This report is for student s1915409

The raw overall mark is (72.5/75)
The awarded mark is (72.5/75)

The awarded mark includes a deduction for any days late, following the standard School and University scheme: the deduction is 5% of the maximum mark per calendar day up to 7 days. For submissions more than 7 days late after the original deadlne, 0 is awarded.

Overall and awarded marks returned to the ITO are rounded to the nearest integer.

The submission deadline was 16:00 on Friday 3rd December 2021.

Submission time-stamp: 2021-12-10-12-47-18
Number of further days allowed for an extension and/or learning adjustments: 7
Days late after any further days allowed: 0

Overall comments (may be empty):
BEGIN COMMENTS

END COMMENTS

As described on pages 19 and 20 of the coursework instructions, marks are awarded for four aspects:

1. the report,
2. JavaDoc documentation,
3. the implementation,
4. correctness and effectiveness.

See below for your mark for each aspect and associated comments.

1. REPORT (20/20)

The instructions asked you to submit a report with two sections providing a clear and accurate description of your implementation. Information on what was asked for in each section and comments on what you provided in your report are as follows:

1.1 Software architecture description.

From instructions:

> This section provides a description of the software architecture of your application. Your application is made up of a collection of Java classes; explain why you identified these classes as being the right ones for your application.

BEGIN COMMENTS
Your software architecture section provides a good description of why classes were identified.
You have defined classes to structure your solution. It is evident that you have thought about decomposing your application code into classes; this was a good design decision.
You have defined several classes which represent the semantic objects of the problem (for example, classes like Drone or similar). This was a very good design decision because it makes for a good object-oriented design.
You have defined data classes which will allow you to import JSON data into your application; this was a good design decision.
Your naming of classes is descriptive and consistent.
END COMMENTS

1.2 Drone control algorithm

From instructions:

> This section explains the algorithm which is used by your drone to control their flight around the locations of interest and back to the start location of their flight, while avoiding all of the no-fly zones and trying to maximise the drone's score on the sampled average percentage monetary value metric described on page 10.
>
> This section of your report should contain two graphical figures (similar to Figure 6 in this document) which have been made using the http://geojson.io website, rendering the flights of your drone on two dates of your choosing on top of the background provided by the file testing/all.geojson.

BEGIN COMMENTS
Your drone control algorithm section provides a good description of the elements of your algorithm.
You provided a good description of how your drone maximises its score on the sampled average percentage monetary value metric.
You provided a good description of how your drone manages the flight home to the start location of the flightpath.
You provided a good description of how your drone manages its flightpath to avoid the No-Fly zones of the map.

As required, you included two plots of the flight path for your drone.
END COMMENTS

1.3 Further comments

(May be blank)
BEGIN COMMENTS

END COMMENTS

2. JAVADOC DOCUMENTATION (10/10)

From the instructions:

> You are to document your Java code using the JavaDoc format. Your documentation should be informative and useful, clearly describing the classes, fields and methods of your project. Your JavaDoc code should compile to give a valid HTML document.

BEGIN COMMENTS
You have provided good JavaDoc documentation giving helpful information about the classes and methods which you have implemented.
END COMMENTS

3. IMPLEMENTATION (28.9/30)

From the instructions:

> Your submission should faithfully implement the drone behaviour described above, hosted in a framework which allows the drone to make a maximum of 1500 moves on any day. Your application should be usably efficient, without significant stalls while executing. Your code should be readable and clear, making use of private values, variables and functions, and encapsulat- ing code and data structures.

The mark awarded here is based on:

+ a review of your submitted code,
+ whether your code compiled and was packaged correctly using maven, and whether tests ran as expected,
+ the extent to which your code run-times were under 60s and tests of core features of your code all passed,
+ the extent to which a set of tests of basic features of your code were successful.

Comments from the review of your submitted code:
BEGIN COMMENTS
You have made good use of visibility modifiers such as private or protected to encapsulate fields and methods.
You have made good use of the final modifier to mark fields as being immutable.
You have made good use of exception handling to catch the exceptions which are thrown by I/O operations.
You do not use the Java throws mechanism to avoid exception handling. This is good practice.
You used the Gson parser to parse the JSON documents with WhatThreeWords data obtained from the web server. This is a good approach.
You used the GeoJSON SDK to read and write GeoJSON files. This is the right way to consume and create these files.
You made a connection to the database and accessed the metadata about tables in the database.
You catch the SQLExceptions which can be thrown by database operations. This is good practice, allowing you to generate more precise error messages.
Your code does not contain any TODO or FIXME comments for issues which should have been resolved before submission. This is good to see.
Your code does not contain commented-out Java code which should have been removed before submission. This is good to see.
END COMMENTS

Comments on compiling your code using Maven (may be empty if no issues):
BEGIN COMMENTS

END COMMENTS

Comments from running your code (may be empty if no issues):
BEGIN COMMENTS

END COMMENTS

See Section 6 below for further information on the testing of your code.

4. CORRECTNESS AND EFFECTIVENESS (13.6/15)

From the instructions:

> The flightpaths in the flightpath database table generated by your application will be tested to ensure that the moves made by the drone are legal according to the description given above, considering the drone confinement area and the no-fly zones described above. The drone's score on the sampled average percentage

monetary value metric described on page 10 will be considered: the higher the score the better the quality of the drone.

The mark awarded here is based on:

+ the extent to which a set of tests of advanced features of your code were successful,
+ the sampled average percentage monetary value, counting a test's monetary value as zero if any core feature checks failed in that test

See Section 6 below for further information on the testing of your code.

5. SUBMITTED GEOJSON DRONE FILES
---

1. drone-07-07-2022.geojson has 809 points
2. drone-02-02-2022.geojson has 317 points
3. drone-05-05-2022.geojson has 566 points
4. drone-06-06-2022.geojson has 570 points
5. drone-03-03-2022.geojson has 295 points
6. drone-04-04-2022.geojson has 336 points
7. drone-01-01-2022.geojson has 312 points
8. drone-08-08-2022.geojson has 880 points
9. drone-10-10-2022.geojson has 938 points
10. drone-12-12-2022.geojson has 1049 points
11. drone-09-09-2022.geojson has 768 points
12. drone-11-11-2022.geojson has 1130 points

6. REPORT ON TESTS

Following the instructions, first your code was compiled and packaged into a jar file using:

```
mvn package
```

Then 5 tests on your code were carried out by running it on 5 dates using a command similar to:

```
java -jar target/ilp-1.0-SNAPSHOT.jar 01 01 2022 80 5127
```

and analysing the database entries and GeoJSON file generated in each case. Runs were carried out on student.compute using storage under /tmp for the database and web servers, as this gave much better performance than when using storage on the DICE AFS file system.

A. SUMMARY ACROSS ALL TESTS

Basic features successfully checked: 99.1%
Advanced features successfully checked: 96.0%
Fast run-time score: 80.0%
Average monetary value: 80.0%

Each test looked at 38 design features. A feature was considered successfully checked if it was checked at least once and the check passed every time.

Features are divided into two categories:

+ `Basic features' which concern restricted parts of output database tables or the GeoJSON output file.

+ `Advanced features' which concern checking consistency across whole outputs and between the different outputs, the inputs from the provided initial database, and the inputs from the web server.

See Section C below for a list of the features and an indication of which are basic and which are advanced.

The percentage of successful checks of basic and advanced features were calculated separately, and these contributed respectively to your Implementation and your Correctness and Effectiveness marks presented above in Sections 3 and 4.

Some features are considered 'core' features. If the checks of all core features were successful in a test, the run-time and monetary value for the test were considered reliable.

+ The `Fast run-time score' is the fraction of tests where the test is considered reliable and the run-time is below 60s.

+ The `Average monetary value' is the average of the monetary values from the individual tests, except that a test's monetary value is taken as 0 if the test is considered unreliable.

Section C below indicates which features are considered to be core.

The Fast run-time score contributed towards your Implementation mark and the Average monetary value towards your Correctness and Effectiveness mark.

B. INFORMATION ON EACH TEST

Test 1:
Date: 01-01-2022
Web server port: 9890
Database server port: 9868
Return code (0 if test terminated normally): 0
Run-time: 1.8s
Run reliable (all core feature checks passed): YES
Check log file: check-01-01-2022.log
Number of rows in deliveries table: 4
Number of rows in flightpath table: 311
Number of points in GeoJSON drone output file: 312
Monetary value: 100.0%

Test 2:
Date: 06-06-2022
Web server port: 9891
Database server port: 9869
Return code (0 if test terminated normally): 0
Run-time: 2.2s
Run reliable (all core feature checks passed): YES
Check log file: check-06-06-2022.log
Number of rows in deliveries table: 9
Number of rows in flightpath table: 569
Number of points in GeoJSON drone output file: 570
Monetary value: 100.0%

Test 3:
Date: 12-12-2022
Web server port: 9892
Database server port: 9870
Return code (0 if test terminated normally): 0
Run-time: 2.2s
Run reliable (all core feature checks passed): YES
Check log file: check-12-12-2022.log
Number of rows in deliveries table: 15
Number of rows in flightpath table: 1048
Number of points in GeoJSON drone output file: 1049
Monetary value: 100.0%

Test 4:
Date: 06-06-2023
Web server port: 9893
Database server port: 9871
Return code (0 if test terminated normally): 0
Run-time: 2.4s
Run reliable (all core feature checks passed): YES
Check log file: check-06-06-2023.log
Number of rows in deliveries table: 21

Number of rows in flightpath table: 1287
Number of points in GeoJSON drone output file: 1288
Monetary value: 100.0%

Test 5:
Date: 12-12-2023
Web server port: 9894
Database server port: 9872
Return code (0 if test terminated normally): 0
Run-time: 2.4s
Run reliable (all core feature checks passed): NO
Check log file: check-12-12-2023.log
Number of rows in deliveries table: 21
Number of rows in flightpath table: 1500
Number of points in GeoJSON drone output file: 1501
Monetary value: 88.4%

For further information on a given test, consult the respective log file attached to this email.

C. INFORMATION ON EACH FEATURE

Format of each line: "a b. c (d/e) [f] <g> h" where
a - feature category (C=core, B=basic, A=advanced)
b - feature number
c - fraction of tests for which the feature check is successful
d - number of times across all tests that feature check passes
e - number of times across all tests that feature exercised
f - test numbers for when feature check fails
g - feature name
h - further description of feature (optional)

To investigate further why a feature check has failed in some test, consult the log files for the indicated tests, and search for mention of the feature name. Each time a feature is exercised and the check fails, an entry is added to the relevant test's log giving further information on the failure.

In the feature names and descriptions, a drone is considered to have made a stop at a pick-up or drop-off location if it hovers at one of the pick-up or drop-off locations valid for the order named in the relevant flightpath table row.

Deliveries table features

CB 1. 100% (5/ 5) [   ] <delivs.read: table present>
CB 2. 100% (5/ 5) [   ] <delivs.read: columns found> Column names are as expected
CB 3. 100% (70/ 70) [   ] <delivs.read: orderNo not null>
CB 4. 100% (70/ 70) [   ] <delivs.read: deliveredTo not null>
CB 5. 100% (70/ 70) [   ] <delivs.read: costInPence not null>
CA 6. 100% (70/ 70) [   ] <delivs: orderNo expected> Is valid for the test date
CA 7. 100% (70/ 70) [   ] <delivs: orderNo not repeated>
CA 8. 100% (70/ 70) [   ] <delivs: deliveredTo good> Matches order information from web server
CA 9. 100% (70/ 70) [   ] <delivs: costInPence good> Matches order information from web server

Basic flightpath table features

CB 10. 100% (5/ 5) [   ] <fpath.read: table present>
CB 11. 100% (5/ 5) [   ] <fpath.read: columns found> Column names are as expected
CB 12. 100% (4715/4715) [   ] <fpath.read: angle not null>
CB 13. 100% (5/ 5) [   ] <fpath: not empty> More than 0 rows in table
CB 14. 100% (5/ 5) [   ] <fpath: within limit> At most 1500 rows in table
CB 15. 100% (5/ 5) [   ] <fpath: drone starts at AT>
CB 16. 80% (4/ 5) [5   ] <fpath: drone ends at AT>
 B 17. 100% (4710/4710) [   ] <fpath.move: end start same> End of one move is same as start of next
CB 18. 100% (4710/4710) [   ] <fpath.move: end start close> Distance is under 0.00015
CB 19. 100% (4715/4715) [   ] <fpath.move: start inside> Move start is within confinement area
CB 20. 100% (4715/4715) [   ] <fpath.move: end inside>
CB 21. 100% (4715/4715) [   ] <fpath.move: good angle>
CB 22. 100% (4715/4715) [   ] <fpath.move: good end> End is correctly calculated
CA 23. 100% (4465/4465) [   ] <fpath.move: avoids no-fly zone>
CA 24. 100% (5/ 5) [   ] <fpath.seg: flightpath has stops>

Flightpath table features relating to delivery of an order

CA 25. 100% (71/ 71) [   ] <fpath.seg: order not repeated>
CA 26. 100% (71/ 71) [   ] <fpath.seg: order valid> Order is valid for test date
CA 27. 80% (70/ 71) [5   ] <fpath.seg: order in deliveries>
CA 28. 100% (71/ 71) [   ] <fpath.seg: pick-up stops visited>
CA 29. 80% (70/ 71) [5   ] <fpath.seg: drop-off stop visited>
CA 30. 80% (70/ 71) [5   ] <fpath.seg: enough stops> Drone stops at least once along flightpath
CA 31. 100% (5/ 5) [   ] <fpath.seg: all deliveries made> As promised in deliveries table

GeoJSON drone file features

 B 32. 100% (5/ 5) [   ] <dronefile: file present>
 B 33. 100% (5/ 5) [   ] <dronefile: JSON syntax good>
 B 34. 100% (5/ 5) [   ] <dronefile: GeoJSON structure good>
 B 35. 100% (5/ 5) [   ] <dronefile: List<Point> non-empty>
 B 36. 100% (4465/4465) [   ] <dronefile: move length good> Move has distance 0 or 0.00015
 A 37. 100% (5/ 5) [   ] <dronefile: trail lengths same> When compared with flightpath with hover moves removed

A 38. 100% (4470/4470) [ ] <dronefile: trail points match> When compared with flightpath with hover moves removed

END OF REPORT