**Lab 1 Report**

Name:呂聖文

Student ID:108598007

Date:2020/3/19

1. **Test Plan**
   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 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 80%*”.

* 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 | 2 | 3/11 |
| 2 | Learn JUnit | 1 | 3/11 |
| 3 | Design test cases for the selected methods | 1 | 3/11 |
| 4 | Implement test cases | 5 days | 3/12~3/16 |
| 5 | Read Info class |
| 6 | Read Geohash class |
| 7 | Read Base32 |
| 8 | Perform test |
| *9* | Complete Lab1 report | 3 days | 3/17~3/19 |

* 1. **Success criteria**

All test cases designed for the selected methods must pass and *the statement coverage should have achieved at least 80%.*

1. **Test Design**

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

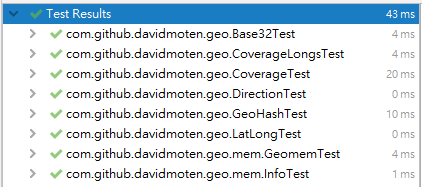
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Class** | **Method** | **Test Objective** | **Inputs** | **Expected Outputs** |
|  | Base32 | encodeBase32() | 編碼是否正確 | (75324,4) | “29jw” |
|  | Base32 | encodeBase32() | 負數是否正確 | (-75324,4) | “-29jw” |
|  | Base32 | encodeBase32() | 測試無length參數是否為12位數 | (75324) | "0000000029jw" |
|  | Base32 | decodeBase32() | 解碼是否正確 | (“29jw”) | 75324 |
|  | Base32 | decodeBase32() | 負數解碼是否正確 | (“-29jw”) | -75324 |
|  | Base32 | getCharIndex() | 搜尋’j’為陣列第幾index | ('j') | 17 |
|  | Base32 | getCharIndex() | 搜尋非這列中字元是否正常拋出例外 | (‘I’) | IllegalArgumentException |
|  | Base32 | padLeftWithZerosToLength() | 測試補0功能 | ("jw",6) | "0000jw" |
|  | Coverage | Coverage() | 使用CoverageLongs物件套入Coverage中使用是否正常 | **long**[] hashes=**new long**[]{40,36,34}; **int** count=3; **double** ratio=3.14; | getRatio()==3.14  getHashLength()==2 |
|  | Coverage | getHashes() | 參數陣列與回傳陣列是否為同一記憶體位置 | Coverage(**hashes**,3.14); | getHashes== **hashes**(object) |
|  | Coverage | getRatio() | 傳入參數與回傳參數是否為同一個數字 | 3.14 | getRatio() ==3.14 |
|  | Coverage | getHashLength() | 只取陣列中第一個元素的前四個位元 測試10001 AND 000001111 結果為1 | 17 | 1 |
|  | Coverage | getHashLength() | 測試陣列沒有元素應回傳0字元 | hashes={} | getHashLength()==0 |
|  | CoverageLongs | getHashes() | 測試CoverageLongs的getHashes 比對參數陣列與回傳陣列內容是否相同 | hashes={40,36,34} | getHashes()==hashes |
|  | CoverageLongs | getRatio() | 測試參數ratio與回傳ratio是否同一數字 | 3.14 | getRatio()==3.14 |
|  | CoverageLongs | getCount() | 測試hashes陣列的元素共有幾個 | hashes={17,36,34} | getCount()==3 |
|  | GeoHash | adjacentHash() | km區域上面是kq | ("km",Direction.TOP) | kq |
|  | GeoHash | adjacentHash() | w往左五格是9 | ("w",Direction.LEFT,5) | 9 |
|  | GeoHash | decodeHash() | 將geohash “0000”轉為經緯度 | ("0000") | getLat()==  -89.91210938  getLon()==  -179.82421875 |
|  | GeoHash | encodeHash() | 將經緯度-38.23242188, -149.58984375 轉換為geohash | (-38.23242188,  -149.58984375,4) | “26jw” |
|  | GeoHash | Left() | 測試w區域左邊(西邊)區域 | (**"w"**) | “t” |
|  | GeoHash | Right() | 測試w區域左邊(東邊)區域 | (**"w"**) | “x” |
|  | GeoHash | Top() | 測試w區域左邊(北邊)區域 | (**"w"**) | “y” |
|  | GeoHash | Bottom() | 測試w區域左邊(南邊)區域 | (**"w"**) | “q” |
|  | GeoHash | neighbours() | 測試w區域八方位 | (**"w"**) | geohash.get(0)==”t”  geohash.get(1)==”x”  geohash.get(2)==”y”  geohash.get(3)==”q”  geohash.get(4)==”v”  geohash.get(5)==”m”  geohash.get(6)==”z”  geohash.get(7)==”r” |
|  | GeoHash | hashContains() | 測試geohash wz是否再此經緯度參數區域內 | (**"wz"**,42.18750000, 129.37500000) | isContain==true |
|  | GeoHash | fromLongToString() | 測試當hash小於1將拋出例外 | -1 | Catch IllegalArgumentException |
|  | Info | id() | 測試回傳id是否與optional為同一記憶體位置 | **optional** = Optional.*of*(**"NTUT"**); | Optional==info.id() |
|  | Info | lat() | 測試回傳lat是否為當初設定的3.123 | **info**=**new** Info(3.123,4.123,10, 5, **optional**); | **info**.lat()==3.123 |
|  | Info | lon() | 測試回傳lon是否為當初設定的4.123 | **info**=**new** Info(3.123,4.123,10, 5, **optional**); | **info**.lat()==4.123 |
|  | Info | time() | 測試回傳time是否為當初設定的10 | **info**=**new** Info(3.123,4.123,10, 5, **optional**); | **info**.time()==10 |
|  | Info | value() | 測試回傳value是否為當初設定的5 | **info**=**new** Info(3.123,4.123,10, 5, **optional**); | **info**.value()==5 |

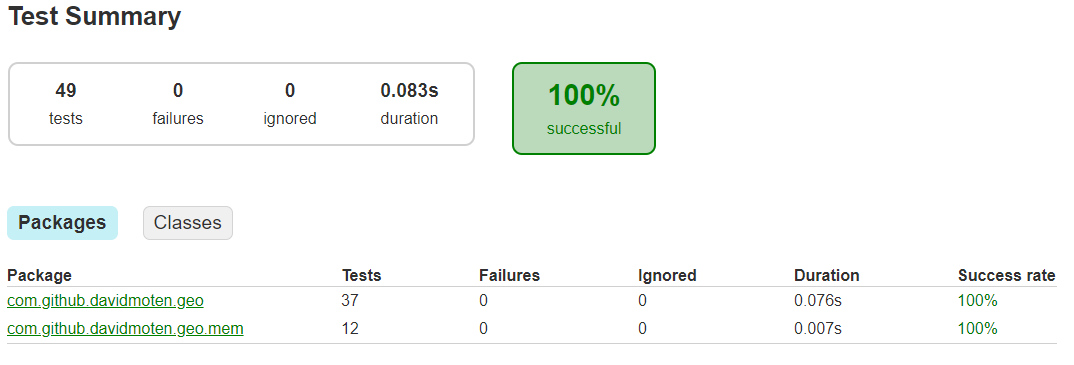
1. **Test Implementation**

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](https://github.com) (or JUnit files).

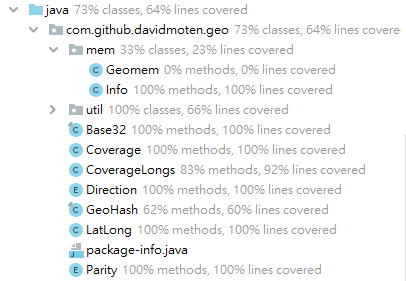
|  |  |  |
| --- | --- | --- |
| **No.** | **Test method** | **Source code** |
|  | encodeBase32() | <https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java> |
|  | encodeBase32\_negative() |
|  | encodeBase32\_noLength() |
|  | decodeBase32() |
|  | decodeBase32\_negative() |
|  | getCharIndex() |
|  | getCharIndex\_exception() |
|  | getCharIndex\_padLeftWithZerosToLength() |
|  | Coverage() | <https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/CoverageTest.java> |
|  | getHashes() |
|  | getRatio() |
|  | getHashLength() |
|  | getHashLength\_sizeZero() |
|  | testToString() |
|  | getHashes() | <https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/CoverageLongsTest.java> |
|  | getRatio() |
|  | getHashLength () |
|  | getHashLength\_sizeZero() |
|  | getCount() |
|  | adjacentHash() | <https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java> |
|  | adjacentHash\_steps() |
|  | decodeHash() |
|  | encodeHash\_lat\_long() |
|  | left() |
|  | right() |
|  | top() |
|  | bottom() |
|  | neighbours() |
|  | hashContains() |
|  | fromLongToString() |
|  | hashLengthToCoverBoundingBox() |
|  | coverBoundingBoxLongs() |
|  | id() | <https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java> |
|  | lat() |
|  | lon() |
|  | time() |
|  | value() |
|  | testToString() |

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

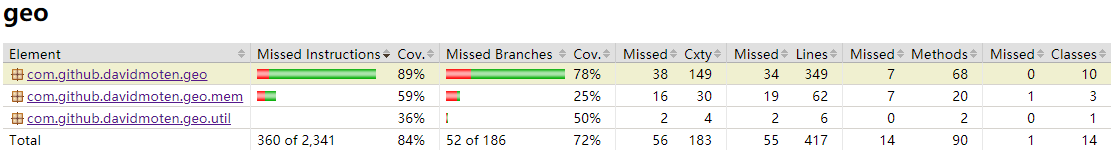
****

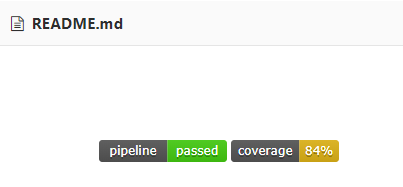
****

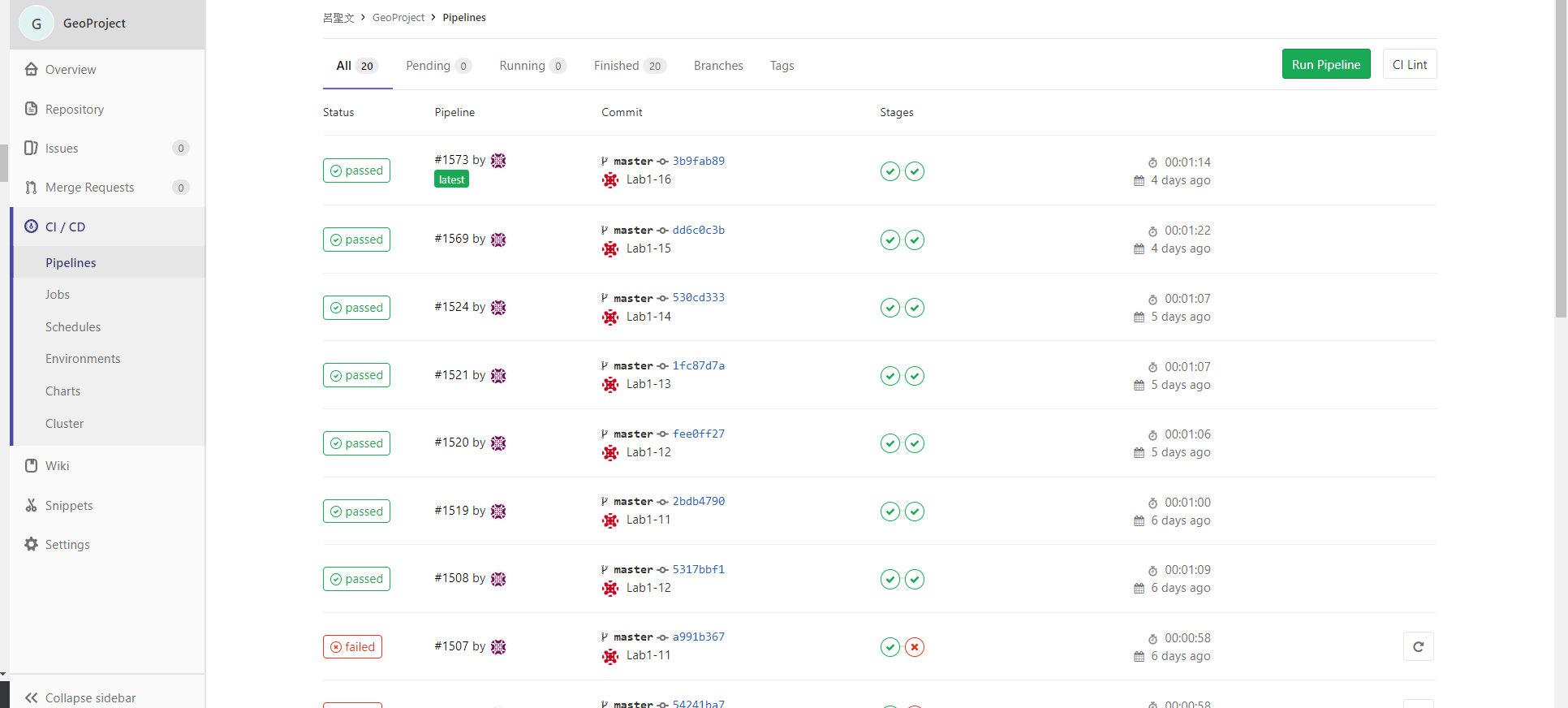
* 1. **Code coverage snapshot**
* Coverage of each selected method

****

* Total coverage

****

* 1. **CI result snapshot (3 iterations for CI)**
* CI#1****
* CI#2****
* CI#3****
* CI#4****
* CI#5****
* CI#6****
* CI#7****
* CI#8
* ****
* CI#9
* 
* CI Pipeline

****

1. **Summary**

In Lab 1, **41 test cases have been designed and implemented using JUnit**. The test is conducted in 9 CI and **the execution results of the 15 test methods are all passed**. **The total statement coverage of the test is 84%.** Thus, the test requirements described in Section 1 are satisfied.

過去只有POSD有寫過測試經驗，但那時沒有所謂的覆蓋率問題，本次作業是我第一次針對覆蓋率進行撰寫測試，以前我們測的是自己寫的程式，因此馬上就可開始撰寫測試，知道自己要測些甚麼，而這次卻不一樣，我們要測試別人寫的程式，導致需要花許多時間去讀懂程式碼才能知道如何撰寫測試，看懂別人的的程式碼也是需要許多精力去完成，且一次又一次覆蓋率增加，心裡也有一種成就感，最終達到自己的覆蓋率目標80%!

以前我一直疑惑軟體測試到底在做些甚麼，上學期POSD也僅測試一點點而已，這學期修了軟體測試這門課程，開始慢慢了解到原來軟體測試分為很多層面，經過第一次作業也慢慢能夠進入狀況，知道該怎麼做會比較好一 些。