Lab 3 Report

Name:呂聖文
Student ID:108598007
Date: 5/19

1 Test Plan

1.1 Test requirements

The Lab 3 requires to (1) select 6 methods from 6 classes of the SUT (GeoProject), (2) design Unit test cases by using **basis path or graph coverage** 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 graph coverage technique.

In particular, based on the target coverage criteria (i.e., statement, branch, or others), the **test requirements** for Lab 3 are to design test cases *with* **graph coverage technique** for each selected method so that "*each statement and branch (or path) of the method under test will be covered by at least one test case* and *the both minimum* ***statement*** *(node) and* ***branch*** *(edge) coverage are greater than those of Lab 2 and 90%, respectively*."

1.2 Test Strategy

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

(1) select **3 methods that were chosen in Lab1 or Lab2** and **3 new methods** that are NOT selected previously. The selected methods MUST contain **predicate** and/or **loop** structures (as many as possible).

(2) set the objective of the minimum statement or branch (or path) coverage to be greater than that of Lab 2 and adjust the test objective (e.g., 90%, 95% or 100%) based on the time available (if necessary).

(3) design the test cases for those selected methods by using the **basis path or graph coverage** testing 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 hour	5/13
2	Learn **basis path and**	3 hour	5/13

	graph coverage		
3	Design test cases for the selected methods	1 days	5/14
4	Implement test cases	3 days	5/15-5/17
5	Perform tests and check code coverage. If not satisfy, design more test cases…	1 days	5/18
6	Complete Lab3 report	3 hour	5/19

1.4 Design Approach

The **basis path and graph coverage** technique will be used to design the test cases. Specifically, the control flow graph (CFG) of each selected method shall be drawn first, and the possible test paths that satisfy the test requirements (i.e., **statement (node), branch (edge), or path coverage**) shall be derived from the CFG. The possible **inputs** and **expected outputs** for the derived test paths shall be computed from the specification of SUT for each method under test. *Add more test cases by considering to satisfy other coverage criteria, such as edge-pair, all-use, or prime-path coverage criteria.*

1.5 Success criteria

All test cases designed for the selected methods must pass (or 90% of all test cases must pass) and *both statement and branch (or path) coverage should have achieved at least 90%, respectively*.

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	Source Code Links	CFG Links	Test Paths	Inputs	Expected Outputs
1	GeoHash	`gridAsString()`	https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java	Figure/ Figure1.png	{n1,n2,n3,n4, n5,n7,n8,n9, n4,n10,n11,n 2,n12}	hash:"dr" fromRight:0 fromBottom:1 toRight:1 toBottom:1 set=["dq","te"]	"DQ dw \n"
2	GeoHash	adjacentHash()		Figure/ Figure2.png	P1:{n1,n2}	hash:"" direction:RIGHT	IllegalArgumentException
3	GeoHash	adjacentHash()			P2:{n1,n3,n4}	hash:"z", direction:RIGHT	"b"

4	GeoHash	adjacentHash()	https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java	Figure/ Figure2.png	P3:{n1,n3,n5, n6,n8,n9,n10 }	hash:"drdz" direction:RIGHT	"drep"
5	GeoHash	adjacentHash()			P4:{n1,n3,n5, n6,n8,n10}	hash:"dr" direction:RIGHT	"dx"
6	GeoHash	adjacentHash()			P5:{n1,n3,n5, n7,n8,n9,n10 }	hash:"ssz",direct ion:RIGHT	"sub"
7	GeoHash	adjacentHash()			P6:{n1,n3,n5, n7,n8,n10}	hash:"sss",direct ion:RIGHT	"sst"
8	GeoHash	widthDegrees( )		Figure/ Figure3.png	P1:{n1,n2}	n:1	45.0
9	GeoHash	widthDegrees( )			P2:{n1,n3}	n:13	4.1909515 85769653 E-8
10	Base32	decodeBase32 ()	https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java	Figure/ Figure4.png	P1:{n1,n2,n4, n5,n6,n7}	hash:"-a" direction:RIGHT	IllegalArgu mentExce ption
11	Base32	decodeBase32 ()			P2:{n1,n2,n4, n5,n6,n8,n5, n9,n10,n11}	hash:"-b" direction:RIGHT	-10
12	Base32	decodeBase32 ()			P3:{n1,n2,n4, n5,n6,n8,n5, n9,n11}	hash:"-c" direction:RIGHT	-11
13	Base32	decodeBase32 ()			P4:{n1,n3,n4, n5,n6,n7}	hash:"a" direction:RIGHT	IllegalArgu mentExce ption
14	Base32	decodeBase32 ()			P5:{n1,n3,n4, n5,n6,n8,n5, n9,n10,n11}	hash:"b" direction:RIGHT	10
15	Base32	decodeBase32 ()			P6:{n1,n3,n4, n5,n6,n8,n5, n9,n11}	hash:"c" direction:RIGHT	11
16	Base32	encodeBase32 ()		Figure/ Figure5.png	P1:{n1,n2,n3, n4,n5,n4,n6, n7}	i:31 length:4	"000z"
17	Base32	encodeBase32 ()			P2:{n1,n2,n3, n4,n5,n4,n6, n8}	i:10 length:4	"000b"
18	Base32	encodeBase32 ()			P3:{n1,n2,n4, n5,n4,n6,n7}	i:-31 length:4	"-000z"
19	Base32	encodeBase32 ()			P4:{n1,n2,n4, n5,n4,n6,n8}	i:-10 length:4	"-000b"
20	Base32	padLeftWithZe rosToLength()		Figure/ Figure6.png	P1:{n1,n2}	s:"29jw" length:4	"29jw"
21	Base32	padLeftWithZe rosToLength()			P2:{n1,n3,n4, n5,n4,n6}	s:"29jw" length:5	"029jw"

The details of the design are given below:

The Excel file of test cases…

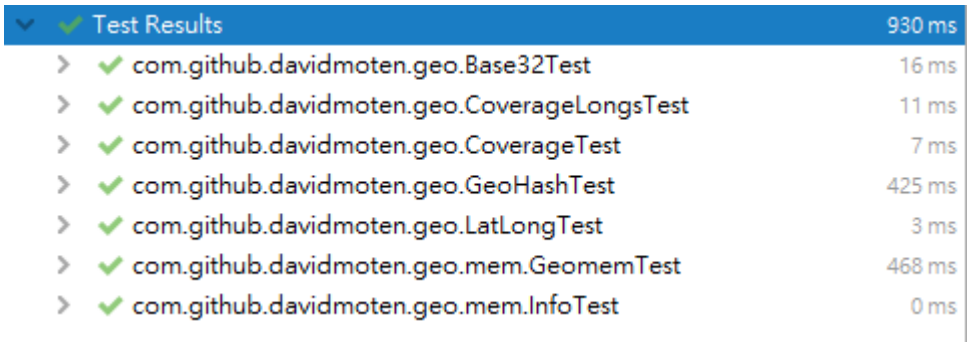
3 Test Implementation

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 link (or JUnit files).

No.	Test method	Source test code
1	`gridAsString_CFG()`	https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java
2	`adjacentHash_CFG_P1()`	
3	`adjacentHash_CFG_P2()`	
4	`adjacentHash_CFG_P3()`	
5	`adjacentHash_CFG_P4()`	
6	`adjacentHash_CFG_P5()`	
7	`adjacentHash_CFG_P6()`	
8	`widthDegrees_CFG_P1()`	
9	`widthDegrees_CFG_P2()`	
10	`decodeBase32_CFG_P1()`	https://stv.csie.ntut.edu.tw/108598007/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java
11	`decodeBase32_CFG_P2()`	
12	`decodeBase32_CFG_P3()`	
13	`decodeBase32_CFG_P4()`	
14	`decodeBase32_CFG_P5()`	
15	`decodeBase32_CFG_P6()`	
16	`encodeBase32_CFG_P1()`	
17	`encodeBase32_CFG_P2()`	
18	`encodeBase32_CFG_P3()`	
19	`encodeBase32_CFG_P4()`	
20	`padLeftWithZerosToLength_CFG_P1()`	
21	`padLeftWithZerosToLength_CFG_P2()`	

4 Test Results

4.1 JUnit test result snapshot



Test Summary

130	0	0	0.327s
tests	failures	ignored	duration

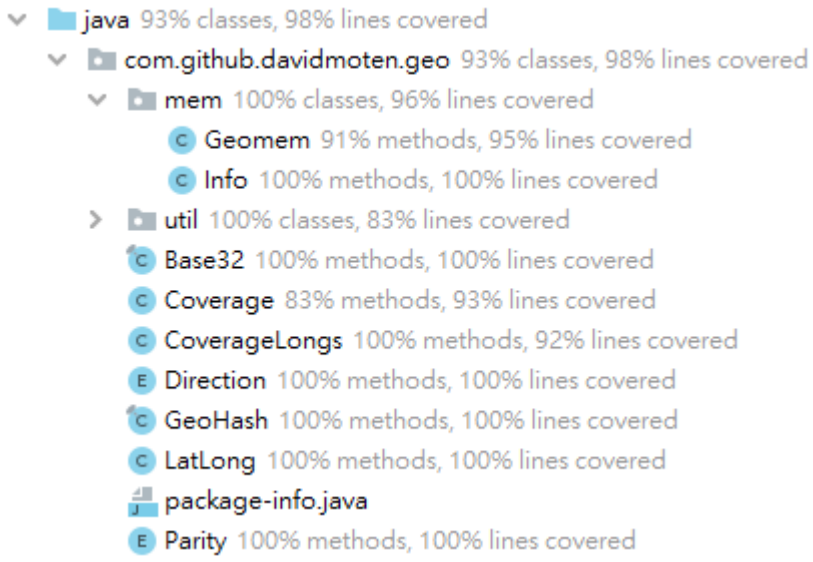
100% successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	128	0	0	0.191s	100%
com.github.davidmoten.geo.mem	2	0	0	0.136s	100%

4.2 Code coverage snapshot

- Coverage of each selected method under test



- Total coverage

geo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		98%		92%	13	149	2	348	1	68	0	10
com.github.davidmoten.geo.mem		96%		65%	8	30	2	62	1	20	0	3
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	44 of 2,327	98%	20 of 186	89%	22	183	5	416	2	90	0	14

4.3 CI result snapshot (3 iterations for CI)

- CI#1

README.md

- CI#2

README.md

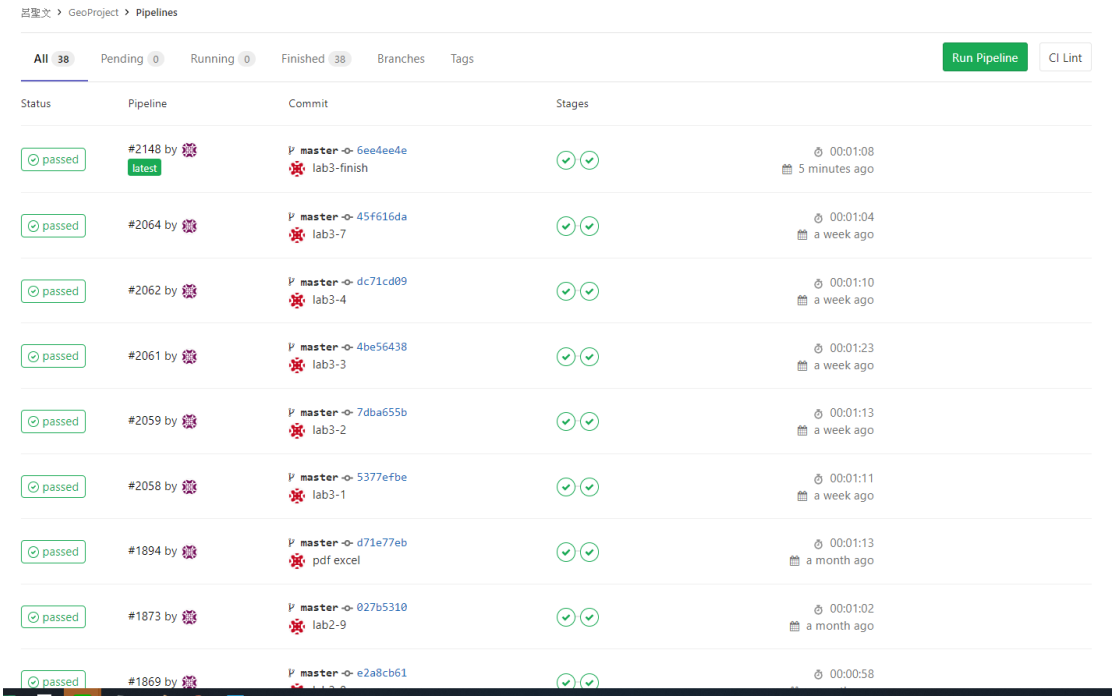
- CI#3

README.md

- CI#4

README.md

- CI Pipeline



5 The Coverage Comparison

The code coverage of Lab1 (and/or Lab2) and Lab3 are listed in the below Table. The results show that the statement and branch coverage are increased from X% to Y% in Lab3.

No.	Test method	Lab2)		Lab3	
		statement coverage	branch coverage	statement coverage	branch coverage
1	gridAsString()	0	0	100	100
2	adjacentHash()	98	97	100	100
3	widthDegrees()	70	50	100	100
4	decodeBase32()	100	100	100	100
5	encodeBase32()	100	100	100	100
6	padLeftWithZerosToLength()	100	100	100	100

6 Summary

In Lab 3, **21 test cases have been designed and implemented using JUnit and the basis path/graph coverage technique**. The test is conducted in 4 CI and **the execution results of the 6 test methods are all passed. The total statement and branch coverage of the test are 95% and 100%, respectively.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are …

本次作業使用 CFG 方式進行 21 個測試，並達到 coverage 91%，這次做完才知道有些路徑是不可能達到，因判斷中會有相同的變數，在不改變變數的情況不可能走另一個路徑，導致 basic path 會有無法執行的路徑。

經過這次作業 CFG 畫圖讓我更能夠快速了解到函數的流程，並且藉由圖形來分析路徑能夠大幅提升測試效率，減少思考如何執行路徑的時間。