LO3: Apply a wide variety of testing techniques and compute test coverage and yield according to a variety of criteria

Introduction

This document outlines the testing techniques employed during the PizzaDronz project and evaluates the quality of the testing process. Comparisons are drawn between the executed tests and the plan outlined in the LO2-Test_Planning_Document.pdf. The evaluation criteria are applied to determine how well the tests met the specified requirements, with a focus on their impact on satisfying these requirements.

The Tests

JUnit, a Java unit testing framework, was utilized for both unit and system testing as planned in LO2. Tests were distributed across multiple Java classes, especially for robustness testing. The specific tests and their corresponding requirements are as follows:

- AppTest.java:
  - Unit tests for validating arguments (Input Validation Tests).
  - System tests for program robustness, ensuring no crashes due to unhandled exceptions (Robustness Tests).
- OrderValidatorTest.java:
  - Unit and system tests for validating orders (Order Validation Tests).
  - System tests for associating orders with outcomes (Outcome Recording Test).
- JsonWritersTest.java:
  - System tests for validating the creation of accurate output files (Outcome Recording Test).
- RestServiceReaderTest.java, LngLatHandlerTest.java, PathFinderTest.java:
  - Robustness tests focusing on exception handling (Robustness Tests).

As planned, a significant amount of scaffolding was needed for testing, primarily involving mocked orders and restaurant data.

Range of Techniques

1. Functional Testing (Systematic)
   Test cases were derived from the requirements and the test plan. Unit-level tests verified specific constraints, while system-level tests assessed component behaviour.

2. Structural Testing
   This technique complemented functional testing by ensuring all components were covered. It proved especially useful for identifying edge cases, such as unusual causes of invalid card numbers, and verifying all possible branch conditions.

3. Combinatorial Testing
   Combinations of different inputs were tested to identify scenarios requiring special handling. This was extensively applied to functional requirements (input validation, order validation and outcome recording) and robustness testing by testing combinations like missing server data.

4. Model-Based Testing
   Although this method was considered, it was not implemented due to time constraints. It could have been useful for testing specific models (Movement and Orders) separately and as part of the system.

Evaluation Criteria for the Adequacy of Testing

1. Code Coverage
   Code coverage measures how much of the code is executed by the tests, including control and data flow across different paths. This was assessed in two ways:

   - Overall test coverage: Evaluated the robustness of the entire program.

   - Component-level coverage: Focused on satisfying input validation, order validation and outcome recording.

2. Test Passing
   Passing the tests indicated correct program behaviour according to the requirements.

3. Test Diversity
   A diverse range of tests was developed, covering various input types and combinations to ensure the program's correctness. However, since the inputs were manually generated, some edge cases may have been missed.

Results of Testing

The application successfully passed all the designed tests (100%), including both unit and system tests for all the requirements. The testing process also uncovered some previously unnoticed issues, such as:

- Null checks for order fields.

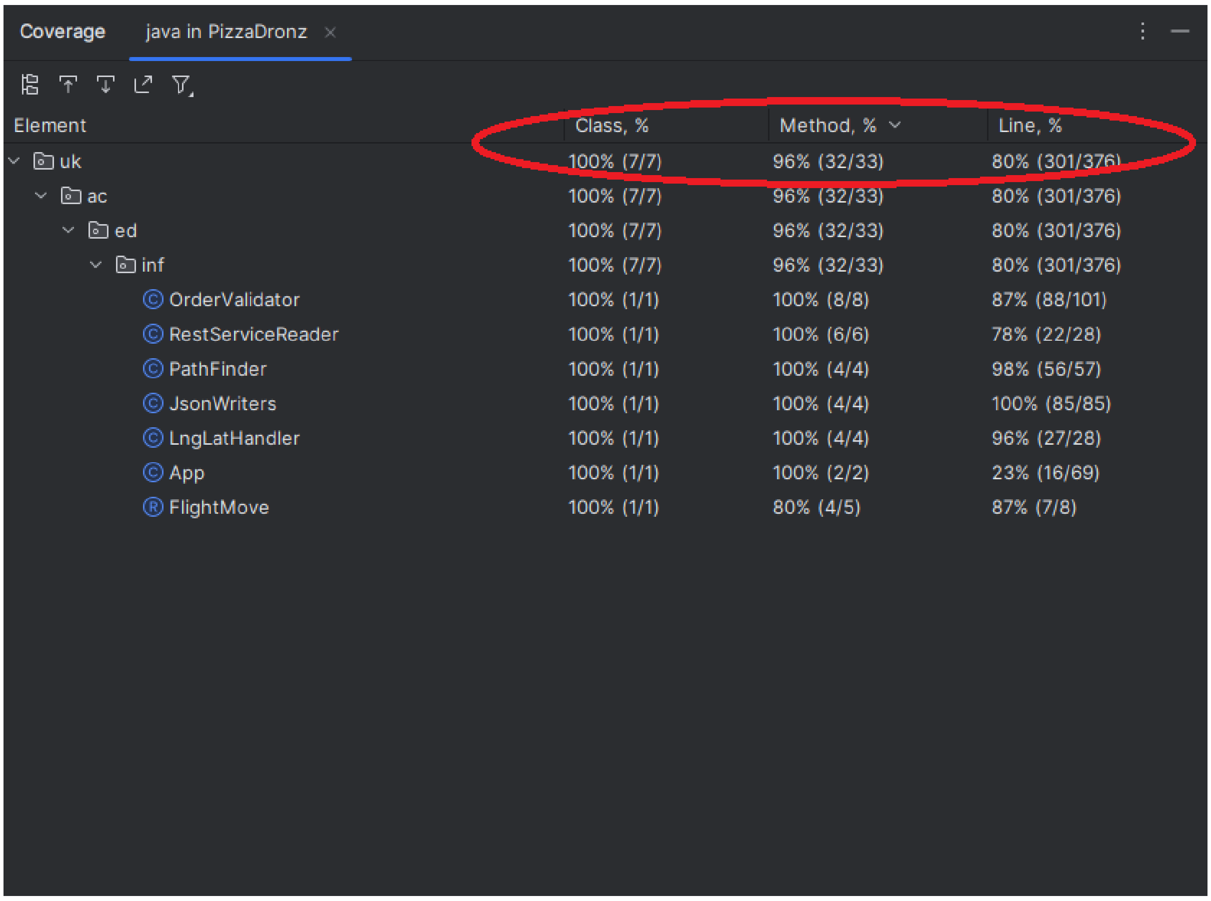
- Handling of invalid or missing dates.

Evaluation of the Results

1. Code Coverage:

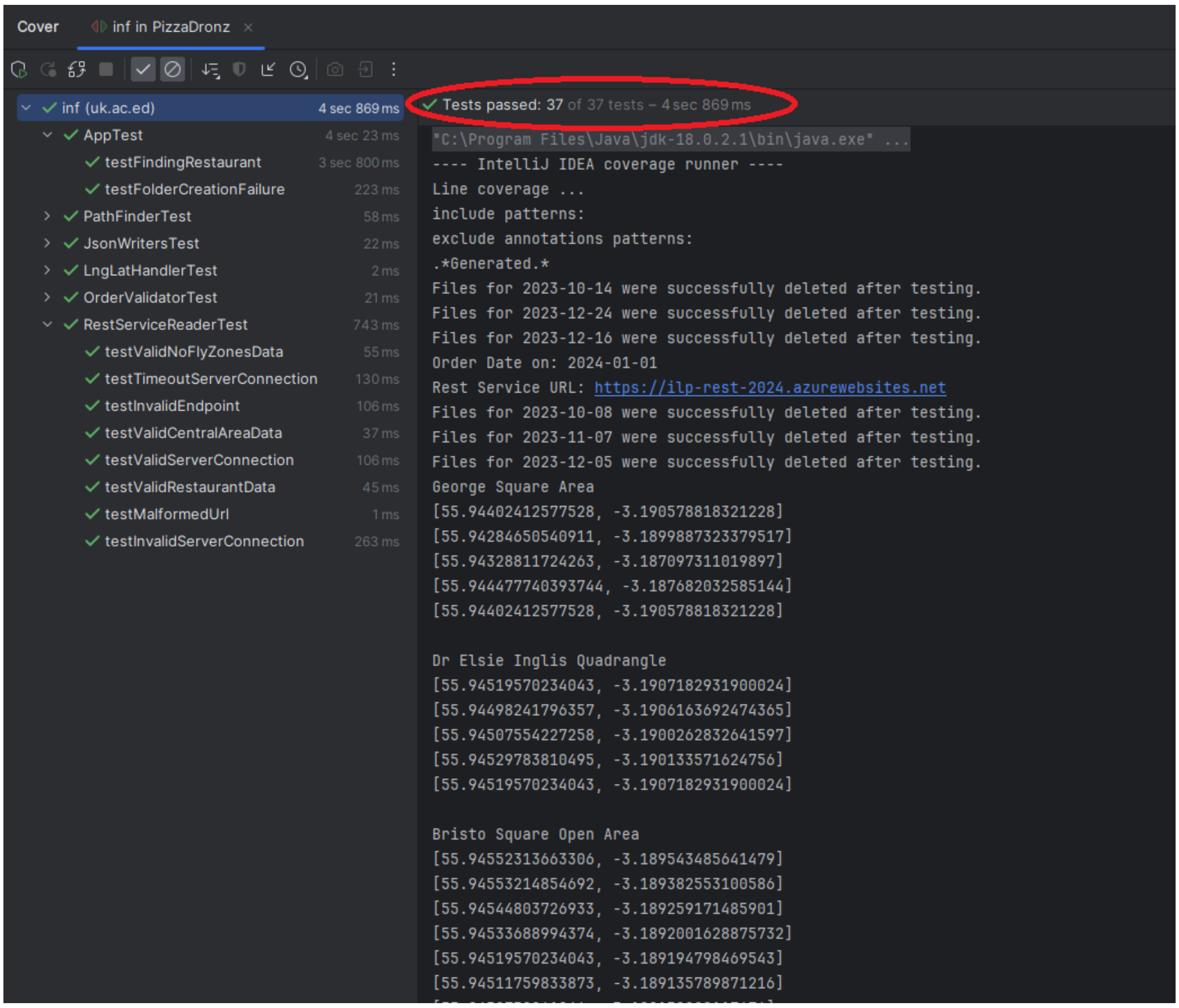
- Overall coverage: 80% line coverage and 96% method coverage.
- Component-level coverage: 100% line coverage for input validation, order validation and outcome recording.

2. Test Passing:
    - All tests passed successfully.
3. Test Diversity:
    - Test scenarios accounted for a wide variety of inputs and edge cases. However, since the inputs were manually generated, some rare scenarios might have been overlooked.



Element	Class, %	Method, %	Line, %
uk	100% (7/7)	96% (32/33)	80% (301/376)
ac	100% (7/7)	96% (32/33)	80% (301/376)
ed	100% (7/7)	96% (32/33)	80% (301/376)
inf	100% (7/7)	96% (32/33)	80% (301/376)
OrderValidator	100% (1/1)	100% (8/8)	87% (88/101)
RestServiceReader	100% (1/1)	100% (6/6)	78% (22/28)
PathFinder	100% (1/1)	100% (4/4)	98% (56/57)
JsonWriters	100% (1/1)	100% (4/4)	100% (85/85)
LngLatHandler	100% (1/1)	100% (4/4)	96% (27/28)
App	100% (1/1)	100% (2/2)	23% (16/69)
FlightMove	100% (1/1)	80% (4/5)	87% (7/8)

Figure 1: Test coverage after executing all tests



```
inf (uk.ac.ed)                          4 sec 869 ms
  AppTest                               4 sec 23 ms
    testFindingRestaurant               3 sec 800 ms
    testFolderCreationFailure           223 ms
  PathFinderTest                        58 ms
  JsonWritersTest                       22 ms
  LngLatHandlerTest                     2 ms
  OrderValidatorTest                    21 ms
  RestServiceReaderTest                 743 ms
    testValidNoFlyZonesData             55 ms
    testTimeoutServerConnection         130 ms
    testInvalidEndpoint                 106 ms
    testValidCentralAreaData            37 ms
    testValidServerConnection           106 ms
    testValidRestaurantData             45 ms
    testMalformedUrl                    1 ms
    testInvalidServerConnection         263 ms
```

```
Tests passed: 37 of 37 tests – 4 sec 869 ms
"C:\Program Files\Java\jdk-18.0.2.1\bin\java.exe" ...
---- IntelliJ IDEA coverage runner ----
Line coverage ...
include patterns:
exclude annotations patterns:
.*Generated.*
Files for 2023-10-14 were successfully deleted after testing.
Files for 2023-12-24 were successfully deleted after testing.
Files for 2023-12-16 were successfully deleted after testing.
Order Date on: 2024-01-01
Rest Service URL: https://ilp-rest-2024.azurewebsites.net
Files for 2023-10-08 were successfully deleted after testing.
Files for 2023-11-07 were successfully deleted after testing.
Files for 2023-12-05 were successfully deleted after testing.
George Square Area
[55.94402412577528, -3.190578818321228]
[55.94284650540911, -3.1899887323379517]
[55.94328811724263, -3.187097311019897]
[55.944477740393744, -3.187682032585144]
[55.94402412577528, -3.190578818321228]

Dr Elsie Inglis Quadrangle
[55.94519570234043, -3.1907182931900024]
[55.94498241796357, -3.1906163692474365]
[55.94507554227258, -3.1900262832641597]
[55.94529783810495, -3.190133571624756]
[55.94519570234043, -3.1907182931900024]

Bristo Square Open Area
[55.94552313663306, -3.189543485641479]
[55.94553214854692, -3.189382553100586]
[55.94544803726933, -3.189259171485901]
[55.94533688994374, -3.1892001628875732]
[55.94519570234043, -3.189194798469543]
[55.94511759833873, -3.189135789871216]
```

Figure 2: All test passing

Conclusion

Based on the results and evaluation, the testing process was effective and improved confidence in the application's behaviour. The conducted tests satisfied all the requirements, demonstrating the robustness and functionality of the system.