LO3

# 3.1. Range of techniques

There were various ways in which the system was tested to achieve reliability. These were the following techniques used:

* **Junit testing (Unit testing) –** This was the first defence and usually caught most of the bugs that were found to appear within the code itself. Junit is a specified library within java that allows for accurate unit testing. Many of these tests only were made to validate a single method or a smaller part of a module/class. Larger integration tests were also carried using Junit tests.
* **Breakpoint debugging –** One of the most common ways of debugging is to use breakpoints. These were placed on any line of code that was deemed faulty or had a potential to have an underlying error. Once debug mode was entered, the program would stop at the specified line of code during runtime. While the program is in this paused state, variables, functions and different data structures able to be viewed one by one. Their contents are able to be reviewed and checked for any anomalies. Multiple breakpoints can be installed which then made it possible to jump between these two breakpoints.
* **Integration testing –** Conducted with the involvement of multiple classes and modules. These tests were used to determine how well 2 or more compoennets of the system worked. Best way to conduct these was to have several of these between varying methods. You would then have a couple final integration tests that would run across the bigger classes and modules.
* **System level testing –** System tests are one of the final tests that are conducted before a system can move to the next step of testing. These a conducted across the entire system including the majority (if not all) of the modules and classes used in the system. Purpose of this is to conclude how well the who system works together.
* **Step-through –** A less commonly used technique, however still useful. These were used during debugging and when things got really desperate. A step-through is similar to a breakpoint, but a bit less accurate. This is where the debugging software going though the whole code line by line.

# 3.2. Evaluation criteria for the adequacy of the testing

The evaluation of each technique goes as follows:

* **Junit testing (Unit testing) –** Very useful and quick. Use by far the most, however they lack detail. Harder to find more complicated bugs.
* **Breakpoint debugging –** Very useful but very slow. Breakpoints allow for a greater degree of detail and accuracy making the code practically transparent.
* **Integration testing –** Useful tests to conduct before a system level test. Integration testing is like a midway between unit tests and system tests. They allow for a decent amount of detail while also allowing for a good range of tests to conclude between the different functions.
* **System level testing –** Necessary tests. These tests should get conducted as testing the system as a whole is very important to achieving a good and reliable piece of software.
* **Step-through –** Not necessary. This technique is very rarely due to them being extremely slow and detail oriented. Only useful when all other options have been exhausted.

# 3.3. Results of testing

Each of the as forementioned techniques above were used in some regard during the testing process of the system. Overall, the results seem to be satisfactory. Each test highlighted a minimum of 1 or two error or bugs. There were a few issues regarding the customers card validation process. The expiry date was not being compared correctly, with the system discarding a majority of good orders with valid payment systems. This was identified by unit testing. There were also some errors recognised in the JSON that contained all the orders requested by customers. There were a few where none of the prices matched what the program calculated. It was found that the program was in fact correct as appose to the JSON. Eventually all tests began to success more and more, with all the Unit tests passing after a few debugging sessions.

A screenshot of a computer

Description automatically generated

# 3.4. Evaluation of the results

This results above prove that the tests conducted were adequate for the system that was being designed. The system is medium in size and complexity so no further testing techniques were necessary. Only a few systems wide tests were necessary with consistent results. This shows that the system is rigorous and reliable. If it were to be a slightly larger system, more integration tests would be required, however this was not the case in this situation.

Overall, the results seem to conclude that the conducted tests did their jobs successfully and manged to catch the majority of errors and bugs during testing.