Lab 2 Report

吳俊青
108598014
2020/04/20

1 Test Plan

1.1 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 scripts on the selected methods, (5) report the test results, and (6) specify your experiences of designing test cases systematically using the ISP technique.

In particular, based on the statement coverage criterion, the **test requirements** for Lab 2 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 70% (greater than Lab 1)*".

1.2 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 are 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 test 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.

1.3 Test activities

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

No.	Activity Name	Plan hours	Schedule Date
1	Study GeoProject	1	2020/04/10
2	Learn **ISP** and JUnit	2	2020/04/11~2020/04/12
3	Design test cases for the selected methods	7	2020/04/13~2020/04/15
4	Implement test cases	6	2020/04/17~2020/04/19

5	Perform tests	1	2020/04/17~2020/04/19
6	Refactor test cases	1	2020/04/17~2020/04/19
7	Complete Lab2 report	3	2020/04/20

1.4 Design Approach

The **ISP** technique will be used to design the test cases. Specifically, the possible partitions and boundary values of input parameters shall be identified first using the **Mine Map** and **domain knowledge** (if applicable). The possible **valid** combinations of the partitions (i.e., **all combination coverage**) 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.*

1.5 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 70%*.

2 Test Design

To fulfill the test requirements listed in section 1.1, the following methods are selected and corresponding test cases are designed.紅色 method 為新增的 method，並且因版面關係，每一個 method 只列出第一個 test case 的資訊。

No.	Class	Method	Test Objective	Inputs	Expected Outputs
1	Base32	encodeBase32 (long i, int length)	Boundary value 能夠產生正確 output	i=9223372036854775807L length = 14	"07zzzzzzzzzzz"
2	Base32	encodeBase32 (long i)	Boundary value 能夠產生正確 output	i=9223372036854775807L	"7zzzzzzzzzzz"
3	Base32	decodeBase32(String hash)	Boundary value 能夠產生正確 output，且輸入例外資料，能夠產生預期 throw	hash="-8000000000000"	-9223372036854775808L
4	GeoHash	adjacentHash (String hash, Direction direction)	能正常產生 hash 的不同方向對應字串，且不合格的 input hash 能產生出預期例外	hash = "STVV" direction= Direction.BOTTOM	"sttu"
5	GeoHash	adjacentHash(String hash, Direction direction, int steps)	能正常產生 hash 的不同方向對應的 steps 個字串，且不合格的 input hash 能產生出預期例外	hash = "STVV" direction = Direction.BOTTOM steps = 5	"stvb"

6	GeoHash	Neighbours (String hash)	能正常產生hash的八個方便的對應字串，並且針對部分字串能夠產生預期throw	hash = “-6”	[“-1”, “-3”, “-4”, “-d”, “-5”, “-e”, “-7”, “-9”]
7	GeoHash	hashLengthToCoverBoundingBox (double topLeftLat, double topLeftLon,double bottomRightLat, double bottomRightLon)	能正常產生 input 邊界對應的字串集合為長度多長的 hash 組成的結果	topLeftLat = 1 topLeftLon = 1 bottomRightLat = 1 bottomRightLon = 1	12
8	GeoHash	widthDegrees (int n)	能夠產生長度 n 的 hash 的 widthDegrees	n = 13	4.190951585769653E-8 delta=0.001
9	GeoHash	heightDegrees (int n)	能夠產生長度 n 的 hash 的 heightDegrees	n = 13	4.190951585769653E-8 delta=0.001
10	GeoHash	coverBoundingBoxLongs (double topLeftLat, final double topLeftLon, final double bottomRightLat, final double bottomRightLon, final int length)	能正常產生數量 length 的 input 邊界對應的字串，並且對不合格 input 確實產生 throw	topLeftLat = 1 topLeftLon = 1 bottomRightLat = 1 bottomRightLon = 1	CoverageLongs object
11	GeoHash	coverBoundingBoxMaxHashes (double topLeftLat, final double topLeftLon,final double bottomRightLat, final double bottomRightLon, int maxHashes)	能正常產生 input 邊界對應且數量 maxHashes 的 hash 字串集合，並且對不合格的 input 確實產生 throw	topLeftLat = 1 topLeftLon = 1 bottomRightLat = 1 bottomRightLon = 1 maxHashes = 2147483647	Coverage object
12	GeoHash	right(String hash)	能正常產生hash的對應方向的hash字串	hash="22"	“28”
13	GeoHash	left(String hash)	能正常產生hash的對應方向的hash字串	hash="22"	“20”
14	GeoHash	top(String hash)	能正常產生hash的對應方向的hash字串	hash="ab"	“ac”
15	GeoHash	bottom (String hash)	能正常產生hash的對應方向的hash字串	hash="10"	“11”

The details of the design are given below:

The Excel file of test case: 108598014 Lab2 (ISP test case design).xlsx

此檔與 Lab2Report 在同一資料夾中

3 Test Implementation

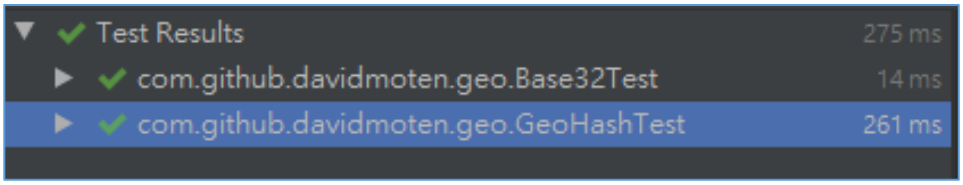
The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 3 selected test cases are given below. The rest of the test script implementations can be found in the below link.

https://stv.csie.ntut.edu.tw/108598014/GeoProject

No.	Test method	Source code
1	encodeBase32WithPositiveNumAndBiggerLength()	Base32Test.java
2	hashLengthToCoverBoundingBoxWithSixteenBoundary()	GeoHashTest.java
3	`testAdjacentWithDifferentDirectionAndSingularHash()`	GeoHashTest.java

4 Test Results

4.1 JUnit test result snapshot



Test Summary

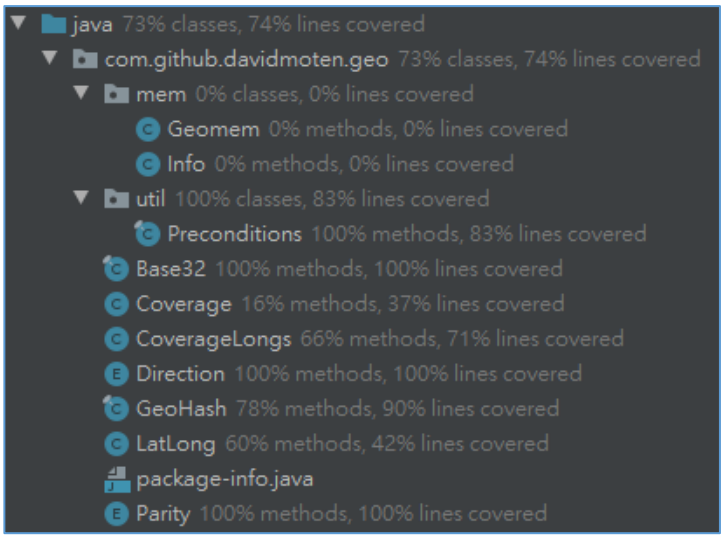
46 tests	0 failures	0 ignored	0.402s duration	100% successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	46	0	0	0.402s	100%

4.2 Code coverage snapshot

- Coverage of each selected method



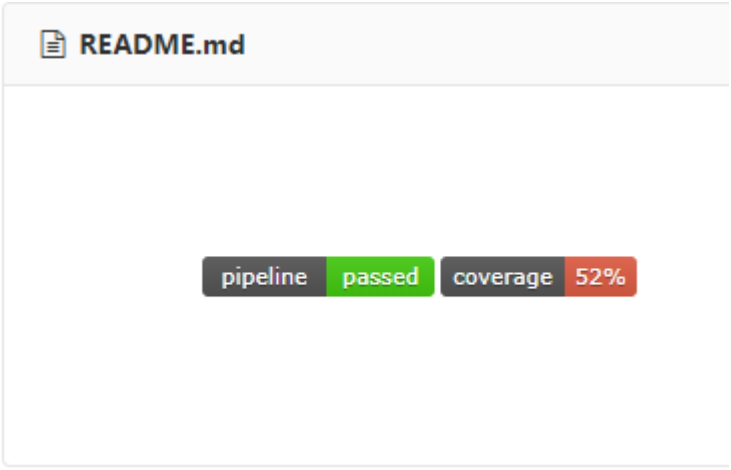
- Total coverage

geo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo.mem		0%		0%	30	30	61	61	20	20	3	3
com.github.davidmoten.geo		85%		86%	33	149	47	348	18	68	0	10
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	627 of 2,326	73%	43 of 186	76%	64	183	109	415	38	90	3	14

4.3 CI result snapshot (3 iterations for CI)

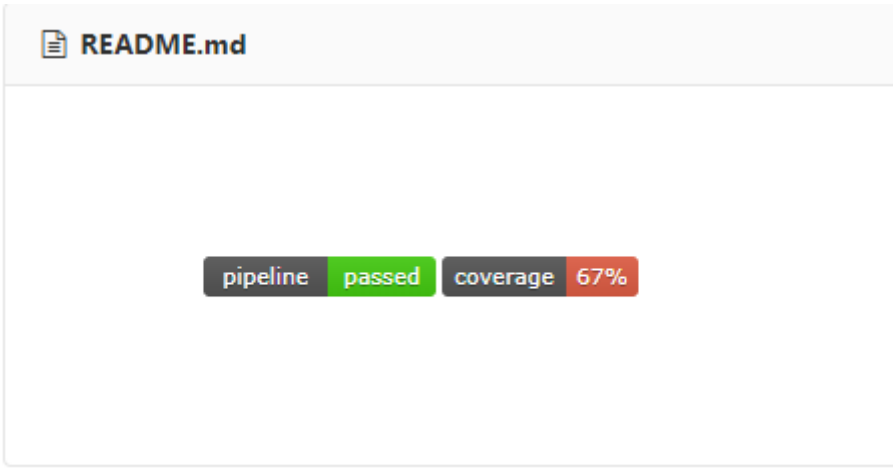
- CI#1



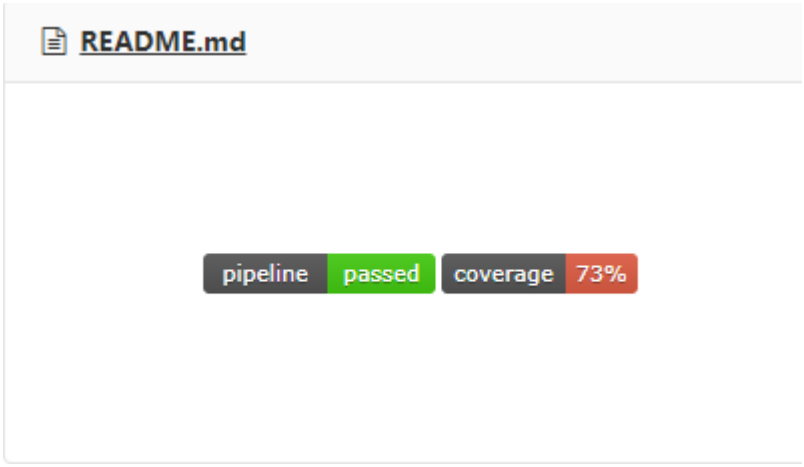
- CI#2


README.md
pipeline
passed
coverage
56%

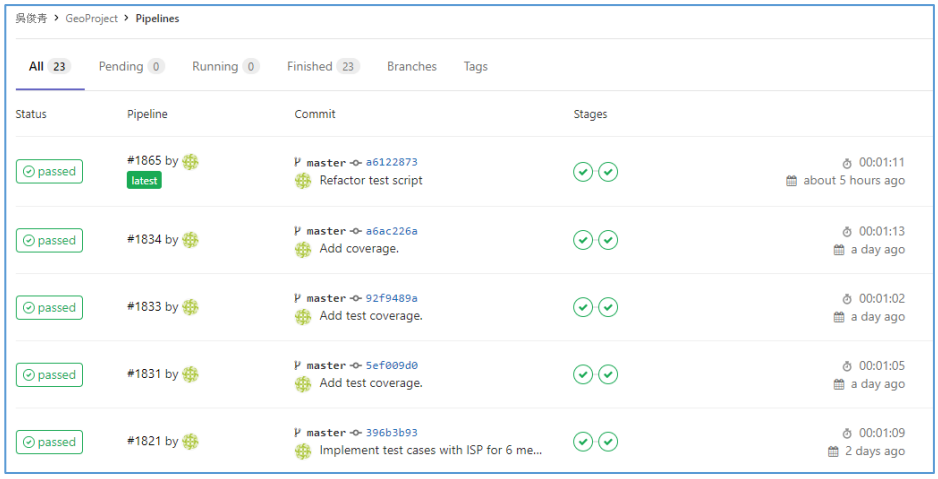
- CI#3


README.md
pipeline
passed
coverage
67%

- CI#4


README.md
pipeline
passed
coverage
73%

- CI Pipeline



5 Summary

In Lab 2, **46 test cases have been designed and implemented using JUnit and the ISP technique**. 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 73%.** Thus, the test requirements described in Section 1 are satisfied.

在 Lab2 中我從 Lab1 中挑選出了 10 個 method 以及新增 5 個 method，並且利用 ISP 設計每一個 method 所屬的 test case，在這次作業中我已經較會運用新的 IDE 開發工具及 Junit，並且我也發現我花了較多時間在事前的設計，將所認為有可能的特性一一列出，並且思考該如何才可以達到最大的 statement coverage，根據以上的原則，使我循序漸進地完成了 Lab2，最後測試覆蓋率也達到了 73%，高於原先 Lab1 的 56%，後續會繼續研究程式碼來補足我測試未覆蓋到的部分。