# Lab 1 Report

Name 張家齊
Student ID 110598109
Date 2022/03/16

## 1 Test Plan

### 1.1 Test requirements

The Lab 1 requires to (1) select **15 methods** from **6 classes** of the SUT (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 cases for 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 40%*".

### 1.2 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

i. the possible **valid values** and **combinations** of the input parameters.

ii. the **boundary values** of the input parameters.

### 1.3 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 | 2022/03/11 |
| 2 | Learn JUnit | 3 | 2022/03/12 |
| 3 | Design test cases for the selected methods | 5 | 2022/03/13-15 |
| 4 | Implement test cases | 5 | 2022/03/13-15 |
| 5 | Perform test | 2 | 2022/03/15 |
| 6 | Complete Lab1 report | 2 | 2022/03/16 |

### 1.4 Success criteria

All test cases designed for the selected methods must pass (or "90% of all test cases must pass) and *the statement coverage should have achieved at least 45%*.

## 2 Test Design

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

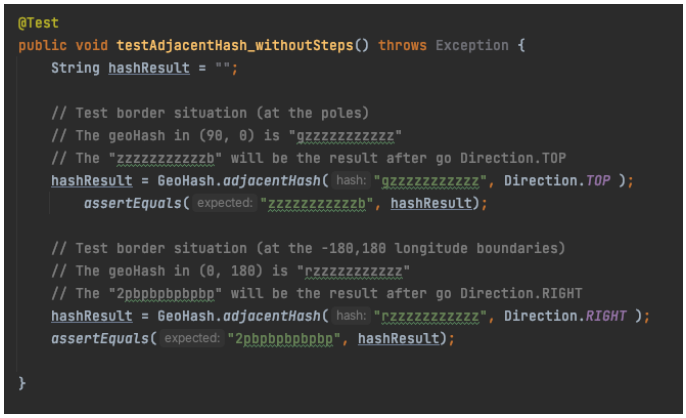
| No. | Class | Method | Test Objective | Inputs | Expected Outputs |
|---|---|---|---|---|---|
| 1 | Info | id() | Functional correctness | None | Optional<String> "this is id" |
| 2 | Info | lat() | Functional correctness | None | double 25.04360 |
| 3 | Info | lon() | Functional correctness | None | double 121.533823 |
| 4 | Info | time() | Functional correctness | None | long 30 |
| 5 | Info | value() | Functional correctness | None | String "this is the info" |
| 6 | Info | toString() | Functional correctness | None | String "Info [lat=25.043608, lon=121.533823, time=30, value=this is the info, id=Optional.of(this is id)]" |
| 7 | Base32 | encodeBase32(long i, int length) | Functional correctness | long i = 753<br>int length = 1 | String "rj" |
| | | | | long i = 753<br>int length = 12 | String "0000000000rj" |
| | | | | long i = -753<br>int length = 1 | String "-rj" |
| 8 | Base32 | encodeBase32(long i) | Functional correc | i = 753 | String "0000000000rj" |

| | | | | | |
|---|---|---|---|---|---|
| | | | tness | | |
| 9 | Base 32 | decodeBase 32() | Functional correctness | String "rj" | 753 |
| | | | | String "0000000000rj") | 753 |
| | | | | String "-rj" | -753 |
| 10 | Base 32 | getCharIndex() | Functional correctness | Char '0' | 0 |
| | | | | Char 'z' | 31 |
| | | | | Char 'a' | Exception Message "not a base32 character: a" |
| 11 | GeoHash | adjacentHash(String hash, Direction direction) | Functional correctness | String "gzzzzzzzzzzz" Direction. *TOP* | String "zzzzzzzzzzb" |
| | | | | String "rzzzzzzzzzzz", Direction. *RIGHT* | String "2pbpbpbpbpbpbp" |
| 12 | GeoHash | adjacentHash(String hash, Direction direction, int steps) | Functional correctness | String "wsqqmx41x6fs" Direction. *RIGHT* 5 | String "wsqqmx41x6gu |
| | | | | String "wsqqmx41x6fs" Direction. *RIGHT* -5 | String "wsqqmx41x6ck" |
| 13 | GeoHash | encodeHash (double latitude, double longitude) | Functional correctness | （25.043608, 121.533823） | "wsqqmx41x6fs" |
| 14 | GeoHash | neighbours( String hash) | Functional correctness | "7zzzzzzzzzzz" | "7zzzzzzzzzzx", "kpbpbpbpbpbpbp" "ebpbpbpb |

| | | | | | |
|---|---|---|---|---|---|
| | | | | | pbpb", "7zzzzzzzzzy", "ebpbpbpbpbp8", "7zzzzzzzzzw", "s00000000000", "kpbpbpbpbpbn" |
| 15 | GeoHash | decodeHash(String geohash) | Functional correctness | "wsqqmx41x6fs" | （25.043608, 121.533823） |

## 3 Test Implementation

The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 3 selected test cases are given below. The rest of test script implementations can be found in the link (or JUnit files).

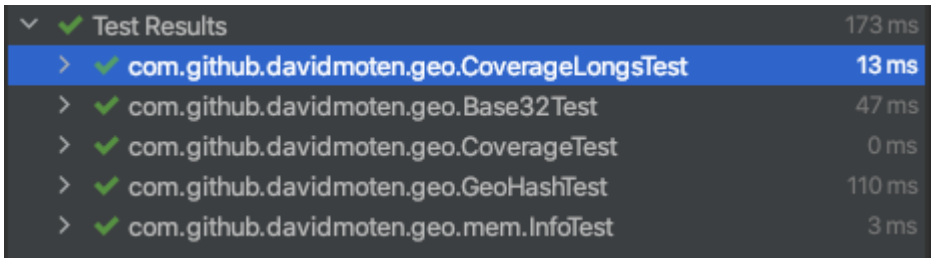
| No. | Test method | Source code |
|---|---|---|
| 1 | adjacentHash() |  |

```
@Test
public void testAdjacentHash_withoutSteps() throws Exception {
    String hashResult = "";

    // Test border situation (at the poles)
    // The geoHash in (90, 0) is "gzzzzzzzzzzz"
    // The "zzzzzzzzzzzb" will be the result after go Direction.TOP
    hashResult = GeoHash.adjacentHash( hash: "gzzzzzzzzzzz", Direction.TOP );
        assertEquals( expected: "zzzzzzzzzzzb", hashResult);

    // Test border situation (at the -180,180 longitude boundaries)
    // The geoHash in (0, 180) is "rzzzzzzzzzzz"
    // The "2pbpbpbpbpbp" will be the result after go Direction.RIGHT
    hashResult = GeoHash.adjacentHash( hash: "rzzzzzzzzzzz", Direction.RIGHT );
    assertEquals( expected: "2pbpbpbpbpbp", hashResult);

}
```

| | | |
|---|---|---|
| 2 | encodeBase32() | `@Test`<br>`public void encodeBase32_withParameterLength() throws Exception {`<br>`    // the num 753 is "rj" in encodeBase32`<br>`        int length = 0;`<br>`        long i = 753;`<br>`    String encode = "";`<br><br>`    // test length less than actually need`<br>`    length = 1;`<br>`    encode = Base32.encodeBase32( i, length );`<br>`    assertEquals( expected: "rj", encode);`<br><br>`    // test length more than actually need`<br>`    length = 12;`<br>`    encode = Base32.encodeBase32( i, length );`<br>`    assertEquals( expected: "0000000000rj", encode);`<br><br>`    // test given negative i`<br>`    length = 1;`<br>`    i = -i;`<br>`    encode = Base32.encodeBase32( i, length );`<br>`    assertEquals( expected: "-rj", encode);`<br>`}` |
| 3 | getCharIndex() | `@Test`<br>`public void getCharIndex() throws Exception {`<br>`    // get the index inside characters[](inside Base32.java)`<br><br>`    char ch = '0'; // test first index`<br>`    assertEquals( expected: 0, Base32.getCharIndex(ch) );`<br><br>`    ch = 'z'; // test last index`<br>`    assertEquals( expected: 31, Base32.getCharIndex(ch) );`<br><br>`    ch = 'a'; // test char not in list`<br>`    String exceptionMessage = "not a base32 character: " + ch;`<br>`    try {`<br>`        Base32.getCharIndex(ch);`<br>`    } catch ( IllegalArgumentException throwMessage ) {`<br>`        assertEquals(  exceptionMessage , throwMessage.getMessage());`<br>`    }`<br>`}` |

# 4 Test Results

## 4.1 JUnit test result snapshot



## Test Summary

| 24 tests | 0 failures | 0 ignored | 0.173s duration |
|---|---|---|---|

100% successful

| Package | Tests | Failures | Ignored | Duration | Success rate |
|---|---|---|---|---|---|
| com.github.davidmoten.geo | 18 | 0 | 0 | 0.170s | 100% |
| com.github.davidmoten.geo.mem | 6 | 0 | 0 | 0.003s | 100% |

### 4.2 Code coverage snapshot

- Coverage of each selected method


- Total coverage

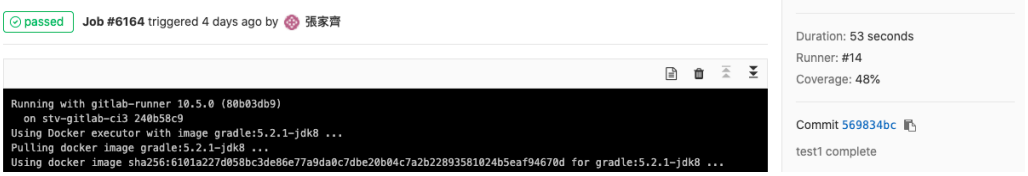
**geo**

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods | Missed | Classes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| com.github.davidmoten.geo | | 61% | | 50% | 82 | 149 | 149 | 348 | 35 | 68 | 4 | 10 |
| com.github.davidmoten.geo.mem | | 19% | | 0% | 23 | 30 | 48 | 61 | 13 | 20 | 2 | 3 |
| com.github.davidmoten.geo.util | | 36% | | 50% | 2 | 4 | 2 | 6 | 0 | 2 | 0 | 1 |
| Total | 1,047 of 2,326 | 54% | 103 of 186 | 44% | 107 | 183 | 199 | 415 | 48 | 90 | 6 | 14 |

### 4.3 CI result snapshot (3 iterations for CI)

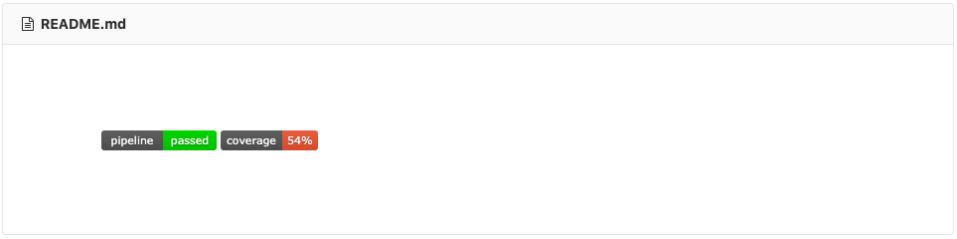
這部分因為跑 CI 時並未每次截圖，因此改以擷取 **Pipeline Jobs** 部分之 **coverage** 紀錄

- CI#1

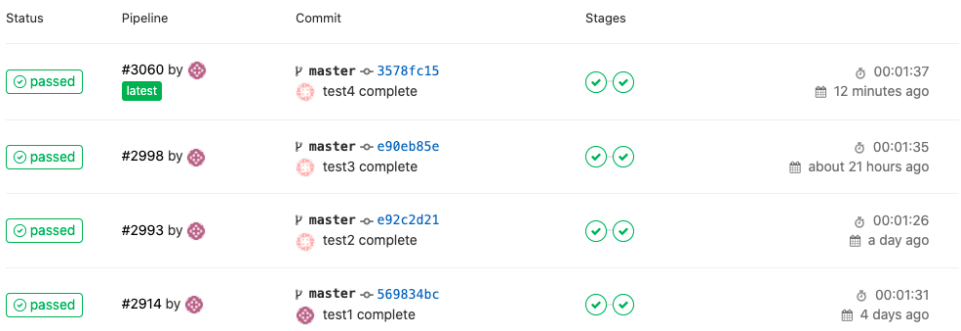



- CI#2


- CI#3




- CI Pipeline




## 5 Summary

In Lab 1, 15 **test cases have been designed and implemented using JUnit**. The test is conducted in 4 CI and **the execution results of the 15 test methods are all passed**. **The total statement coverage of the test is 54%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are :

1.The skills of unit test, decomposition code into different part.

2.How to analysis code state and make the coverage higher than naive test.

## 6 Reference

https://en.wikipedia.org/wiki/Geohash

https://davidmoten.github.io/geo/apidocs/index.html