Lab 2 Report

張嗣岱
107598069
3/31

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

(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	4	3/26

2	Design test cases for the selected methods	1.5	3/29
3	Implement test cases	2	3/29
5	Perform test	1	3/29
6	Complete excel	2	3/31
6	Complete Lab1 report	2	3/31

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.

No.	Class	Method	Inputs	Expected Outputs
1	Base32	encodeBase32(i,length)	1,1	“1”
2	Base32	encodeBase32(i)	1	000000000001
3	Base32	decodeBase32(hash)	“123”	1091
4	Base32	getCharIndex(ch)	‘1’	1
5	Base32	padLeftWithZerosToLength(s,length)	“abc”,4	0abc
6	Coverage	getHashes()	Null	[abc]
7	Coverage	getRatio()	Null	0
8	Coverage	getHashLength()	Null	3
9	Coverage	toString()	Null	Coverage [hashes=[abc], ratio=0.0]
10	CoverageLongs	getHashes()	Null	1
11	CoverageLongs	getRatio()	Null	1.1
12	CoverageLongs	getHashLength()	Null	1
13	CoverageLongs	getCount()	Null	1
14	GeoHash	adjacentHash(hash,direction)	“111”,Direction.Left	“110”
15	GeoHash	right(hash)	“k”	“m”
16	GeoHash	left(hash)	“k”	“7”

17	GeoHash	top(hash)	“k”	“s”
18	GeoHash	bottom(hash)	“k”	“h”
19	GeoHash	adjacentHash(hash,direction,step)	“111”,Direction.Top,1	“113”
20	GeoHash	Neighbors(hash)	“1”	[0,4,3,j,2,h,6,n]
21	GeoHash	encodeHash(latitude,longitude)	45,45	“v00000000000”
22	GeoHash	encodeHash(latitude,longitude,length)	45,45,2	“v0”
23	GeoHash	heightDegrees(int n)	15	4.190951585769653E-8
24	GeoHash	widthDegrees(int n)	15	1.3096723705530167E-9
25	GeoHash	fromLongToString(long hash)	1	“0”
26	GeoHash	decodeHash(String geohash)	“1”	"LatLong [lat=-67.5, lon=-112.5]"
27	GeoHash	encodeHash(LatLong p)	45,45	“v00000000000”
28	GeoHash	encodeHash(LatLong p, int length)	45,45,2	“v0”
29	GeoHash	encodeHashToLong(double latitude, double longitude, int length)	45,45,0	0
30	GeoHash	hashLengthToCoverBoundingBox(double topLeftLat, double topLeftLon,double bottomRightLat, double bottomRightLon)	10,10,10,10	12
31	GeoHash	Coverage coverBoundingBoxMaxHashes(double topLeftLat, final double topLeftLon, final double bottomRightLat, final double bottomRightLon, int maxHashes)	10,10,10,10,1	1

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 code

1	Base32.encodeBase32(i, length)	`@Test`
		`public void encodeBase32()`
		`{ assertEquals("1",Base32.encodeBase32(1,1));`
		`assertEquals("1",Base32.encodeBase32(1,-1));`
		`assertEquals("-1",Base32.encodeBase32(-1,1));`

		`assertEquals("-1",Base32.encodeBase32(-1,-1));`
		`}`

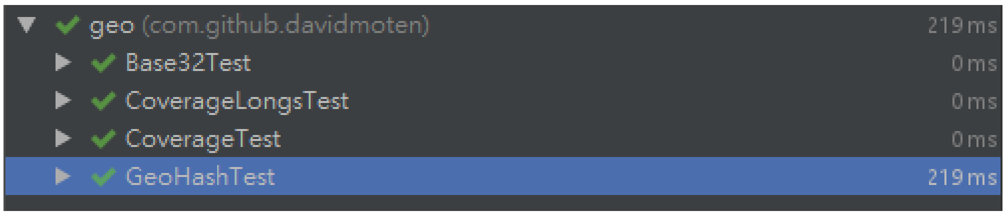
2	GeoHash.right(hash)	`@Test`
		`public void right() {`
		`    //e// assertEquals("m",GeoHash.right("k"));`
		`    assertEquals("ks",GeoHash.right("kk"));`
		`    assertEquals("b",GeoHash.right("a"));`
		`    assertEquals("bp",GeoHash.right("aa"));`
		`}`

3	GeoHash.hashLengthToCoverBoundingBox(
	double topLeftLat,
	double topLeftLon,
	double bottomRightLat,
	double bottomRightLon)
		`@Test`
		`public void hashLengthToCoverBoundingBox() {`
		`    //e//`
		`    assertEquals(12,`
		`GeoHash.hashLengthToCoverBoundingBox(10,10,10,10));`
		`    assertEquals(0,`
		`GeoHash.hashLengthToCoverBoundingBox(10,10,10,-10));`
		`    assertEquals(0,`
		`GeoHash.hashLengthToCoverBoundingBox(10,10,-10,10));`
		`    assertEquals(0,`
		`GeoHash.hashLengthToCoverBoundingBox(10,10,-10,-`
		`10));`
		
		`    assertEquals(0,`
		`GeoHash.hashLengthToCoverBoundingBox(10,-10,10,10));`
		`    assertEquals(12,`
		`GeoHash.hashLengthToCoverBoundingBox(10,-10,10,-`
		`10));`
		`    assertEquals(0,`
		`GeoHash.hashLengthToCoverBoundingBox(10, -10, -10,`
		`10));`
		`    assertEquals(0,`
		`GeoHash.hashLengthToCoverBoundingBox(10,-10,-10,-`
		`10));`
		
		`    assertEquals(0,`

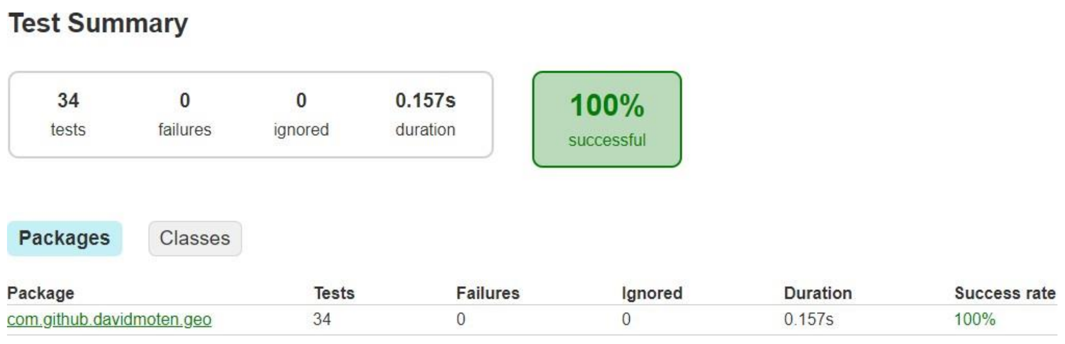
		GeoHash.*hashLengthToCoverBoundingBox*(-10, 10, 10,10));
		    assertEquals(0,
		GeoHash.*hashLengthToCoverBoundingBox*(-10,10,10,-
		10));
		    assertEquals(12,
		GeoHash.*hashLengthToCoverBoundingBox*(-10, 10, -10,
		10));
		    assertEquals(0,
		GeoHash.*hashLengthToCoverBoundingBox*(-10,10,-10,-
		10));
		
		    assertEquals(0,
		GeoHash.*hashLengthToCoverBoundingBox*(-10, -10, 10,
		10));
		    assertEquals(0,
		GeoHash.*hashLengthToCoverBoundingBox*(-10,-10,10,-
		10));
		    assertEquals(0,
		GeoHash.*hashLengthToCoverBoundingBox*(-10, -10, -10,
		10));
		    assertEquals(12,
		GeoHash.*hashLengthToCoverBoundingBox*(-10,-10,-10,-
		10));
		}

4 Test Results

4.1 JUnit test result snapshot



```
▼ ✓ geo (com.github.davidmoten)        219 ms
  ▶ ✓ Base32Test                          0 ms
  ▶ ✓ CoverageLongsTest                   0 ms
  ▶ ✓ CoverageTest                        0 ms
  ▶ ✓ GeoHashTest                       219 ms
```



Test Summary

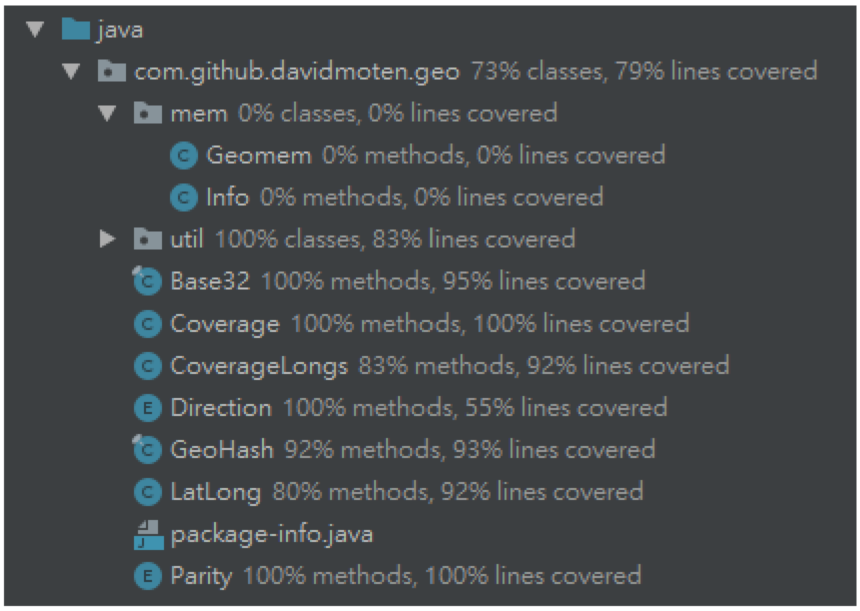
34	0	0	0.157s	100%
tests	failures	ignored	duration	successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	34	0	0	0.157s	100%

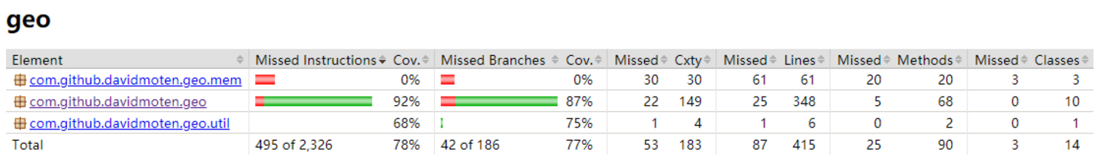
4.2 Code coverage snapshot

- Coverage of each selected method



```
▼ java
  ▼ com.github.davidmoten.geo 73% classes, 79% lines covered
    ▼ mem 0% classes, 0% lines covered
        Geomem 0% methods, 0% lines covered
        Info 0% methods, 0% lines covered
    ▶ util 100% classes, 83% lines covered
      Base32 100% methods, 95% lines covered
      Coverage 100% methods, 100% lines covered
      CoverageLongs 83% methods, 92% lines covered
      Direction 100% methods, 55% lines covered
      GeoHash 92% methods, 93% lines covered
      LatLong 80% methods, 92% lines covered
      package-info.java
      Parity 100% methods, 100% lines covered
```

- Total coverage

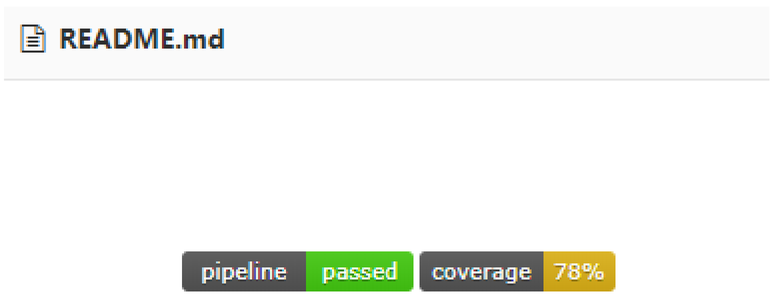


geo

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		92%		87%	22	149	25	348	5	68	0	10
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	495 of 2,326	78%	42 of 186	77%	53	183	87	415	25	90	3	14

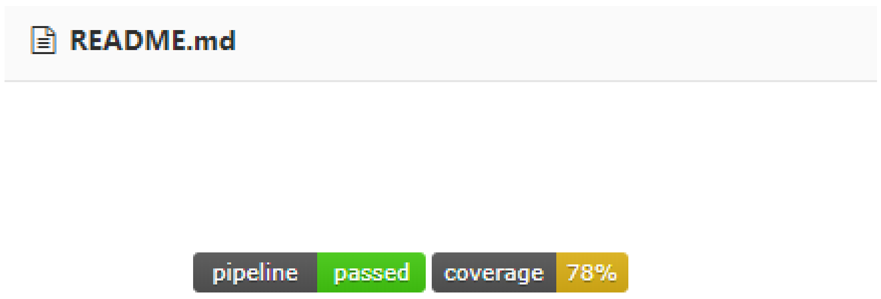
4.3 CI result snapshot (3 iterations for CI)

- CI#1

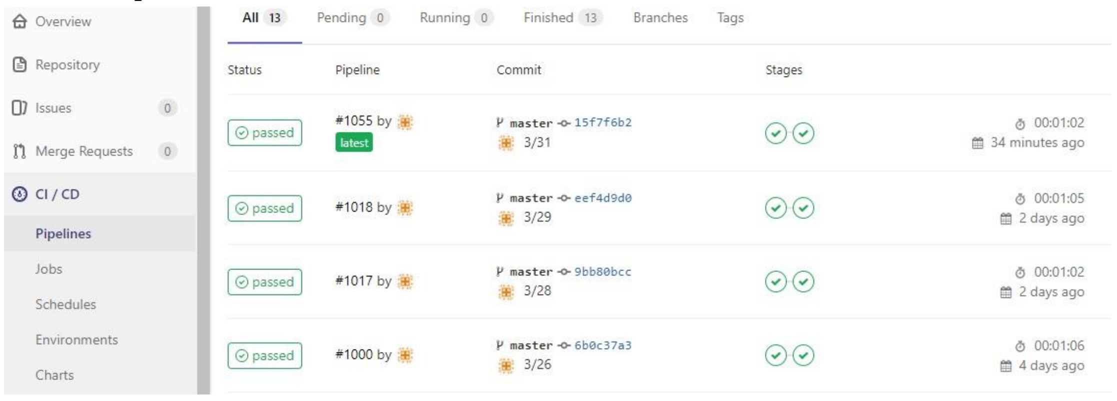


- CI#2

- CI#3



- CI Pipeline



Status	Pipeline	Commit	Stages	
passed	#1055 by latest	master 15f7f6b2 3/31	✓ ✓	00:01:02 34 minutes ago
passed	#1018 by	master eef4d9d0 3/29	✓ ✓	00:01:05 2 days ago
passed	#1017 by	master 9bb80bcc 3/28	✓ ✓	00:01:02 2 days ago
passed	#1000 by	master 6b0c37a3 3/26	✓ ✓	00:01:06 4 days ago

5 Summary

In Lab 2, **24 test cases have been designed and implemented using JUnit and the ISP technique**. 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 78%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are all combinations spent a lot time to design it.