Learning Objective 3

Apply a Wide Variety of Testing Techniques and Compute Test Coverage and Yield According to a Variety of Criteria

The Tests

To test the PizzaDronz system, I created four test classes. They are as follows, which also details what they comprise of:

AppTest.java:

- This test file primarily covers **R1**, **R5** and **R6**, but it also contains tests that cover every single requirement, as System tests are included that runs the entire program.
- It contains Unit tests for the argument validation of **R1**, and for some aspects of **R5**.
- It contains System tests for **R1**, **R2** and **R5**.
- It contains performance tests for **R6.**
- Some Integration tests are present for components involving multiple requirements. These also test **R2**.

LngLatHanderTest.java:

- This test file is for **R4.**
- It contains a mixture of Unit tests for individual functions of LngLatHandler.java.

OrderValidatorTest.java:

- This test file is for **R3**.
- It contains Unit tests for the individual functions of OrderValidator.java.
- It contains System level tests for complete order validation.

RestTest.java:

- This test file corresponds to **R2.**
- It contains Unit tests for individual data points from the REST API.
- There are also Integration tests for the interaction between PizzaDronz and the server.

The tests were created using Java's JUnit library. Significant scaffolding was used for testing **R3**, with some for **R4** too. This involved generating (valid) random order values or locations, as described in previous documents for LO1 and LO2. Code instrumentation was also required for testing **R1** and **R2**, to ensure the proper output assertions were present in the code for PizzaDronz.

3.1 Range of Techniques

Functional Tests

A range of tests have been implemented as outlined in the LO2 document. This has provided numerous methods to correctly verify the behaviour of the PizzaDronz system at all levels of requirements.

Structural Tests

Tests were implemented to verify the functionality of PizzaDronz against both its specification, and the requirements highlighted in LO1, specifically those selected to focus on (**R1** – **R6**). These were used to ensure maximum code coverage, to prevent certain aspects of the system from being omitted in the development process. Structural testing complements functional testing, and it is therefore an important method to implement.

Performance Tests
Performance tests were implemented to verify **R6**. These were simple timing tests to verify that the system can execute in under 60 seconds both in isolation, and repeatedly, to confirm the standard, regular performance of the system.

3.2 Evaluation Criteria for the Adequacy of the Testing

Passing of Tests
Simply put, if a test passes, then it is extremely likely that the corresponding component of the system is working and can guarantee that its respective requirement has been met (unless there is an error in the test, which is even more of a problem). The application's behaviour should be as expected, should it pass the corresponding test(s).

Code Coverage
Code coverage determines how much of a system's code has actually been utilised by the tests themselves. A higher code coverage usually indicates more thorough testing, as a greater percentage of the system has been run by the implemented tests. Usually the aim is for 100% coverage, at least of the methods involved in a program. If something is not covered by a test, its correctness cannot be verified and faults cannot be detected, as these parts of the code have not been run and therefore not encountered by any test.

3.3 Results of Testing

The PizzaDronz system was completed with 100% of the implemented tests being passed. All Unit, System, Integration, Robustness and Performance tests that were created were passed, for all six of the tested requirements. The tests helped to streamline the way areas of the code were written, such as argument handling within App.java.

Runtime: Over 100 iterations, the average runtime of the PizzaDronz system was 1222 milliseconds, or 1.222 seconds (Fig 1). This is well below the 60 second cap, and therefore a strong pass for the performance tests.

Some time was given to mutation testing, which provided some helpful results for development, however these were implemented after the completion of the code so no changes were made following the running of these tests. They were most helpful for OrderValidator.java and LngLatHandler.java, demonstrating high code coverage and a large number of mutants being generated. A significant number of mutants were eliminated, outlining that the functions have been thoroughly implemented with sufficient error handling in mind. See Figs (2-4) for further detail.

3.4 Evaluation of the Results

Passing of Tests
As mentioned, 100% of the implemented tests passed. Including repeated tests, this involved 2349 tests. However, it would be more helpful to break these down into core tests:

- AppTest.java: 14 Tests

- LngLatHandlerTest.java: 7 Tests (Included repeated tests)
- OrderValidatorTest.java: 24 Tests (Included repeated tests)
- RestTest.java: 8 Tests

Please see figs. (5 - 8) for further information.

Code Coverage

AppTest.java:

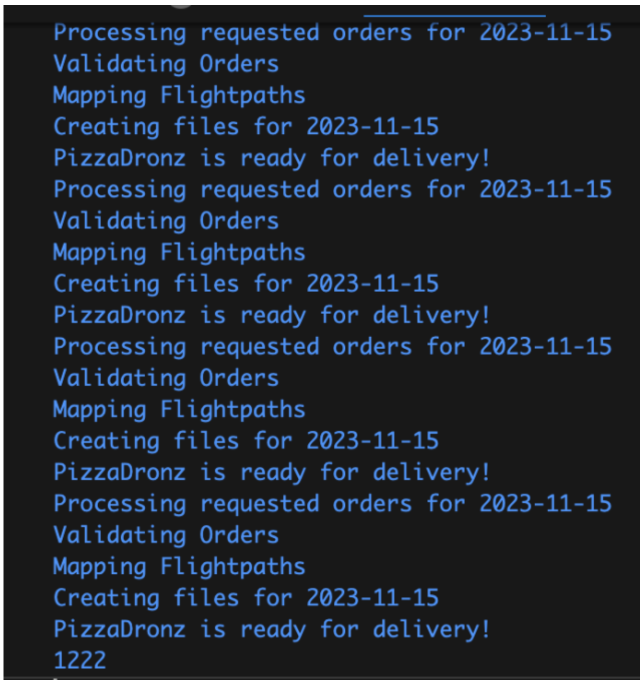
- 100% class coverage
- 97.5% method coverage (one method was not tested. It was re-written in the codebase and is never called by the PizzaDronz system. It is functionally identical to a method that was tested)
- 82% line coverage

RestTest.java:

- Only tested within the class RestReader.java:
  - 100% method coverage

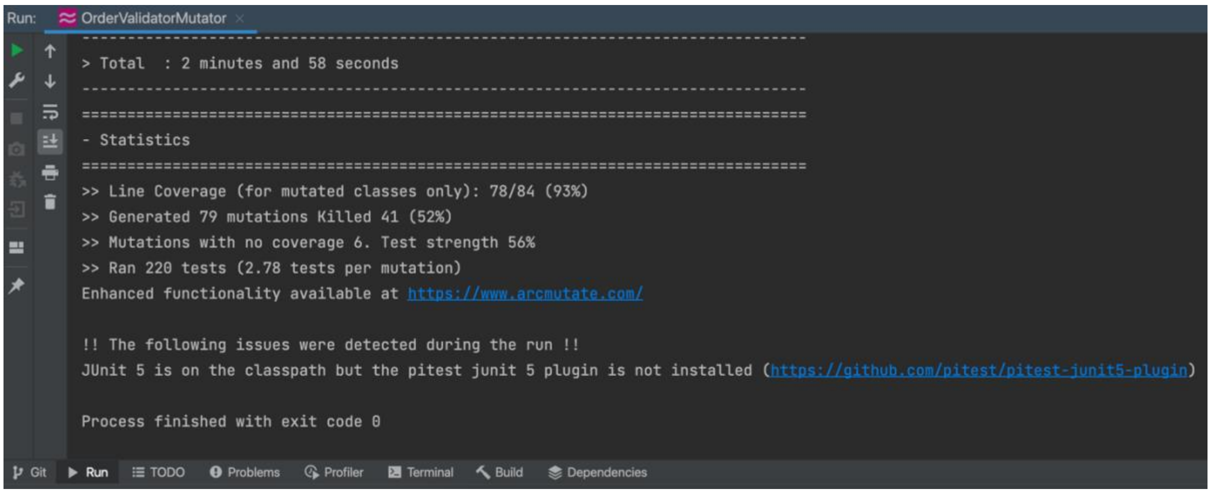
Please see figs. (5 - 8) for further information.

Unfortunately, LngLatHandlerTest.java and OrderValidatorTest.java would not run on IntelliJ, due to the Maven project setup, so statistics are not available for these test classes. However, through inspection 100% method coverage was achieved by both of these in their respective classes, although this cannot be guaranteed as accurate as VSCode does not provide coverage statistics.
There is another limitation in that since all scaffolding was manually generated, there could some small errors or omissions. However, these tests were implemented extremely carefully and thoroughly, so this is an unlikely scenario.



```
Processing requested orders for 2023-11-15
Validating Orders
Mapping Flightpaths
Creating files for 2023-11-15
PizzaDronz is ready for delivery!
Processing requested orders for 2023-11-15
Validating Orders
Mapping Flightpaths
Creating files for 2023-11-15
PizzaDronz is ready for delivery!
Processing requested orders for 2023-11-15
Validating Orders
Mapping Flightpaths
Creating files for 2023-11-15
PizzaDronz is ready for delivery!
Processing requested orders for 2023-11-15
Validating Orders
Mapping Flightpaths
Creating files for 2023-11-15
PizzaDronz is ready for delivery!
1222
```

Fig 1 – Results of Performance Testing; 1222 milliseconds average runtime over 100 iterations

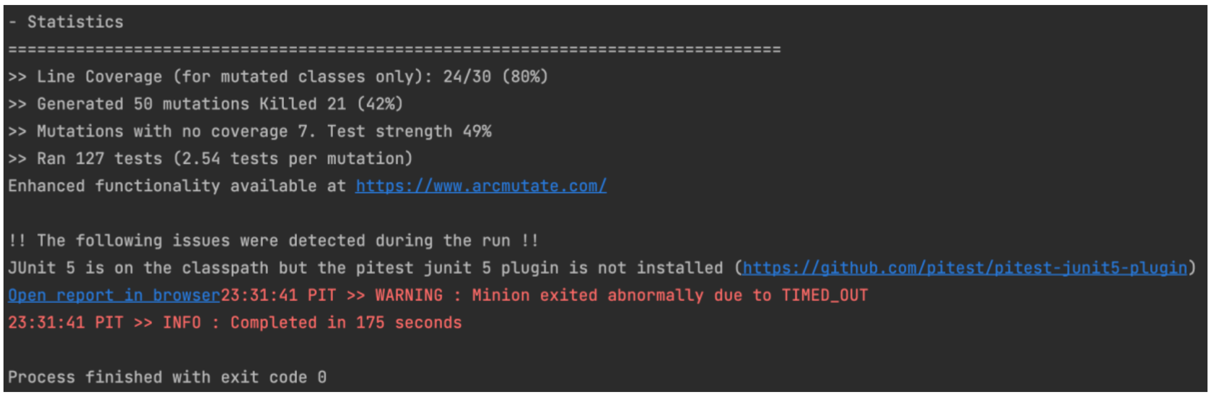


```
--------------------------------------------------------------------------------
> Total  : 2 minutes and 58 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Line Coverage (for mutated classes only): 78/84 (93%)
>> Generated 79 mutations Killed 41 (52%)
>> Mutations with no coverage 6. Test strength 56%
>> Ran 220 tests (2.78 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/

!! The following issues were detected during the run !!
JUnit 5 is on the classpath but the pitest junit 5 plugin is not installed (https://github.com/pitest/pitest-junit5-plugin)

Process finished with exit code 0
```

Fig 2 – Mutant Testing for OrdarValidation.java

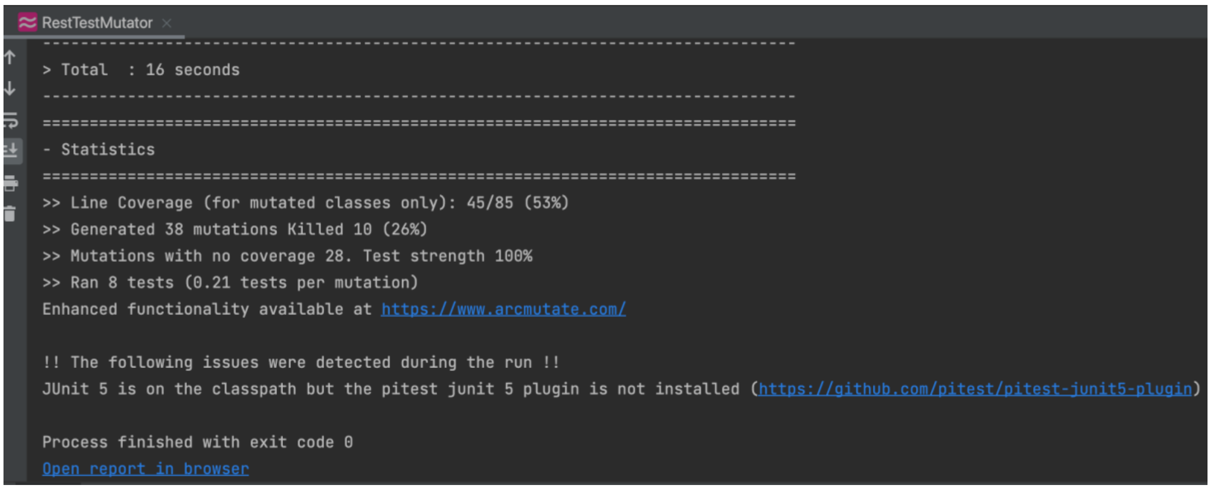


```
- Statistics
================================================================================
>> Line Coverage (for mutated classes only): 24/30 (80%)
>> Generated 50 mutations Killed 21 (42%)
>> Mutations with no coverage 7. Test strength 49%
>> Ran 127 tests (2.54 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/

!! The following issues were detected during the run !!
JUnit 5 is on the classpath but the pitest junit 5 plugin is not installed (https://github.com/pitest/pitest-junit5-plugin)
Open report in browser23:31:41 PIT >> WARNING : Minion exited abnormally due to TIMED_OUT
23:31:41 PIT >> INFO : Completed in 175 seconds

Process finished with exit code 0
```

Fig 3 – Mutant Testing for LngLatHandler.java

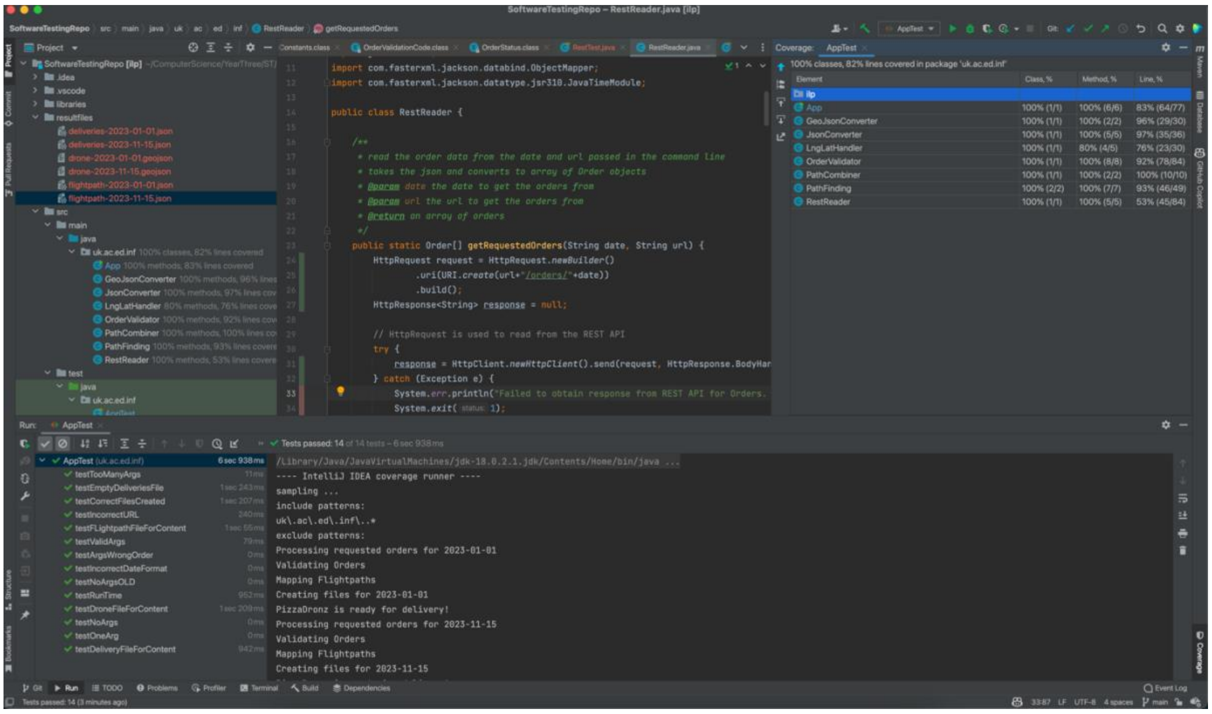


```
--------------------------------------------------------------------------------
> Total  : 16 seconds
--------------------------------------------------------------------------------
================================================================================
- Statistics
================================================================================
>> Line Coverage (for mutated classes only): 45/85 (53%)
>> Generated 38 mutations Killed 10 (26%)
>> Mutations with no coverage 28. Test strength 100%
>> Ran 8 tests (0.21 tests per mutation)
Enhanced functionality available at https://www.arcmutate.com/

!! The following issues were detected during the run !!
JUnit 5 is on the classpath but the pitest junit 5 plugin is not installed (https://github.com/pitest/pitest-junit5-plugin)

Process finished with exit code 0
Open report in browser
```

Fig 4 – Mutant Testing for RestTest.java


SoftwareTestingRepo – RestReader.java [ilp]
SoftwareTestingRepo › src › main › java › uk › ac › ed › inf › RestReader › getRequestedOrders
Constants.class
OrderValidationCode.class
OrderStatus.class
RestTest.java
RestReader.java
AppTest
Project
SoftwareTestingRepo [ilp]
.idea
.vscode
libraries
resultfiles
deliveries-2023-01-01.json
deliveries-2023-11-15.json
drone-2023-01-01.geojson
drone-2023-11-15.geojson
flightpath-2023-01-01.json
flightpath-2023-11-15.json
src
main
java
uk.ac.ed.inf 100% classes, 82% lines covered
App 100% methods, 83% lines covered
GeoJsonConverter 100% methods, 96% lines
JsonConverter 100% methods, 97% lines cov
LngLatHandler 80% methods, 76% lines cove
OrderValidator 100% methods, 92% lines cov
PathCombiner 100% methods, 100% lines co
PathFinding 100% methods, 93% lines covere
RestReader 100% methods, 53% lines covere
test
java
uk.ac.ed.inf
Run: AppTest
Git
Run
TODO
Problems
Profiler
Terminal
Build
Dependencies
Event Log
Tests passed: 14 (3 minutes ago)
33:67 LF UTF-8 4 spaces main

```java
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

public class RestReader {

    /**
     * read the order data from the date and url passed in the command line
     * takes the json and converts to array of Order objects
     * @param date the date to get the orders from
     * @param url the url to get the orders from
     * @return an array of orders
     */
    public static Order[] getRequestedOrders(String date, String url) {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url+"/orders/"+date))
                .build();
        HttpResponse<String> response = null;

        // HttpRequest is used to read from the REST API
        try {
            response = HttpClient.newHttpClient().send(request, HttpResponse.BodyHar
        } catch (Exception e) {
            System.err.println("Failed to obtain response from REST API for Orders.
            System.exit( status: 1);
```

Coverage: AppTest

100% classes, 82% lines covered in package 'uk.ac.ed.inf'

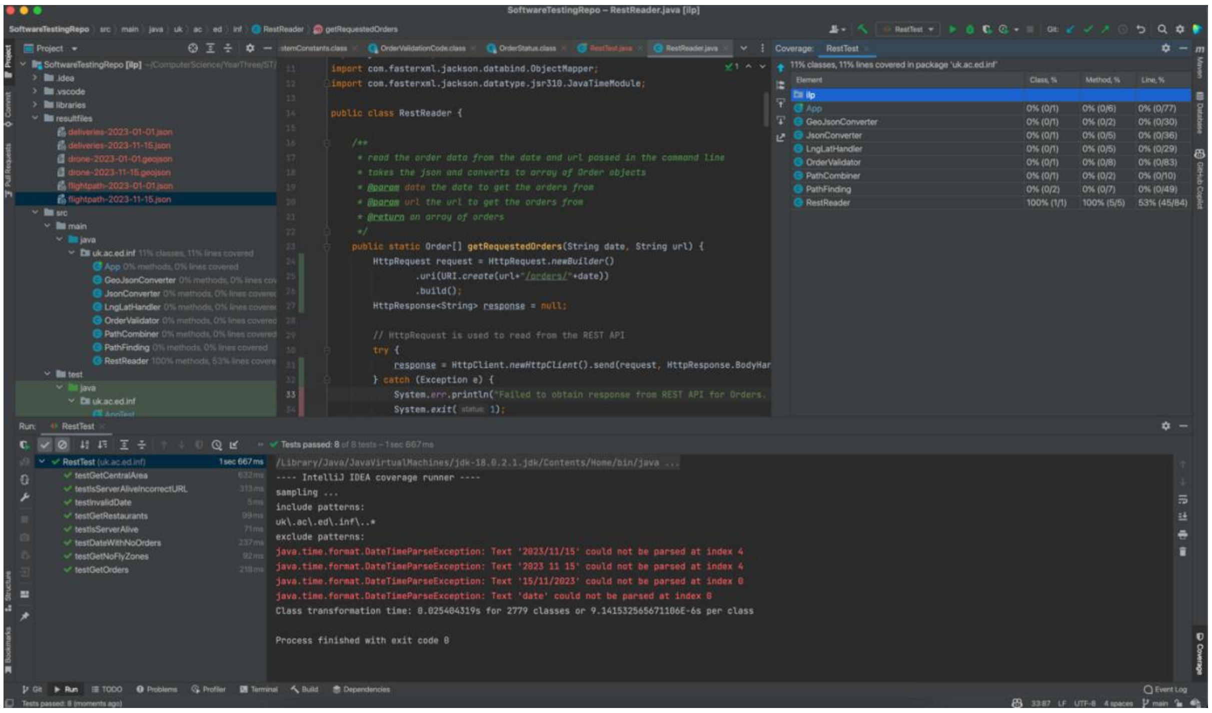
Element	Class, %	Method, %	Line, %
ilp			
App	100% (1/1)	100% (6/6)	83% (64/77)
GeoJsonConverter	100% (1/1)	100% (2/2)	96% (29/30)
JsonConverter	100% (1/1)	100% (5/5)	97% (35/36)
LngLatHandler	100% (1/1)	80% (4/5)	76% (23/30)
OrderValidator	100% (1/1)	100% (8/8)	92% (78/84)
PathCombiner	100% (1/1)	100% (2/2)	100% (10/10)
PathFinding	100% (2/2)	100% (7/7)	93% (46/49)
RestReader	100% (1/1)	100% (5/5)	53% (45/84)

Tests passed: 14 of 14 tests – 6 sec 938 ms

```
AppTest (uk.ac.ed.inf)              6 sec 938 ms
    testTooManyArgs                 11 ms
    testEmptyDeliveriesFile         1 sec 243 ms
    testCorrectFilesCreated         1 sec 207 ms
    testIncorrectURL                240 ms
    testFLightpathFileForContent    1 sec 55 ms
    testValidArgs                   79 ms
    testArgsWrongOrder              0 ms
    testIncorrectDateFormat         0 ms
    testNoArgsOLD                   0 ms
    testRunTime                     952 ms
    testDroneFileForContent         1 sec 209 ms
    testNoArgs                      0 ms
    testOneArg                      0 ms
    testDeliveryFileForContent      942 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
uk\.ac\.ed\.inf\..*
exclude patterns:
Processing requested orders for 2023-01-01
Validating Orders
Mapping Flightpaths
Creating files for 2023-01-01
PizzaDronz is ready for delivery!
Processing requested orders for 2023-11-15
Validating Orders
Mapping Flightpaths
Creating files for 2023-11-15
```

Fig 5 – Test Passing and Code Coverage statistics for AppTest.java


SoftwareTestingRepo – RestReader.java [ilp]
SoftwareTestingRepo › src › main › java › uk › ac › ed › inf › RestReader › getRequestedOrders
stemConstants.class
OrderValidationCode.class
OrderStatus.class
RestTest.java
RestReader.java
RestTest
Project
SoftwareTestingRepo [ilp]
.idea
.vscode
libraries
resultfiles
deliveries-2023-01-01.json
deliveries-2023-11-15.json
drone-2023-01-01.geojson
drone-2023-11-15.geojson
flightpath-2023-01-01.json
flightpath-2023-11-15.json
src
main
java
uk.ac.ed.inf 11% classes, 11% lines covered
App 0% methods, 0% lines covered
GeoJsonConverter 0% methods, 0% lines cov
JsonConverter 0% methods, 0% lines covered
LngLatHandler 0% methods, 0% lines covered
OrderValidator 0% methods, 0% lines covered
PathCombiner 0% methods, 0% lines covered
PathFinding 0% methods, 0% lines covered
RestReader 100% methods, 53% lines covere
test
java
uk.ac.ed.inf
Run: RestTest
Git
Run
TODO
Problems
Profiler
Terminal
Build
Dependencies
Event Log
Tests passed: 8 (moments ago)
33:67 LF UTF-8 4 spaces main

```java
import com.fasterxml.jackson.databind.ObjectMapper;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;

public class RestReader {

    /**
     * read the order data from the date and url passed in the command line
     * takes the json and converts to array of Order objects
     * @param date the date to get the orders from
     * @param url the url to get the orders from
     * @return an array of orders
     */
    public static Order[] getRequestedOrders(String date, String url) {
        HttpRequest request = HttpRequest.newBuilder()
                .uri(URI.create(url+"/orders/"+date))
                .build();
        HttpResponse<String> response = null;

        // HttpRequest is used to read from the REST API
        try {
            response = HttpClient.newHttpClient().send(request, HttpResponse.BodyHar
        } catch (Exception e) {
            System.err.println("Failed to obtain response from REST API for Orders.
            System.exit( status: 1);
```

Coverage: RestTest

11% classes, 11% lines covered in package 'uk.ac.ed.inf'

Element	Class, %	Method, %	Line, %
ilp			
App	0% (0/1)	0% (0/6)	0% (0/77)
GeoJsonConverter	0% (0/1)	0% (0/2)	0% (0/30)
JsonConverter	0% (0/1)	0% (0/5)	0% (0/36)
LngLatHandler	0% (0/1)	0% (0/5)	0% (0/29)
OrderValidator	0% (0/1)	0% (0/8)	0% (0/83)
PathCombiner	0% (0/1)	0% (0/2)	0% (0/10)
PathFinding	0% (0/2)	0% (0/7)	0% (0/49)
RestReader	100% (1/1)	100% (5/5)	53% (45/84)

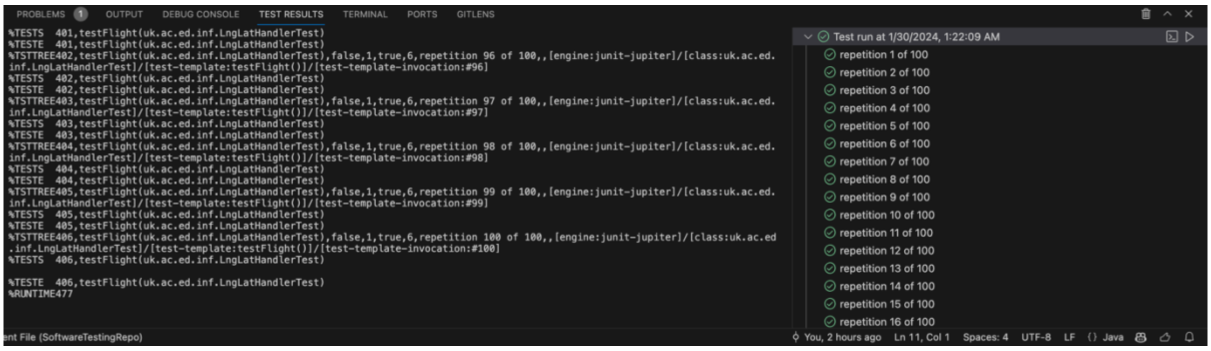
Tests passed: 8 of 8 tests – 1 sec 667 ms

```
RestTest (uk.ac.ed.inf)             1 sec 667 ms
    testGetCentralArea              632 ms
    testIsServerAliveIncorrectURL   313 ms
    testInvalidDate                 5 ms
    testGetRestaurants              99 ms
    testIsServerAlive               71 ms
    testDateWithNoOrders            237 ms
    testGetNoFlyZones               92 ms
    testGetOrders                   218 ms
```

```
/Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
---- IntelliJ IDEA coverage runner ----
sampling ...
include patterns:
uk\.ac\.ed\.inf\..*
exclude patterns:
java.time.format.DateTimeParseException: Text '2023/11/15' could not be parsed at index 4
java.time.format.DateTimeParseException: Text '2023 11 15' could not be parsed at index 4
java.time.format.DateTimeParseException: Text '15/11/2023' could not be parsed at index 0
java.time.format.DateTimeParseException: Text 'date' could not be parsed at index 0
Class transformation time: 0.025404319s for 2779 classes or 9.141532565671106E-6s per class

Process finished with exit code 0
```

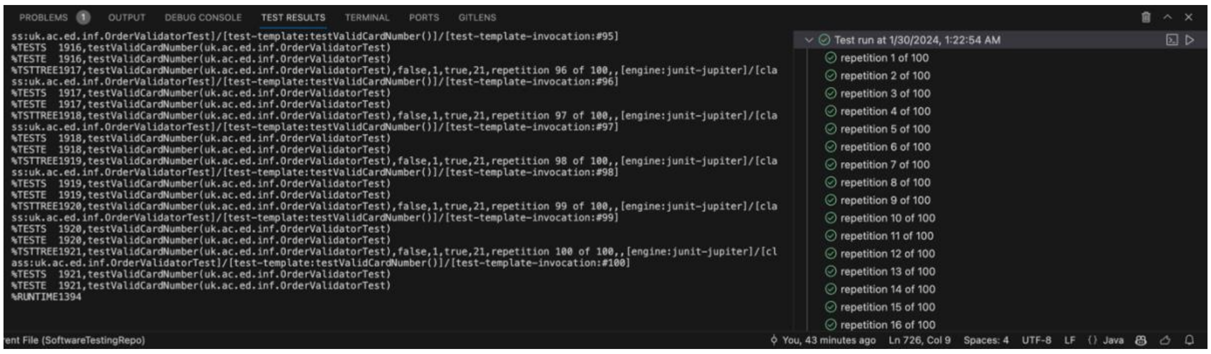
Fig 6 – Test Passing and Code Coverage statistics for RestTest.java



```
%TESTS  401,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TESTE  401,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TSTTREE402,testFlight(uk.ac.ed.inf.LngLatHandlerTest),false,1,true,6,repetition 96 of 100,,[engine:junit-jupiter]/[class:uk.ac.ed.
inf.LngLatHandlerTest]/[test-template:testFlight()]/[test-template-invocation:#96]
%TESTS  402,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TESTE  402,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TSTTREE403,testFlight(uk.ac.ed.inf.LngLatHandlerTest),false,1,true,6,repetition 97 of 100,,[engine:junit-jupiter]/[class:uk.ac.ed.
inf.LngLatHandlerTest]/[test-template:testFlight()]/[test-template-invocation:#97]
%TESTS  403,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TESTE  403,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TSTTREE404,testFlight(uk.ac.ed.inf.LngLatHandlerTest),false,1,true,6,repetition 98 of 100,,[engine:junit-jupiter]/[class:uk.ac.ed.
inf.LngLatHandlerTest]/[test-template:testFlight()]/[test-template-invocation:#98]
%TESTS  404,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TESTE  404,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TSTTREE405,testFlight(uk.ac.ed.inf.LngLatHandlerTest),false,1,true,6,repetition 99 of 100,,[engine:junit-jupiter]/[class:uk.ac.ed.
inf.LngLatHandlerTest]/[test-template:testFlight()]/[test-template-invocation:#99]
%TESTS  405,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TESTE  405,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%TSTTREE406,testFlight(uk.ac.ed.inf.LngLatHandlerTest),false,1,true,6,repetition 100 of 100,,[engine:junit-jupiter]/[class:uk.ac.ed
.inf.LngLatHandlerTest]/[test-template:testFlight()]/[test-template-invocation:#100]
%TESTS  406,testFlight(uk.ac.ed.inf.LngLatHandlerTest)

%TESTE  406,testFlight(uk.ac.ed.inf.LngLatHandlerTest)
%RUNTIME477
```

Fig 7 – Test Passing statistics for LngLatHandlerTest.java



```
ss:uk.ac.ed.inf.OrderValidatorTest]/[test-template:testValidCardNumber()]/[test-template-invocation:#95]
%TESTS  1916,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TESTE  1916,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TSTTREE1917,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest),false,1,true,21,repetition 96 of 100,,[engine:junit-jupiter]/[cla
ss:uk.ac.ed.inf.OrderValidatorTest]/[test-template:testValidCardNumber()]/[test-template-invocation:#96]
%TESTS  1917,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TESTE  1917,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TSTTREE1918,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest),false,1,true,21,repetition 97 of 100,,[engine:junit-jupiter]/[cla
ss:uk.ac.ed.inf.OrderValidatorTest]/[test-template:testValidCardNumber()]/[test-template-invocation:#97]
%TESTS  1918,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TESTE  1918,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TSTTREE1919,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest),false,1,true,21,repetition 98 of 100,,[engine:junit-jupiter]/[cla
ss:uk.ac.ed.inf.OrderValidatorTest]/[test-template:testValidCardNumber()]/[test-template-invocation:#98]
%TESTS  1919,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TESTE  1919,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TSTTREE1920,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest),false,1,true,21,repetition 99 of 100,,[engine:junit-jupiter]/[cla
ss:uk.ac.ed.inf.OrderValidatorTest]/[test-template:testValidCardNumber()]/[test-template-invocation:#99]
%TESTS  1920,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TESTE  1920,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TSTTREE1921,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest),false,1,true,21,repetition 100 of 100,,[engine:junit-jupiter]/[cl
ass:uk.ac.ed.inf.OrderValidatorTest]/[test-template:testValidCardNumber()]/[test-template-invocation:#100]
%TESTS  1921,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%TESTE  1921,testValidCardNumber(uk.ac.ed.inf.OrderValidatorTest)
%RUNTIME1394
```

Fig 8 – Test Passing statistics for OrderValidatorTest.java