Lab 1 Report

林益均
110598110
2022/03/11

1 Test Plan

1.1 Test requirements

The Lab 1 requires to (1) select 22 **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 60%*".

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	2022/2/25
2	Learn JUnit	3	2022/3/1
3	Design test cases for the selected methods	2	2022/3/3
4	Implement test cases	3	2022/3/4
5	Perform test	2	2022/3/7
6	Complete Lab1 report	2	2022/3/11

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 60%*.

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	Base32	encodebase32()		75324,4	29jw
2	Base32	encodebase32()		-75324,4	-29jw
3	Base32	encodebase32()		75324,7	00029jw
4	Base32	encodebase32()		4	000000000004
5	Base32	decodebase32()		29jw	75324
6	Base32	decodebase32()		-29jw	-75324
7	Coverage	Coverage()	CoverageLongs		CoverageLongs
8	Coverage	Coverage()	CoverageLongs		CoverageLongs
9	Coverage	getHashes()	Coverage		29jw
10	Coverage	getHashLength()	Coverage		4
11	Coverage	getHashLength()	Coverage		0
12	Coverage	Tostring()	Coverage		Coverage [hashes=29jw, ratio=4.0]
13	CoverageLongs	CoverageLongs()	CoverageLongs		CoverageLongs
14	CoverageLongs	CoverageLongs()	CoverageLongs		CoverageLongs
15	CoverageLongs	getHashLength()	CoverageLongs		1
16	CoverageLongs	getHashLength()	CoverageLongs		0
17	CoverageLongs	getCount()	CoverageLongs		1
18	GeoHash	right()		29jw	29jy
19	GeoHash	left()		29jw	29jq
20	GeoHash	top()		29jw	29jx
21	GeoHash	bottom()		29jw	29jt
22	GeoHash	adjacentHash()		29jw,Direction.BOTTOM,1	29jt
23	GeoHash	adjacentHash()		29jw,Direction.B	29jx

				OTTOM, -1	
24	GeoHash	neighbours()		29jw	[29jq,29jy,29jx,29jt,29jr,29jm,29jz,29jv]
25	GeoHash	encodeHash()		1,1,1	s
26	GeoHash	hashLengthToCoverBoundingBox()		1,1,1,1	12
27	Direction	opposite()	Direction.BOTTOM		Direction.TOP
28	Direction	opposite()	Direction.TOP		Direction.BOTTOM
29	Direction	opposite()	Direction.LEFT		Direction.RIGHT
30	Direction	opposite()	Direction.RIGHT		Direction.LEFT
31	LatLong	gatLat()	LatLong	1	1
32	LatLong	gatLon()	LatLong	2	2
33	LatLong	add()	LatLong	1,1	2,3
34	LatLong	toString()	LatLong		LatLong [lat=1, lon=2]

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

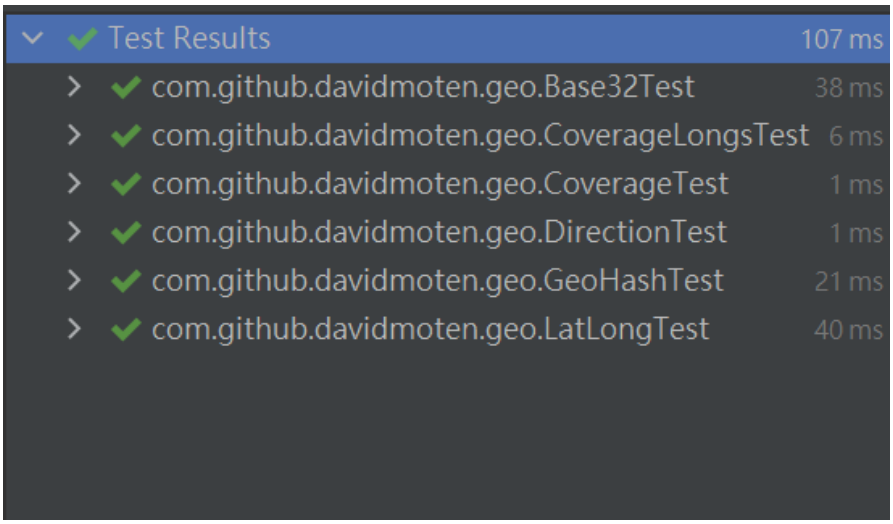
1	encodeBase32()	`@Test`
		`public void encodeBase32() throws Exception {`
		`  String encode = Base32.encodeBase32( i: 75324, length: 4);`
		`  assertEquals( expected: "29jw", encode);`
		`  encode = Base32.encodeBase32( i: -75324, length: 4);`
		`  assertEquals( expected: "-29jw", encode);`
		`  encode = Base32.encodeBase32( i: 75324, length: 7);`
		`  assertEquals( expected: "00029jw", encode);`
		`  encode = Base32.encodeBase32( i: 4);`
		`  assertEquals( expected: "000000000004", encode);`
		`}`

2	adjacentHash()	`@Test`
		`public void adjacentHash() throws Exception{`
		`  String str = GeoHash.adjacentHash(hash,direction, steps: 1);`
		`  assertEquals( expected: "29jt",str);`
		`  str = GeoHash.adjacentHash(hash,direction, steps: -1);`
		`  assertEquals( expected: "29jx",str);`
		`}`

3	neighbours()	`@Test`
		`public void neighbours() throws Exception{`
		`  List<String> res = Arrays.asList("29jq", "29jy", "29jx", "29jt", "29jr", "29jm", "29jz", "29jv");`
		`  List<String> list = GeoHash.neighbours(hash);`
		`  assertEquals(res,list);`
		`}`

4 Test Results

4.1 JUnit test result snapshot



Test Summary

20 tests	0 failures	0 ignored	0.244s duration

100% successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	20	0	0	0.244s	100%

4.2 Code coverage snapshot

- Coverage of each selected method

Element	Class, %	Method, %	Line, %
mem	0% (0/6)	0% (0/38)	0% (0/104)
util	100% (2/2)	66% (4/6)	50% (8/16)
Base32	100% (1/1)	85% (6/7)	93% (42/45)
Coverage	100% (1/1)	83% (5/6)	93% (15/16)
CoverageLongs	100% (1/1)	83% (5/6)	92% (13/14)
Direction	100% (1/1)	100% (2/2)	100% (9/9)
GeoHash	40% (2/5)	56% (26/46)	64% (162/252)
LatLong	100% (1/1)	100% (5/5)	100% (14/14)
Parity	100% (1/1)	100% (1/1)	100% (2/2)

- Total coverage

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		72%		60%	61	149	92	348	19	68	2	10
com.github.davidmoten.geo.mem		0%		0%	30	30	61	61	20	20	3	3
com.github.davidmoten.geo.util		36%		50%	2	4	2	6	0	2	0	1
Total	892 of 2,326	61%	86 of 186	53%	93	183	155	415	39	90	5	14

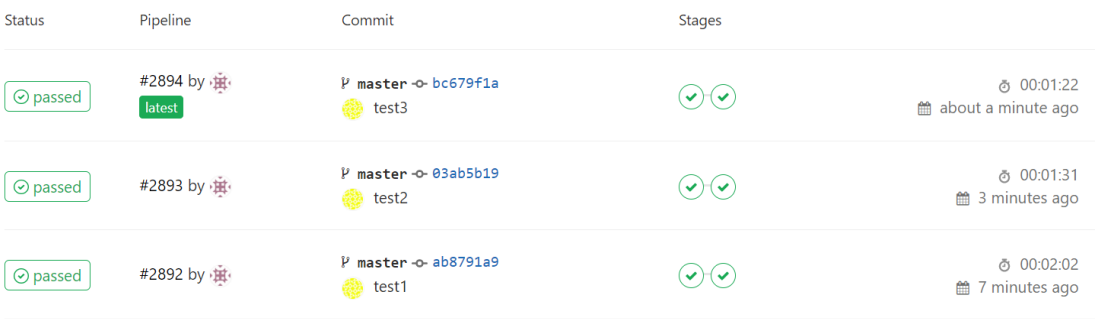
4.3 CI result snapshot (3 iterations for CI)

- CI#1

- CI#2

- CI#3

- CI Pipeline



5 Summary

In Lab 1, 34 **test cases have been designed and implemented using JUnit**. The test is conducted in 3 CI and **the execution results of the 34 test methods are all passed**. **The total statement coverage of the test is 61%.** Thus, the test requirements described in Section 1 are satisfied.