Lab 3 Report

Name 張家齊
Student ID 110598109
Date 2022/05/08

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	2	2022/04/30
2	Learn **basis path and**	2	2022/05/01

	graph coverage		
3	Design test cases for the selected methods	8	2022/05/02-04
4	Implement test cases	0.5	2022/05/05
5	Complete Lab3 report	1	2022/05/08

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.
(為避免圖檔連結造成表格過長，此處移除表格，詳細請參閱 excel 表格)
The details of the design are given below:
The Excel file of test cases are put in the same folder with this report.

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	padLeftWithZerosToLength()	`@Test`
		`public void`
		`TestPadLeftWithZerosToLength()`
		`throws Exception {`
		`    assertEquals("HelloWorld",`

		`Base32.padLeftWithZerosToLength("Hel`
		`loWorld", 1));`
		`    assertEquals("0HelloWorld",`
		`Base32.padLeftWithZerosToLength("Hel`
		`loWorld", 11));`
		`    assertEquals("00HelloWorld",`
		`Base32.padLeftWithZerosToLength("Hel`
		`loWorld", 12));`
		`}`

2	encodeBase32()	`@Test`
		`public void TestEncodeBase32()`
		`throws Exception {`
		`    assertEquals("-1",`
		`Base32.encodeBase32(-1, 0));`
		`    assertEquals("1",`
		`Base32.encodeBase32(1, 0));`
		`    assertEquals("-200",`
		`Base32.encodeBase32(-2048, 0));`
		`}`

3	adjacentHash()	`@Test`
		`public void TestAdjacentHash()`
		`throws Exception {`
		`    try {`
		`        GeoHash.adjacentHash(null,`
		`Direction.TOP);`
		`    } catch`
		`( IllegalArgumentException`
		`throwMessage ) {`
		`        assertEquals(  "hash must be`
		`non-null" ,`
		`throwMessage.getMessage());`
		`    }`
		
		`    try {`
		`        GeoHash.adjacentHash("",`
		`Direction.TOP);`
		`    } catch`
		`( IllegalArgumentException`

		throwMessage) {
		        assertEquals("adjacent has
		no meaning for a zero length hash
		that covers the whole world" ,
		throwMessage.getMessage());
		    }
		    assertEquals("y",
		GeoHash.adjacentHash("w",
		Direction.TOP));
		    assertEquals("wt",
		GeoHash.adjacentHash("ws",
		Direction.TOP));
		    assertEquals("wsw",
		GeoHash.adjacentHash("wsq",
		Direction.TOP));
		    assertEquals("zzzzzzzzzzzb",
		GeoHash.adjacentHash("gzzzzzzzzzzz",
		Direction.TOP));
		
		}

	gridAsString()	@Test
		public void TestGridToString()
		throws Exception {
		    Set<String> hashes = new
		HashSet<String>();
		    hashes.add("f2");
		    hashes.add("f8");
		    assertEquals("",
		GeoHash.gridAsString("dr",1,1,-1,-1,
		hashes));
		    assertEquals("\n",
		GeoHash.gridAsString("dr",1,1,-1,1,
		hashes));
		    assertEquals("dw \n",
		GeoHash.gridAsString("dr",1,1,1,1,
		hashes));
		
		    hashes = new HashSet<String>();

		hashes.add("dw");
		    assertEquals("DW \n",
		GeoHash.gridAsString("dr",1,1,1,1,
		hashes));
		
		    hashes = new HashSet<String>();
		    hashes.add("he");
		    hashes.add("we");
		    assertEquals("dq dw \n",
		GeoHash.gridAsString("dr",0,1,1,1,
		hashes));
		    assertEquals("dx \ndw \n",
		GeoHash.gridAsString("dr",1,0,1,1,
		hashes));
		}

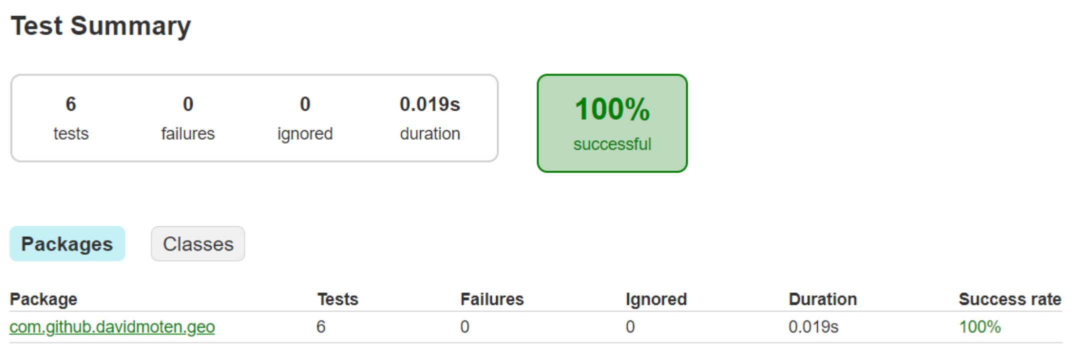
	calculateWidthDegrees()	@Test
		public void
		TestCalculateWidthDegrees() throws
		Exception {
		
		assertEquals(1.0231815394945443E-11,
		GeoHash.widthDegrees(18), 0.001);
		
		assertEquals(1.2789769243681803E-12,
		GeoHash.widthDegrees(19), 0.001);
		}

	coverBoundingBoxMaxHashes()	@Test
		public void
		TestCoverBoundingBoxMaxHashes()
		throws Exception {
		    Coverage coverage = null;
		    coverage =
		GeoHash.coverBoundingBoxMaxHashes(1,
		0,0,1,5);
		    assertEquals("[s00]",
		coverage.getHashes().toString());
		
		    coverage =

		`GeoHash.coverBoundingBoxMaxHashes(0,` `0,0,0,4);` `assertEquals("[s00000000000]",` `coverage.getHashes().toString());` `}`

4 Test Results

4.1 JUnit test result snapshot



Test Summary

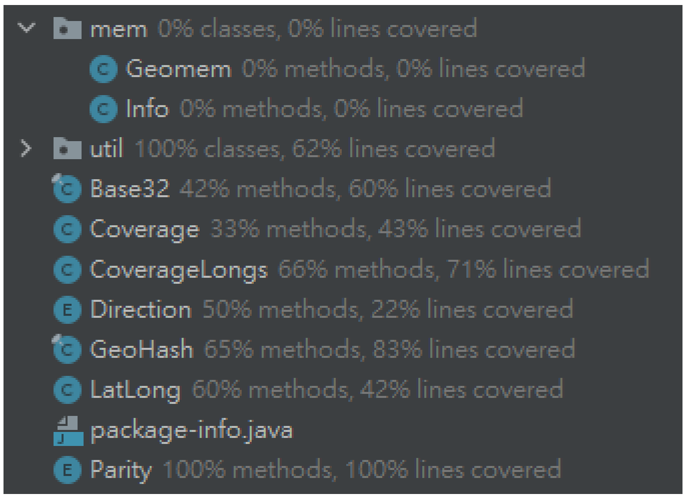
6	0	0	0.019s	100%
tests	failures	ignored	duration	successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	6	0	0	0.019s	100%

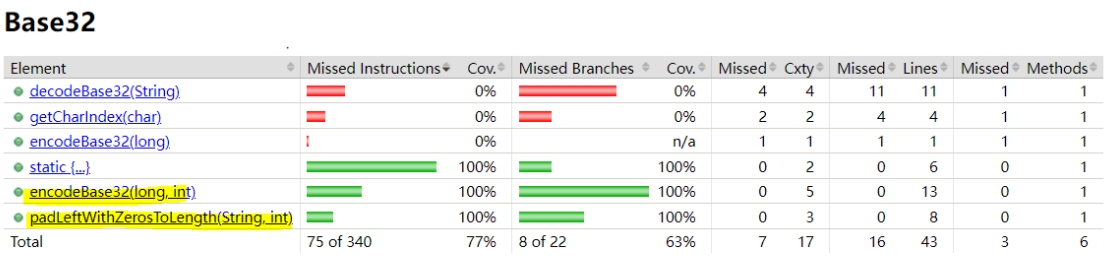
4.2 Code coverage snapshot

- Coverage of each selected method under test



```
mem 0% classes, 0% lines covered
    Geomem 0% methods, 0% lines covered
    Info 0% methods, 0% lines covered
util 100% classes, 62% lines covered
  Base32 42% methods, 60% lines covered
  Coverage 33% methods, 43% lines covered
  CoverageLongs 66% methods, 71% lines covered
  Direction 50% methods, 22% lines covered
  GeoHash 65% methods, 83% lines covered
  LatLong 60% methods, 42% lines covered
  package-info.java
  Parity 100% methods, 100% lines covered
```

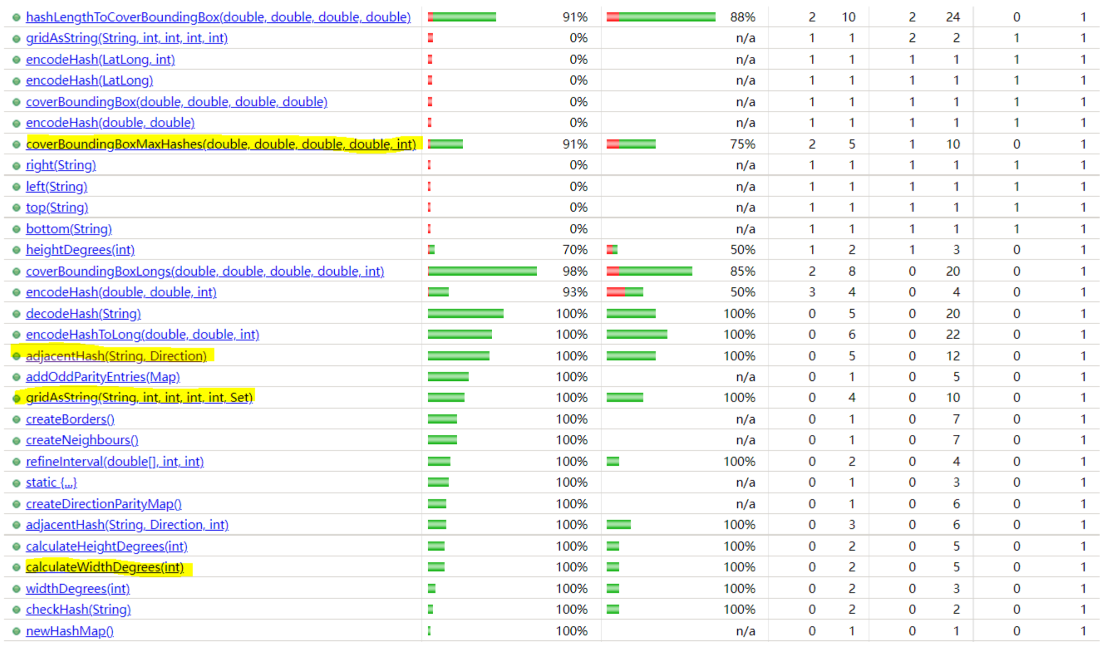
- Method in Base32 coverage



Base32

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
decodeBase32(String)	🟥	0%	🟥	0%	4	4	11	11	1	1
getCharIndex(char)	🟥	0%	🟥	0%	2	2	4	4	1	1
encodeBase32(long)	🟥	0%		n/a	1	1	1	1	1	1
static {...}	🟩	100%	🟩	100%	0	2	0	6	0	1
encodeBase32(long, int)	🟩	100%	🟩	100%	0	5	0	13	0	1
padLeftWithZerosToLength(String, int)	🟩	100%	🟩	100%	0	3	0	8	0	1
Total	75 of 340	77%	8 of 22	63%	7	17	16	43	3	6

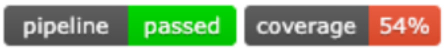
- Method in GeoHash coverage



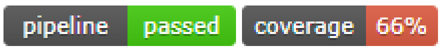
hashLengthToCoverBoundingBox(double, double, double, double)	91%	88%	2	10	2	24	0	1
gridAsString(String, int, int, int, int)	0%	n/a	1	1	2	2	1	1
encodeHash(LatLong, int)	0%	n/a	1	1	1	1	1	1
encodeHash(LatLong)	0%	n/a	1	1	1	1	1	1
coverBoundingBox(double, double, double, double)	0%	n/a	1	1	1	1	1	1
encodeHash(double, double)	0%	n/a	1	1	1	1	1	1
coverBoundingBoxMaxHashes(double, double, double, double, int)	91%	75%	2	5	1	10	0	1
right(String)	0%	n/a	1	1	1	1	1	1
left(String)	0%	n/a	1	1	1	1	1	1
top(String)	0%	n/a	1	1	1	1	1	1
bottom(String)	0%	n/a	1	1	1	1	1	1
heightDegrees(int)	70%	50%	1	2	1	3	0	1
coverBoundingBoxLongs(double, double, double, double, int)	98%	85%	2	8	0	20	0	1
encodeHash(double, double, int)	93%	50%	3	4	0	4	0	1
decodeHash(String)	100%	100%	0	5	0	20	0	1
encodeHashToLong(double, double, int)	100%	100%	0	6	0	22	0	1
adjacentHash(String, Direction)	100%	100%	0	5	0	12	0	1
addOddParityEntries(Map)	100%	n/a	0	1	0	5	0	1
gridAsString(String, int, int, int, int, Set)	100%	100%	0	4	0	10	0	1
createBorders()	100%	n/a	0	1	0	7	0	1
createNeighbours()	100%	n/a	0	1	0	7	0	1
refineInterval(double[], int, int)	100%	100%	0	2	0	4	0	1
static {...}	100%	n/a	0	1	0	3	0	1
createDirectionParityMap()	100%	n/a	0	1	0	6	0	1
adjacentHash(String, Direction, int)	100%	100%	0	3	0	6	0	1
calculateHeightDegrees(int)	100%	100%	0	2	0	5	0	1
calculateWidthDegrees(int)	100%	100%	0	2	0	5	0	1
widthDegrees(int)	100%	100%	0	2	0	3	0	1
checkHash(String)	100%	100%	0	2	0	2	0	1
newHashMap()	100%	n/a	0	1	0	1	0	1

4.3 CI result snapshot (3 iterations for CI)

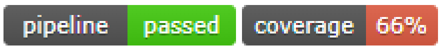
- CI#1



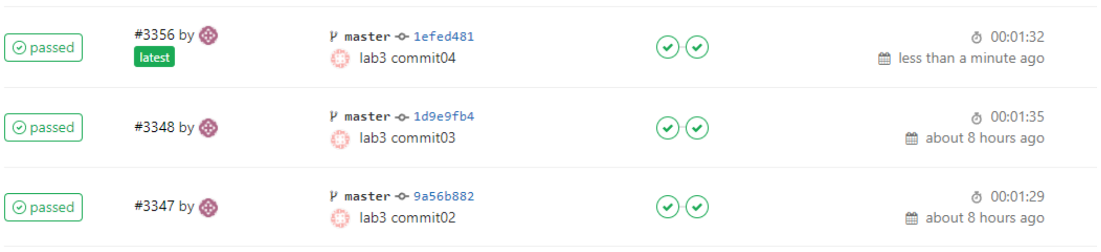
- CI#2



- CI#3



- CI Pipeline


passed #3356 by latest master 1efed481 lab3 commit04 00:01:32 less than a minute ago

passed #3348 by master 1d9e9fb4 lab3 commit03 00:01:35 about 8 hours ago

passed #3347 by master 9a56b882 lab3 commit02 00:01:29 about 8 hours ago

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 100% to 100% in Lab3.此部分因為在 Lab1 使用閱讀程式碼並鑽寫測試的方式，已經抵達 100% banch coverage。

No.	Test method	Lab1 (or Lab2)		Lab3	
		statement coverage	**branch coverage**	**statement coverage**	**branch coverage**
1	padLeftWithZerosToLength()	Both 100%		Both 100%	
2	encodeBase32()	Both 100%		Both 100%	
3	adjacentHash()	Both 100%		Both 100%	

6 Summary

In Lab 3, **6 test cases have been designed and implemented using JUnit and the basis path/graph coverage technique**. The test is conducted in 3 CI and **the execution results of the 6 test methods are all passed**. **The total statement and branch coverage of the test are 91%(in 1 case) and 100%(in 5 cases), respectively.** Thus, the test requirements described in Section 1 are satisfied.

透過 node, edge, prime path 的限制，並且設計相應的 path 與實際 test case，確保並且量化了各個 path 的 coverage。