussed above, but I will cover them here more clearly. Of all software testing methods, I would say that I employed dynamic methods the most, along with some static methods. To start, I’d like to tackle the static methods I covered and did not cover.

I would consider, over the course of this project, that I employed a few static testing methods, such as: informal reviews, walkthroughs, and technical reviews. I consider coding examination assignments that require specific examples technical reviews, as I am talking in depth about what code is being tested, and why. Walkthroughs and informal reviews, to me, fall under the same branch.

As for the methods not covered, I would say anything which requires a more formal review or data flow and control flow. These methods of testing were not used because they are impractical to simulate in the current school setting, or they are unneeded for the project.

In practice, static methods are good administrative tools for test review and requirements refining, and help developers everywhere deliver quality software. Next I will delve into the usage (or lack thereof) of dynamic testing tools.

Dynamic testing tools are tools used to find bugs in running code. In this project I employed component testing through JUnit testing. Dynamic testing offers many more testing options, such as integration testing, which tests several modules together to make sure they interoperate correctly.

# Mindset

To develop code and subsequently test the code is dangerous waters. In order to complete this task, one must tread very carefully. Personally, I know I caught myself writing tests that I knew would pass, without actually challenging the code I was testing. This behavior is almost impossible to avoid, but I had re-wrote the tests to be more appropriate.

I think it is important to have the ability to switch mindsets when testing your own code. I think it is so hard because there is this program you spent X hours making run, and now you have to break it. It was a hard mental barrier to cross. For me, I tried doing the program and the tests on separate days, to kind of separate it in my head. For some developers, that may not be an option, especially if the code fixes an immediate life-or-death scenario.

I tried to remain disciplined by, as I said, testing and coding on two separate days. In order to avoid technical debt, that is; rework done later, it is always easiest to hire contractors to do testing for you. I may be biased, because I work for a software testing company, but if you want quality, unbiased testing, that is the way to do it. Hiring a separate firm to test, strips bias, and allows testers (who only do testing), to find defects in your program. Yes, it might cost more, but the quality will save your company money and time.