Lab 1 Report

Name: 許哲維
Student ID: 111598066
Date: 2023.03.08

1 Test Plan

1.1 Test requirements

The Lab 1 requires to (1) select **20 methods** from **6 classes** of the 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 50%*".

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	2023.03.01
2	Learn JUnit	3	2023.03.02
3	Design test cases for the selected methods	3	2023.03.02
4	Implement test cases	4.5	2023.03.06
5	Perform test	4.5	2023.03.07
6	Complete Lab1 report	2	2023.03.08

1.4 Success criteria

All test cases designed for the selected methods must pass (or "85% of all test cases must pass) and *the statement coverage should have achieved at least 50%*.

2 Test Design

To fulfill the test requirements listed in section 1.1, the following methods are selected and corresponding test cases are designed.

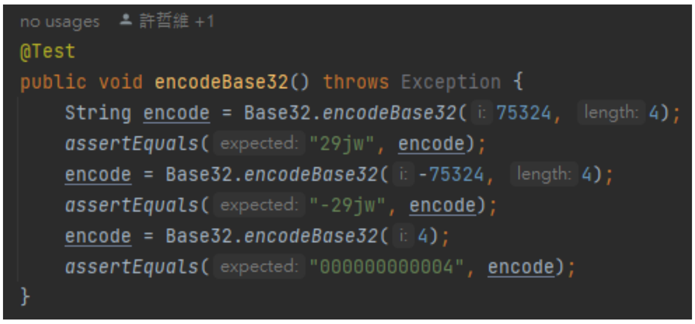
No.	Class	Method	Test Objective	Inputs	Expected Outputs
1	Base32	encodeBase32()		75324, 4	29jw
2	Base32	encodeBase32()		75324, -4	-29jw
3	Base32	encodeBase32()		4	000000000004
4	Base32	decodeBase32()		29jw	75324
5	Base32	getCharIndex()		0	0
6	Base32	padLeftWithZerosToLength()		Lucy, 5	0Lucy
7	Base32	padLeftWithZerosToLength()		JenJen, 6	JenJen
8	Coverage	getHashes()	Coverage		Katayama, Jenjen, not, Chinchin.
9	Coverage	getRatio()	Coverage		4.0
10	Coverage	getHashLength()	Coverage		9
11	Coverage	toString()	Coverage		Coverage [hashes=[Chinchin., not, Katayama, Jenjen], ratio=4.0]
12	CoverageLongs	getHashLength()	CoverageLongs		3
13	CoverageLongs	getHashLength()	CoverageLongs		0
14	CoverageLongs	toString()	CoverageLongs		"Coverage [hashes="+c.getHashes()+", ratio="+c.getRatio()+"]"
15	CoverageLongs	getCount()	CoverageLongs		1

16	Direction	opposite()		Direction.RIGHT	Direction.LEFT
17	Direction	opposite()		Direction.LEFT	Direction.RIGHT
18	Direction	opposite()		Direction.TOP	Direction.BOTTOM
19	Direction	opposite()		Direction.BOTTOM	Direction.TOP
20	GeoHash	adjacentHash()		"29jw",Direction.BOTTOM,1	"29jt"
21	GeoHash	neighbours()		"29jw"	"29jq"
22	GeoHash	hashLengthToCoverBoundingBox()		2,2,1,1	2
23	GeoHash	coverBoundingBoxLongs()		0.1,0.1,0.1,0.1,1	1
24	LatLong	getLat()	LatLong		6
25	LatLong	toString()	LatLong		"LatLong [lat="+lat+", lon="+lon+"]"

3 Test Implementation

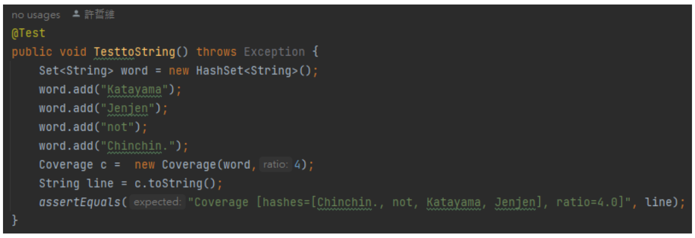
The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 7 selected test cases are given below. The rest of test script implementations can be found in the link (or JUnit files).

<table>
<tr><th>No.</th><th>Test method</th><th>Source code</th></tr>
<tr><td>1</td><td>encodeBase32()</td><td>



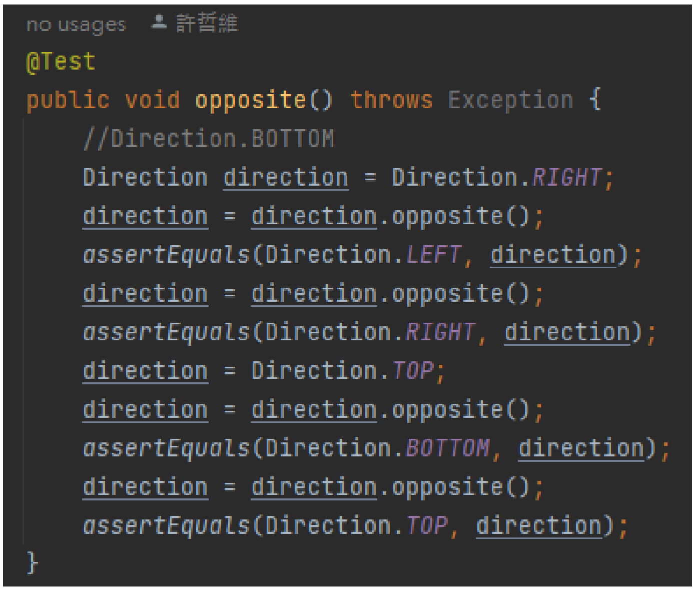
```
no usages  許哲維 +1
@Test
public void encodeBase32() throws Exception {
    String encode = Base32.encodeBase32( i: 75324,  length: 4);
    assertEquals( expected: "29jw", encode);
    encode = Base32.encodeBase32( i: -75324,  length: 4);
    assertEquals( expected: "-29jw", encode);
    encode = Base32.encodeBase32( i: 4);
    assertEquals( expected: "000000000004", encode);
}
```

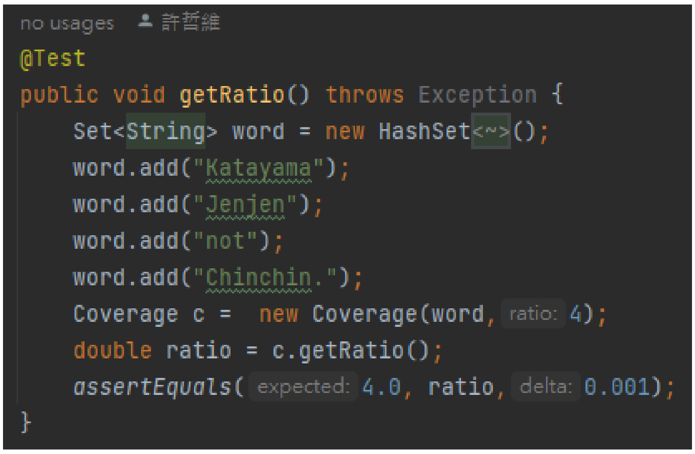
</td></tr>
<tr><td>2</td><td>TesttoString()</td><td>

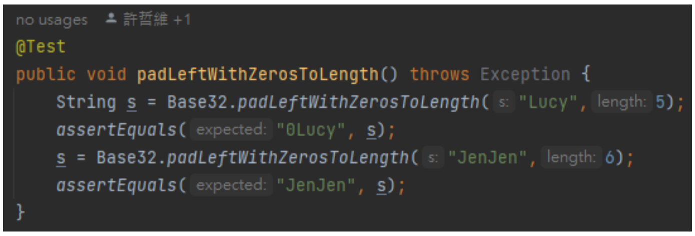


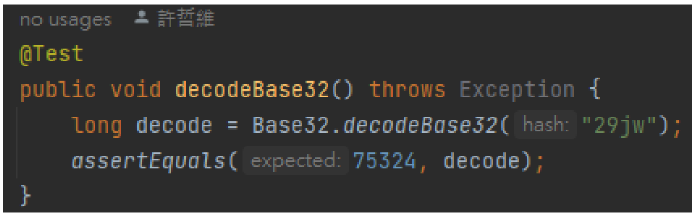
```
no usages  許哲維
@Test
public void TesttoString() throws Exception {
    Set<String> word = new HashSet<String>();
    word.add("Katayama");
    word.add("Jenjen");
    word.add("not");
    word.add("Chinchin.");
    Coverage c =  new Coverage(word, ratio: 4);
    String line = c.toString();
    assertEquals( expected: "Coverage [hashes=[Chinchin., not, Katayama, Jenjen], ratio=4.0]", line);
}
```

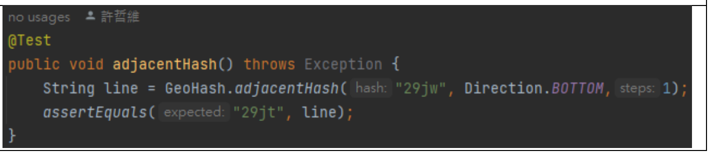
</td></tr>
</table>

3	opposite()	
		no usages 👤 許哲維
		@Test
		public void opposite() throws Exception {
		//Direction.BOTTOM
		Direction direction = Direction.RIGHT;
		direction = direction.opposite();
		assertEquals(Direction.LEFT, direction);
		direction = direction.opposite();
		assertEquals(Direction.RIGHT, direction);
		direction = Direction.TOP;
		direction = direction.opposite();
		assertEquals(Direction.BOTTOM, direction);
		direction = direction.opposite();
		assertEquals(Direction.TOP, direction);
		}

4	getRatio()	
		no usages 👤 許哲維
		@Test
		public void getRatio() throws Exception {
		Set<String> word = new HashSet<~>();
		word.add("Katayama");
		word.add("Jenjen");
		word.add("not");
		word.add("Chinchin.");
		Coverage c = new Coverage(word, ratio: 4);
		double ratio = c.getRatio();
		assertEquals(expected: 4.0, ratio, delta: 0.001);
		}

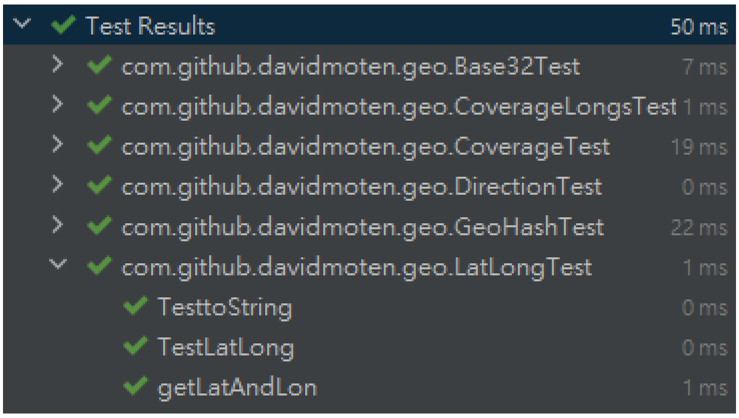
5	padLeftWithZerosToLength()	
		no usages 👤 許哲維 +1
		@Test
		public void padLeftWithZerosToLength() throws Exception {
		String s = Base32.padLeftWithZerosToLength(s: "Lucy", length: 5);
		assertEquals(expected: "0Lucy", s);
		s = Base32.padLeftWithZerosToLength(s: "JenJen", length: 6);
		assertEquals(expected: "JenJen", s);
		}

6	decodeBase32()	
		no usages 👤 許哲維
		@Test
		public void decodeBase32() throws Exception {
		long decode = Base32.decodeBase32(hash: "29jw");
		assertEquals(expected: 75324, decode);
		}

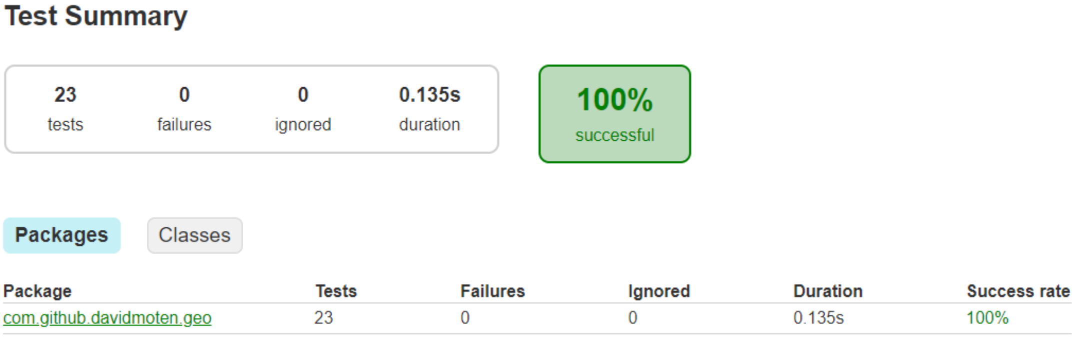
7	adjacentHash()	
		no usages 👤 許哲維
		@Test
		public void adjacentHash() throws Exception {
		String line = GeoHash.adjacentHash(hash: "29jw", Direction.BOTTOM, steps: 1);
		assertEquals(expected: "29jt", line);
		}

4 Test Results

4.1 JUnit test result snapshot



```
Test Results                                      50 ms
  com.github.davidmoten.geo.Base32Test             7 ms
  com.github.davidmoten.geo.CoverageLongsTest      1 ms
  com.github.davidmoten.geo.CoverageTest          19 ms
  com.github.davidmoten.geo.DirectionTest          0 ms
  com.github.davidmoten.geo.GeoHashTest           22 ms
  com.github.davidmoten.geo.LatLongTest            1 ms
    TesttoString                                   0 ms
    TestLatLong                                    0 ms
    getLatAndLon                                   1 ms
```



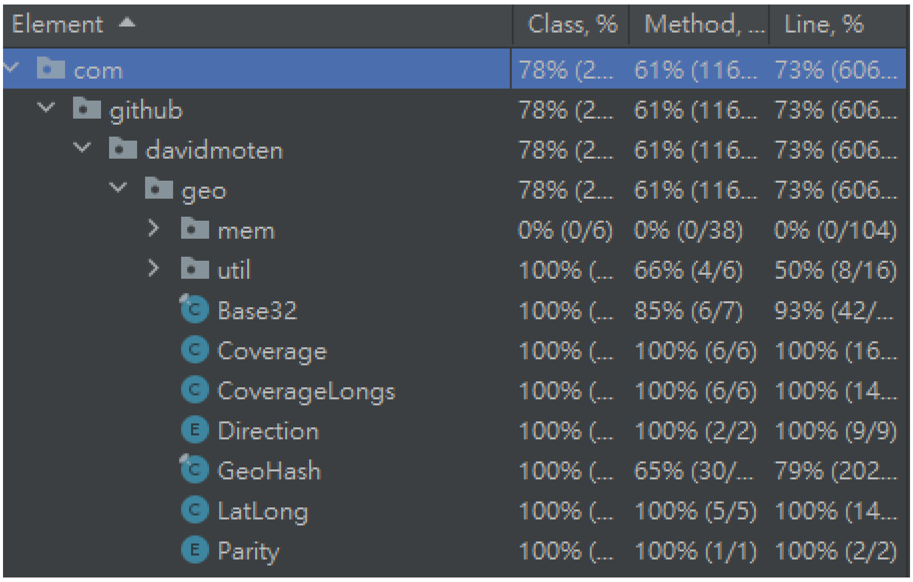
Test Summary

23	0	0	0.135s	100%
tests	failures	ignored	duration	successful

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	23	0	0	0.135s	100%

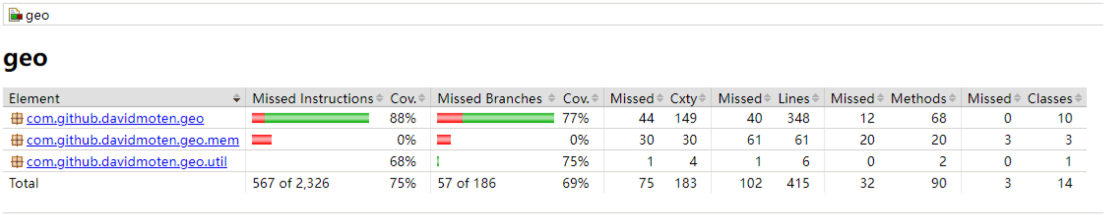
4.2 Code coverage snapshot

- Coverage of each selected method



Element	Class, %	Method, ...	Line, %
com	78% (2...	61% (116...	73% (606...
github	78% (2...	61% (116...	73% (606...
davidmoten	78% (2...	61% (116...	73% (606...
geo	78% (2...	61% (116...	73% (606...
mem	0% (0/6)	0% (0/38)	0% (0/104)
util	100% (...	66% (4/6)	50% (8/16)
Base32	100% (...	85% (6/7)	93% (42/...
Coverage	100% (...	100% (6/6)	100% (16...
CoverageLongs	100% (...	100% (6/6)	100% (14...
Direction	100% (...	100% (2/2)	100% (9/9)
GeoHash	100% (...	65% (30/...	79% (202...
LatLong	100% (...	100% (5/5)	100% (14...
Parity	100% (...	100% (1/1)	100% (2/2)

- Total coverage



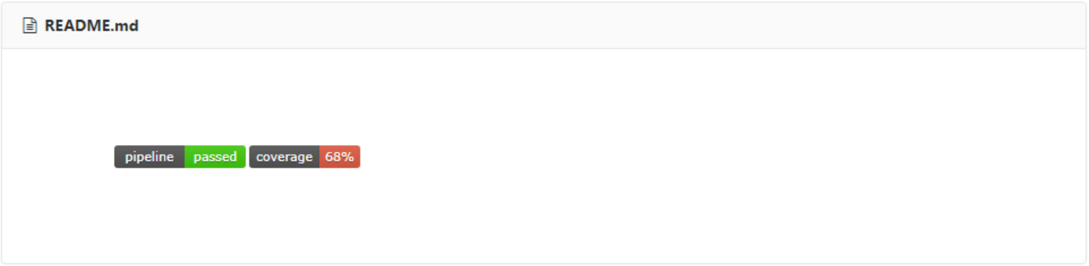
geo

geo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		88%		77%	44	149	40	348	12	68	0	10
com.github.davidmoten.geo.mem		0%		0%	30	30	61	61	20	20	3	3
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	567 of 2,326	75%	57 of 186	69%	75	183	102	415	32	90	3	14

4.3 CI result snapshot (3 iterations for CI)

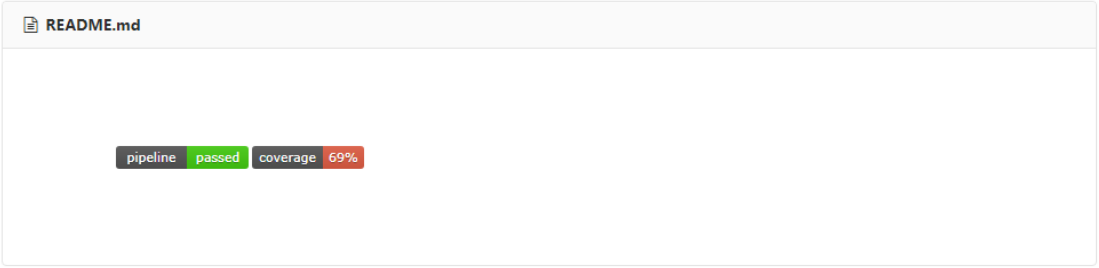
- CI#1



README.md

pipeline passed coverage 68%

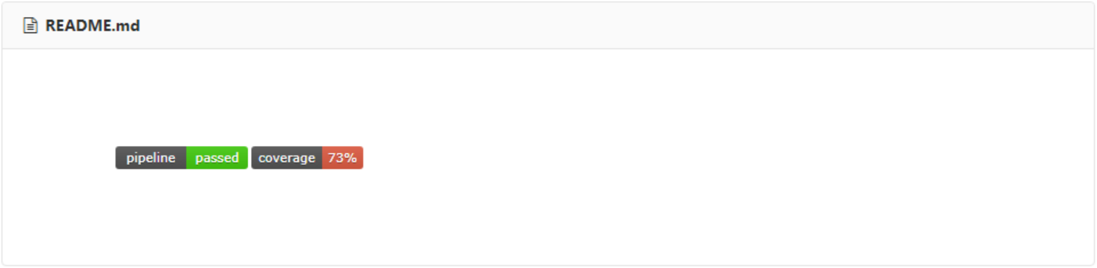
- CI#2



README.md

pipeline passed coverage 69%

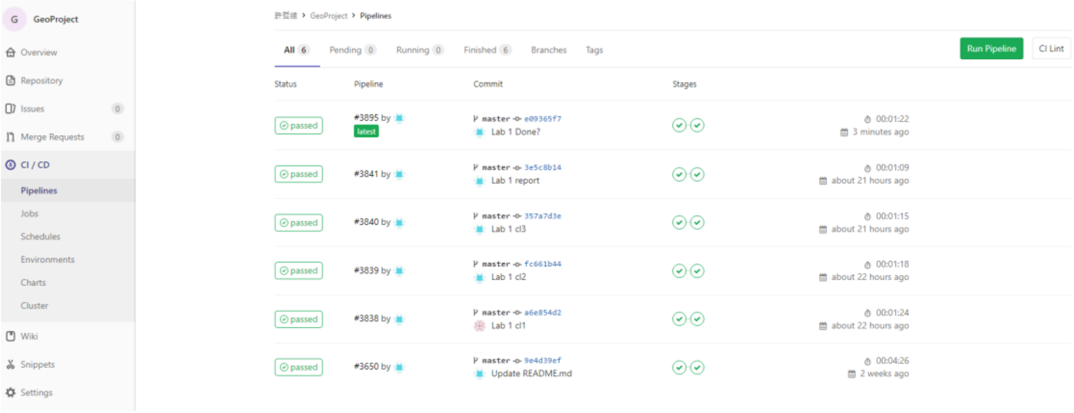
- CI#3



README.md

pipeline passed coverage 73%

- CI Pipeline



Status	Pipeline	Commit	Stages	
passed	#3895 by latest	master e09365f7 Lab 1 Done?	✓ ✓	00:01:22 3 minutes ago
passed	#3841 by	master 3e5c8b14 Lab 1 report	✓ ✓	00:01:09 about 21 hours ago
passed	#3840 by	master 357a7d3e Lab 1 cl3	✓ ✓	00:01:15 about 21 hours ago
passed	#3839 by	master fc661b44 Lab 1 cl2	✓ ✓	00:01:18 about 22 hours ago
passed	#3838 by	master a6e854d2 Lab 1 cl1	✓ ✓	00:01:24 about 22 hours ago
passed	#3650 by	master 9e4d39ef Update README.md	✓ ✓	00:04:26 2 weeks ago

5 Summary

In Lab 1, **25 test cases have been designed and implemented using JUnit**. The test is conducted in 3 CI and **the execution results of the 25 test methods are all passed**. **The total statement coverage of the test is 73%.** Thus, the test requirements described in Section 1 are satisfied.