**Lab 2 Report**

Name 吳承軒

Student ID 106598068

Date 2018/04/09

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

The Lab 2 requires to (1) select 15 methods from 6 classes of the SUT (GeoProject), (2) design Unit test cases by using **input space partitioning (ISP)** technique for the selected methods, (3) develop test scripts to implement the test cases, (4) execute the test script on the selected methods, (5) report the test result, and (6) learn and gain the experience of designing test cases systematically with ISP technique.

In particular, based on the statement coverage criterion, the **test requirements** for Lab 1 are to design test cases *with ISP* 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 80% (greater than that of Labe 1)*”.

* 1. **Test Strategy**

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

1. select those 10 methods that were chosen in Lab1 and 5 new methods that were NOT selected previously. If possible, some of the methods do NOT have primitive types of input or output parameters (if possible).
2. set the objective of the minimum statement coverage to be greater than that of Lab 1 and adjust the objective based on the time available (if necessary).
3. design the test cases for those selected methods by using the **input space partitioning (ISP)** technique.
   1. **Test activities**

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

|  |  |  |  |
| --- | --- | --- | --- |
| **No.** | **Activity Name** | **Plan hours** | **Schedule Date** |
| 1 | Study GeoProject | 3hr | 3/28 |
| 2 | Learn JUnit | 1hr | 3/29 |
| 3 | Design test cases for the selected methods | 3hr | 4/4 |
| 4 | Implement test cases | 2hr | 4/7 |
| 5 | Perform test | 2hr | 4/8 |
|  | Complete Lab2 report | 2hr | 4/9 |

* 1. **Design Approach**

The **ISP** technique will be used to design the test cases. Specifically, the possible partitions and boundary values of each input parameter shall be identified first using the **Mine Map** and **domain knowledge** (if applicable). The valid combinations of the partitions as well as the boundary values shall be computed for the input parameters of each selected method. Each of the partition combination can be a possible test case. Add more test cases by considering the possible values and boundary of the outputs for the methods or by using test experiences.

* 1. **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 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** |
| 1 | Base32 | encodeBase32 | Test encoding | long, int | string |
| 2 | Base32 | encodeBase32 | Test encoding | long | string |
| 3 | Base32 | decodeBase32 | Test decoding | string | long |
| 4 | Base32 | padLeftWithZerosToLength | Test padLeftWithZerosToLength | string | string |
| 5 | Base32 | getCharIndex | Test getCharIndex | Char | int |
| 6 | CoverageLongs | getRatio | Test getRatio | hash, int, double | double |
| 7 | CoverageLongs | getHashes | Test getHashes | long, int, double | boolean |
| 8 | Coverage | getRatio | Test getRatio | set, double | double |
| 9 | GeoHash | adjacentHash | Test  adjacentHash | string, int, Directon | string |
| 10 | GeoHash | encodeHash | Test  encodeHash | double, double, int | string |
| 11 | GeoHash | right | Test  Right | string | string |
| 12 | GeoHash | top | Test top | string | string |
| 13 | GeoHash | encodeHash | Test  encodeHash | LatLong | String |
| 14 | GeoHash | fromLongToString | Test  fromLongToString | long | String |
| 15 | LatLong | Add | Test add | double, double | string |

The details of the design is below:

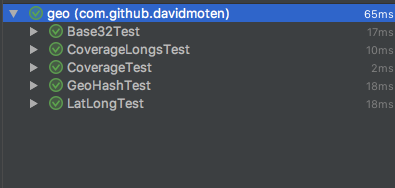
The Excel file of test cases…

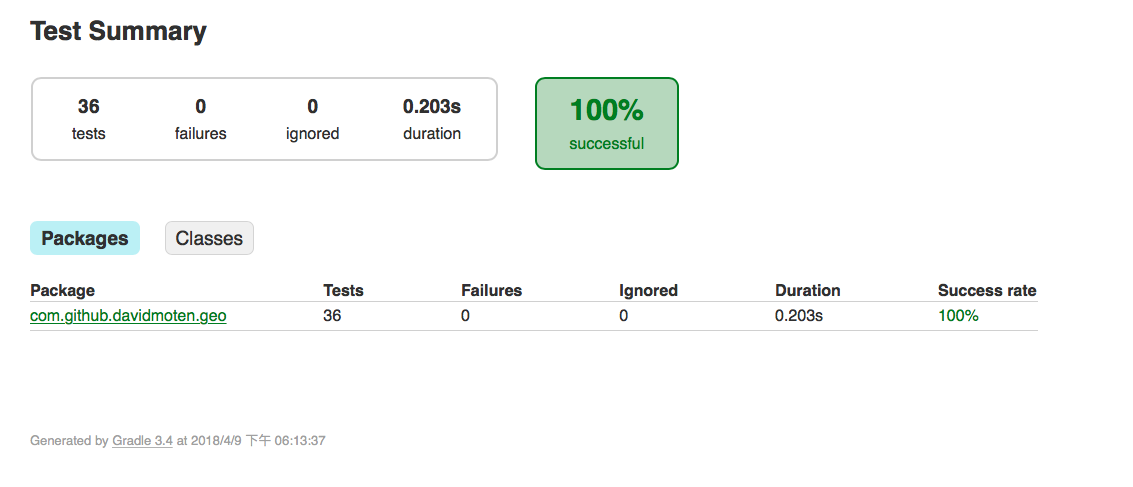
1. **Test Implementation**

The design of test cases specified in Section 2 was implemented using JUnit 4. The test script of 3 selected test cases are given below. The rest of test script implementation can be found in the [link](https://github.com).

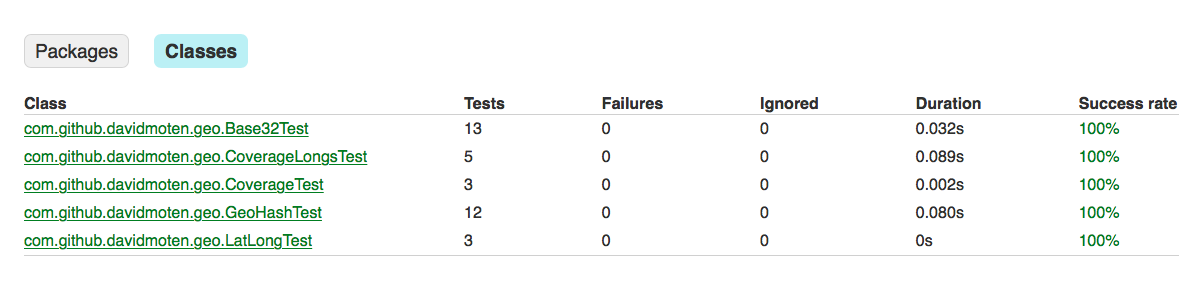
|  |  |  |
| --- | --- | --- |
| **No.** | **Test method** | **Source code** |
| 1 | encodePositiveBase32 () | @Test public void encodePositiveBase32() throws Exception {  String encode = Base32.*encodeBase32*(75324, 4);  *assertEquals*("29jw", encode); } |
| 2 | encodeZeroBase32() | @Test public void encodeZeroBase32() throws Exception {  String encode = Base32.*encodeBase32*(0, 4);  *assertEquals*("0000", encode); } |
| 3 | encodeNegativeBase32() | @Test public void encodeNegativeBase32() throws Exception {  String encode = Base32.*encodeBase32*(-1, 4);  *assertEquals*("-0001", encode); } |

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

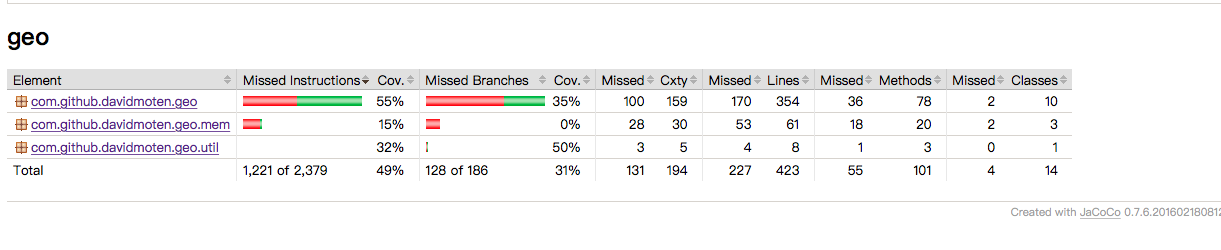
****

****

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

****

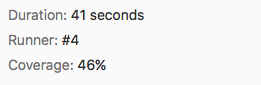
* Total coverage

****

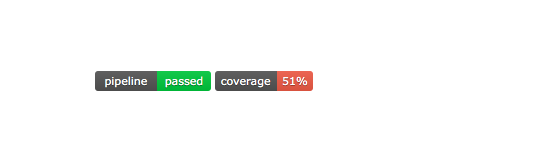
* 1. **CI result snapshot (3 iterations for CI)**
* CI#1

****

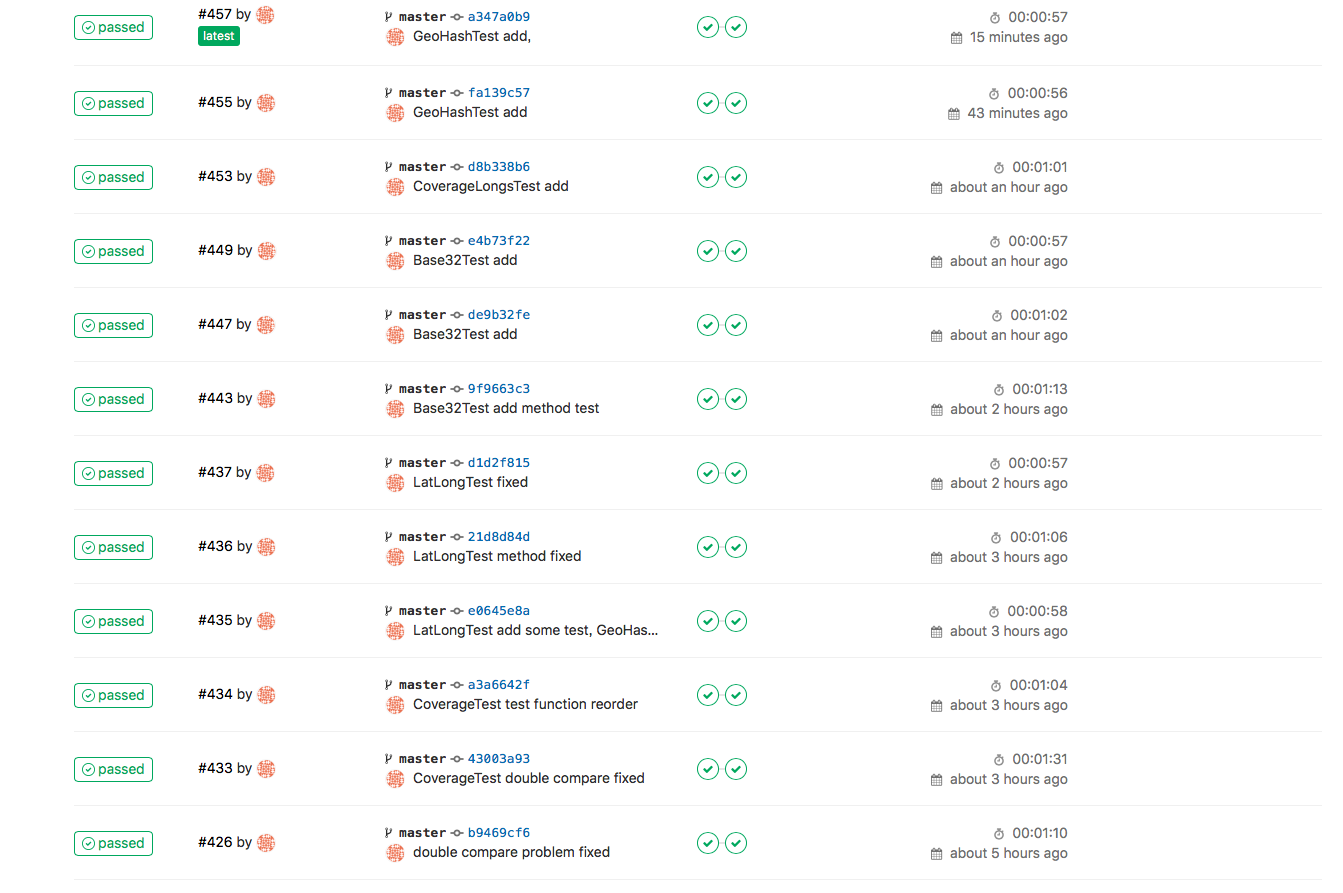
* CI#2



* CI#3

****

* CI Pipeline

****

1. **Summary**

In Lab 1, **15 test cases have been designed and implemented using JUnit and ISP technique**. The test is conducted in 3 CI and **the execution results of the 15 test methods are all passed**. **The total statement coverage of the test is 80%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are …

在這次的Lab裡學習到了多種測試的方法，使用一種以更high-level的方式來看待整體的Test，從而撰寫不同規模大小的測試，來符合時間，成本等因素的考量，藉此使我了解在不同資源下，所需因應的各種不同方法，才能了解未來在職場上面對不同專案，以及所分配到的資源所能採取的對應方法。