**Lab 1 Report**

Name: 許哲維

Student ID: 111598066

Date: 2023.03.08

1. **Test Plan**
   1. **Test requirements**

The Lab 1 requires to (1) select **20 methods** from **6 classes** of the GeoProject, (2) design Unit test cases based on the experience or intuition for the selected methods, (3) develop test scripts to implement the test cases, (4) execute the test script on the selected methods, and (5) report the test results.

In particular, based on the statement coverage criterion, the **test requirements** for Lab 1 are to design test casesfor each selected method so that “*each statement of the method will be covered by at least one test case* and *the minimum statement coverage is 50%*”.

* 1. **Strategy**

To satisfy the test requirements listed in Section 1, a proposed strategy is to

1. select those public methods that are easy to understand and have primitive types of input and output parameters (if possible).
2. set the objective of the minimum statement coverage to be 50% initially and (if necessary) adjust the objective based on the time available.
3. learn the necessary skills and tools as soon as possible.
4. design the test cases for those selected methods by considering
   1. the possible **valid values** and **combinations** of the input parameters.
   2. the **boundary values** of the input parameters.
   3. **Test activities**

To implement the proposed strategy, the following activities are planned to perform.

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Activity Name** | **Plan hours** | **Schedule Date** |
| 1 | Study GeoProject | 3 | 2023.03.01 |
| 2 | Learn JUnit | 3 | 2023.03.02 |
| 3 | Design test cases for the selected methods | 3 | 2023.03.02 |
| 4 | Implement test cases | 4.5 | 2023.03.06 |
| 5 | Perform test | 4.5 | 2023.03.07 |
| *6* | Complete Lab1 report | 2 | 2023.03.08 |

* 1. **Success criteria**

All test cases designed for the selected methods must pass (or "85% of all test cases must pass) and *the statement coverage should have achieved at least 50%.*

1. **Test Design**

To fulfill the test requirements listed in section 1.1, the following methods are selected and corresponding test cases are designed.

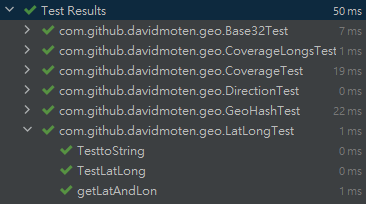
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Class** | **Method** | **Test Objective** | **Inputs** | **Expected Outputs** |
| 1 | Base32 | encodeBase32() |  | 75324, 4 | 29jw |
| 2 | Base32 | encodeBase32() |  | 75324, -4 | -29jw |
| 3 | Base32 | encodeBase32() |  | 4 | 000000000004 |
| 4 | Base32 | decodeBase32() |  | 29jw | 75324 |
| 5 | Base32 | getCharIndex() |  | 0 | 0 |
| 6 | Base32 | padLeftWithZerosToLength() |  | Lucy, 5 | 0Lucy |
| 7 | Base32 | padLeftWithZerosToLength() |  | JenJen, 6 | JenJen |
| 8 | Coverage | getHashes() | Coverage |  | Katayama, Jenjen, not, Chinchin. |
| 9 | Coverage | getRatio() | Coverage |  | 4.0 |
| 10 | Coverage | getHashLength() | Coverage |  | 9 |
| 11 | Coverage | toString() | Coverage |  | Coverage [hashes=[Chinchin., not, Katayama, Jenjen], ratio=4.0] |
| 12 | CoverageLongs | getHashLength() | CoverageLongs |  | 3 |
| 13 | CoverageLongs | getHashLength() | CoverageLongs |  | 0 |
| 14 | CoverageLongs | toString() | CoverageLongs |  | "Coverage [hashes="+c.getHashes()+", ratio="+c.getRatio()+"]" |
| 15 | CoverageLongs | getCount() | CoverageLongs |  | 1 |
| 16 | Direction | opposite() |  | Direction.RIGHT | Direction.LEFT |
| 17 | Direction | opposite() |  | Direction.LEFT | Direction.RIGHT |
| 18 | Direction | opposite() |  | Direction.TOP | Direction.BOTTOM |
| 19 | Direction | opposite() |  | Direction.BOTTOM | Direction.TOP |
| 20 | GeoHash | adjacentHash() |  | "29jw",Direction.BOTTOM,1 | "29jt" |
| 21 | GeoHash | neighbours() |  | "29jw" | "29jq" |
| 22 | GeoHash | hashLengthToCoverBoundingBox() |  | 2,2,1,1 | 2 |
| 23 | GeoHash | coverBoundingBoxLongs() |  | 0.1,0.1,0.1,0.1,1 | 1 |
| 24 | LatLong | getLat() | LatLong |  | 6 |
| 25 | LatLong | toString() | LatLong |  | "LatLong [lat="+lat+", lon="+lon+"]" |

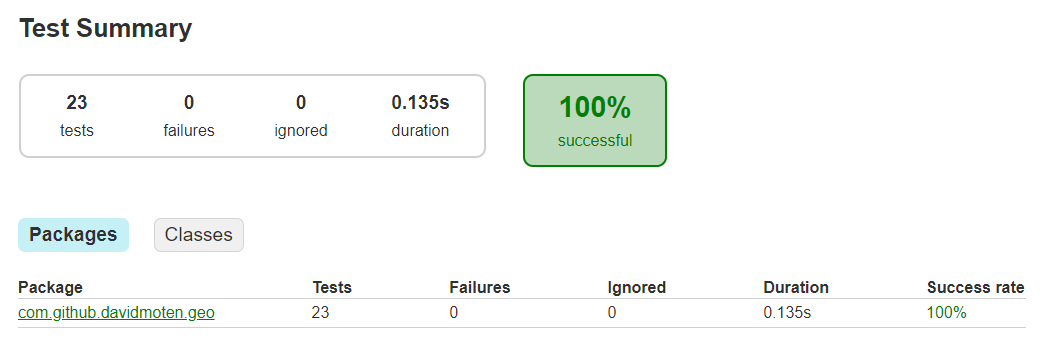
1. **Test Implementation**

The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 7 selected test cases are given below. The rest of test script implementations can be found in the [link](https://github.com/Daaata/STV) (or JUnit files).

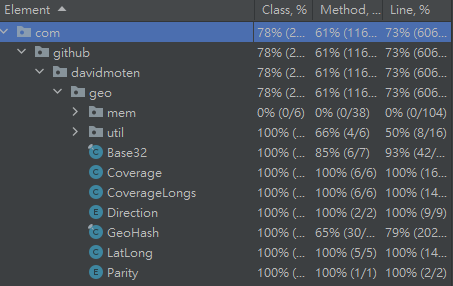
|  |  |  |
| --- | --- | --- |
| **No.** | **Test method** | **Source code** |
| 1 | encodeBase32() |  |
| 2 | TesttoString() |  |
| 3 | opposite() |  |
| 4 | getRatio() |  |
| 5 | padLeftWithZerosToLength() |  |
| 6 | decodeBase32() |  |
| 7 | adjacentHash() |  |

1. **Test Results**
   1. **JUnit test result snapshot**

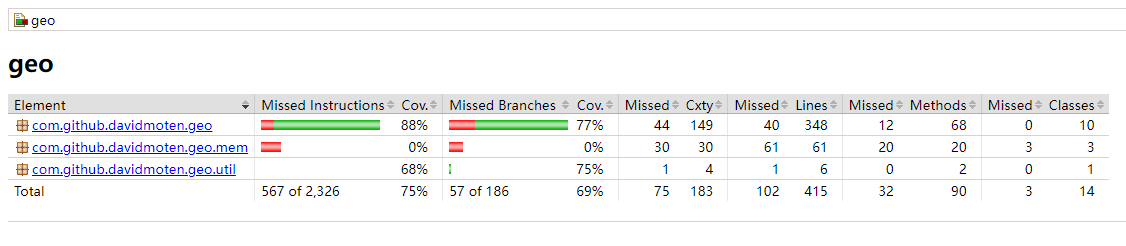
****

****

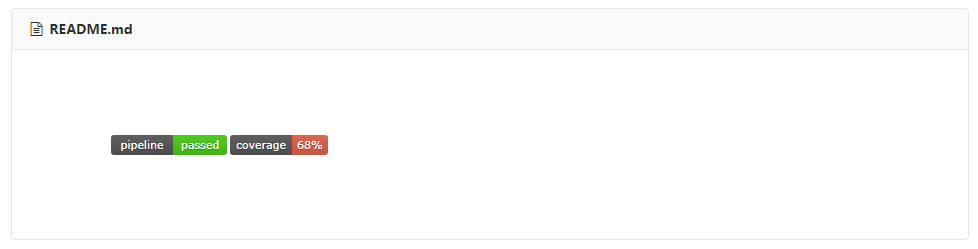
* 1. **Code coverage snapshot**
* Coverage of each selected method

****

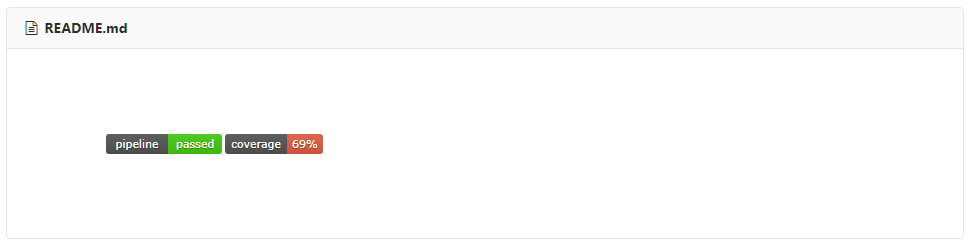
* Total coverage

****

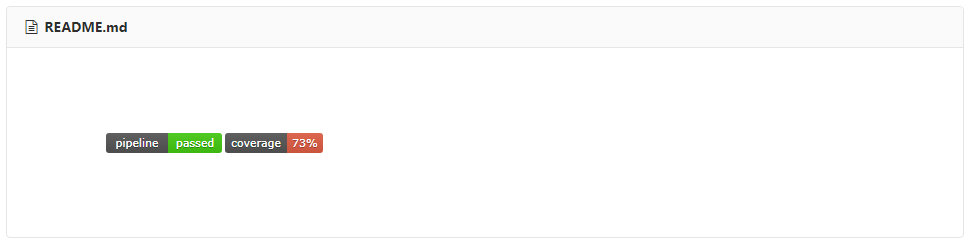
* 1. **CI result snapshot (3 iterations for CI)**
* CI#1

****

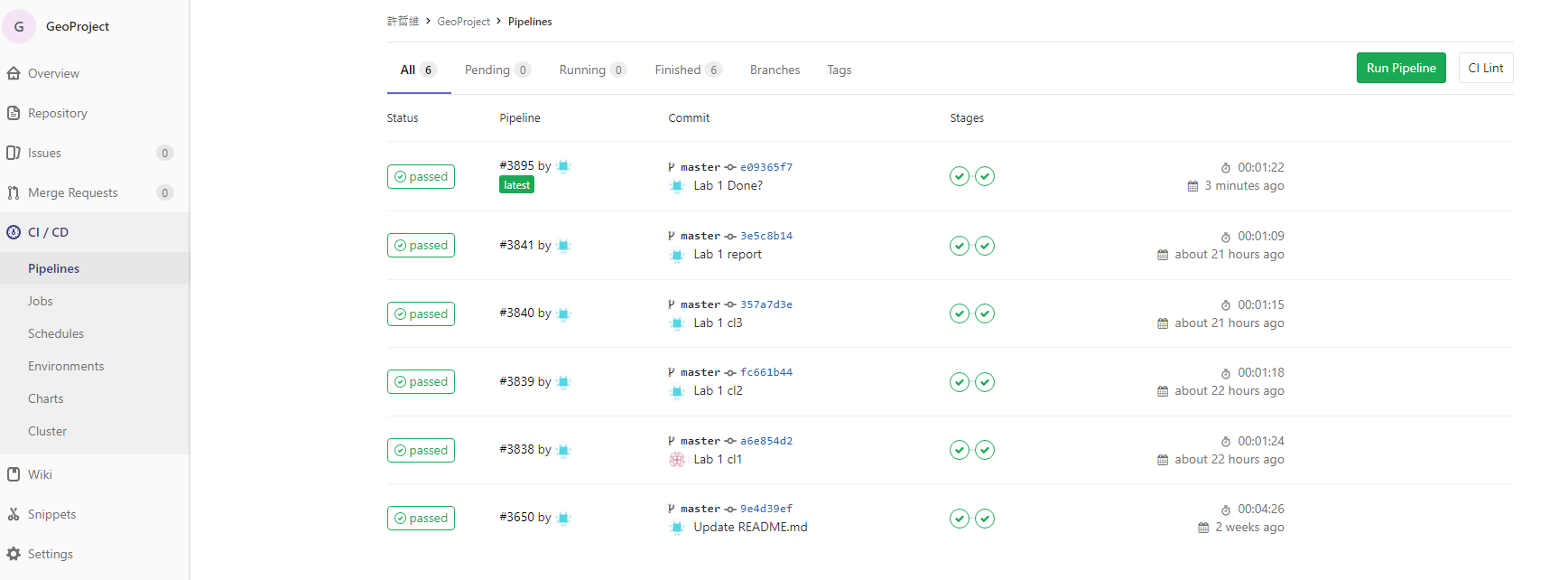
* CI#2



* CI#3

****

* CI Pipeline

****

1. **Summary**

In Lab 1, **25 test cases have been designed and implemented using JUnit**. The test is conducted in 3 CI and **the execution results of the 25 test methods are all passed**. **The total statement coverage of the test is 73%.** Thus, the test requirements described in Section 1 are satisfied.