LO3 - Software Testing

B196275

January 23, 2024

1 Test Files

These are the test files, and what requirement they correspond with.

FR3

`OrderHandlerUnitTests.java`:

- Contains unit-level tests to test each possible validation code of an order. Each possible validation code has multiples scenarios considered. This is used to verify the functionality of order validation.

FR4

`RestUnitTests.java`:

- Unit tests for each rest endpoint to ensure the rest client correctly returns data and handles different types of exceptions properly when thrown. Used to verify the functionality of the REST client.

`RestIntegrationTests.java`:

- Integration tests for each rest endpoint ensure our application properly integrates with the rest server. Used to verify the functionality of the REST client.

FR5

`LngLatHandlingUnitTests.java`:

- Contains unit-level tests to test each function works properly, this is important as PathFindingAlgorithm heavily uses it.

`PathFindingTests`:

- Contains integration-level tests to test the validity of paths generated for different cases.

NFR1

`PathFindingStatisticalTesting`:

- Contains a test that runs the program 100 times and checks if each execution is completed within 60 seconds.

2 Testing Techniques

Testing was primarily performed at the unit and integration level. Requirements that labelled as "extremely important" had both unit and integration tests to verify them.

- **Functional Testing (Systematic Approach)**: Functional testing was carried out following a systematic approach. This was carried out in FR3, FR4, and FR5 to verify specifications defined in the LO2 test plan against the constraints to prove that the components behaved as expected. Functional test cases were generated using techniques such as equivalence partitioning.

- **Structural Testing (Branch Testing)**: I used this alongside functional testing for testing my order validation. This allowed me to identify abnormal cases that may not appear in black-box testing. It helped me identify an edge case I had not anticipated with expiry dates that I should have accounted for in my functional testing. Further to this, I also use other coverage metrics like method and line coverage.
- **Performance Testing (Statistical Testing)**: This was used to verify NFR1, it allowed me to ensure that my program's runtime never exceeded 60 seconds.
- **Model-Based Testing**: This was used to generate test cases for my PathFindingAlgorithm. I modelled the method responsible for choosing neighbours, as this is responsible for ensuring each move chosen is valid. I used it to choose test cases such that every possible path is covered and allows me to verify that my algorithm is generating paths comprised of only valid moves. Refer to `decisionflowgraph.pdf` for the model I used.

3 Evaluation Criteria for Test Adequacy

1. All Tests Pass (Functional & Model-based testing):

   Passing all tests means that the application behaves as expected under various conditions.

2. Code Coverage (Structural Testing)

   (a) Coverage of order validation can leave us confident that it can correctly identify erroneous orders and applies to the verification of FR3.

   (b) Coverage of the REST client can leave us confident that it can correctly retrieve data from the REST server, and handle exceptions appropriately.

   (c) Coverage of LngLatHandler can leave us confident that it correctly carries out the geometric calculations it's supposed to do and applies to the verification of FR5.

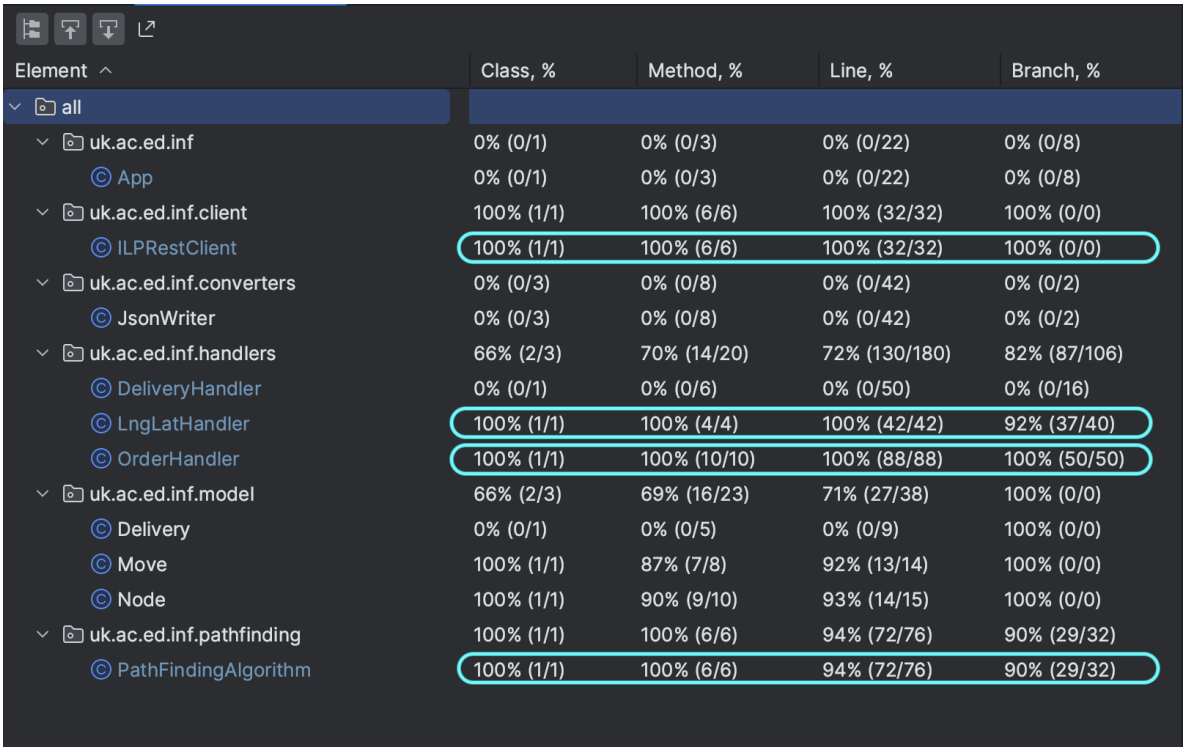
3. Runtime is always below 60 seconds (Statistical Testing).

4 Results of Testing

These are the following faults that were caught by the testing that I carried out:

- Faults were discovered in the "isInRegion" method where I initially considered points on the edges of the polygon as outside, which was wrong.
- Faults were discovered with the expiry date, where I was checking if it was valid concerning the actual date (today's date) when I should have been checking it against the order date.

Figure 1: All the Tests Passing (Unit and Integration Level)



Element	Class, %	Method, %	Line, %	Branch, %
all				
uk.ac.ed.inf	0% (0/1)	0% (0/3)	0% (0/22)	0% (0/8)
App	0% (0/1)	0% (0/3)	0% (0/22)	0% (0/8)
uk.ac.ed.inf.client	100% (1/1)	100% (6/6)	100% (32/32)	100% (0/0)
ILPRestClient	100% (1/1)	100% (6/6)	100% (32/32)	100% (0/0)
uk.ac.ed.inf.converters	0% (0/3)	0% (0/8)	0% (0/42)	0% (0/2)
JsonWriter	0% (0/3)	0% (0/8)	0% (0/42)	0% (0/2)
uk.ac.ed.inf.handlers	66% (2/3)	70% (14/20)	72% (130/180)	82% (87/106)
DeliveryHandler	0% (0/1)	0% (0/6)	0% (0/50)	0% (0/16)
LngLatHandler	100% (1/1)	100% (4/4)	100% (42/42)	92% (37/40)
OrderHandler	100% (1/1)	100% (10/10)	100% (88/88)	100% (50/50)
uk.ac.ed.inf.model	66% (2/3)	69% (16/23)	71% (27/38)	100% (0/0)
Delivery	0% (0/1)	0% (0/5)	0% (0/9)	100% (0/0)
Move	100% (1/1)	87% (7/8)	92% (13/14)	100% (0/0)
Node	100% (1/1)	90% (9/10)	93% (14/15)	100% (0/0)
uk.ac.ed.inf.pathfinding	100% (1/1)	100% (6/6)	94% (72/76)	90% (29/32)
PathFindingAlgorithm	100% (1/1)	100% (6/6)	94% (72/76)	90% (29/32)

Figure 2: Coverage across all relevant classes

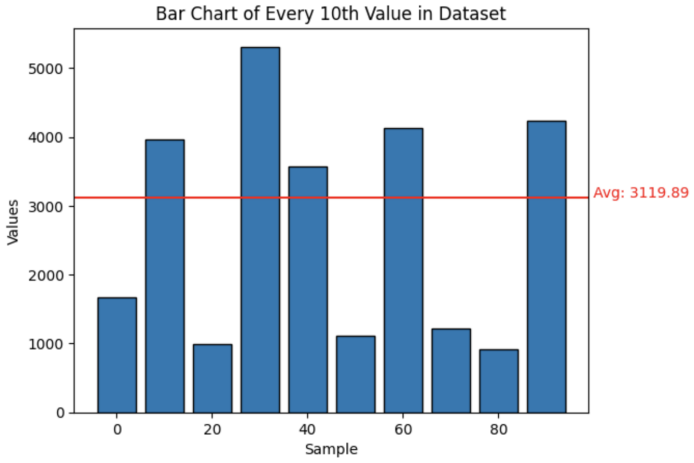


Figure 3: Bar Chart of Every Tenth Execution

5 Evaluation of Results

- As shown in Figure 1, all unit tests passed, and thus we can confidently say that our software functions as we expected as these tests were comprehensive and covered a large number of cases. This is further strengthened by the high method, line, and branch coverage as shown in Figure 2 achieved by structural testing (white-box testing) carried out to account for cases that may have not been covered by our functional tests (black-box testing).
- All the tests in PathFindingTests passed. The test cases were generated through a model-based approach, of which we wanted to cover all possible paths. And as all the tests passed, we

can be confident that we have full path coverage, and that our PathFindingAlgorithm properly integrates methods from LngLatHandler to generate and only select valid moves.

- As shown by Figure 3, the average execution time is 3.11989 seconds which is well below the 60 second threshold. Thus we can be confident that our system always runs quickly, and satisfies the stakeholder's want for an efficient system.

Overall, we can be confident that the testing carried out satisfies all the chosen requirements (FR3, FR4, FR5, and NFR1), that our program functions as expected, and that our testing has been effective.