# Lab 3 Report

Name: 童柏鈞 Student ID:111598027 Date:2022/05/09

#### 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 <u>at least one test case</u> and the both <u>minimum</u> statement (node) and branch (edge) coverage are <u>greater than those of Lab 2</u> and <u>90%</u>, 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	2022/04/23	
2	Learn basis path and	2	2022/04/25	
	graph coverage		_ = = = 1	
3	Design test cases for	2	2022/04/26	
3	the selected methods	2	2022/04/20	
4	Implement test cases	2	2022/04/28	
5	Perform tests and check code coverage. If not satisfy, design more test cases	3	2022/04/29	
6	Complete Lab3 report	1	2022/05/01	

### 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 <u>both statement and branch (or path) coverage should have achieved at least 90%, respectively.</u>

## 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	Base32	getCharI ndex(cha r ch)	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/getchar index.jpg	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/getchar index.jpg	P1:{n1,n 2,n3}	inputs:(ch ='e')	"not a base32 character: e"
2	Base32	encodeB ase32(lo ng I, int length)	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/encode base32.j	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/encode base32.j	P1:{n1,n 2,n4,n5,n 6,n7,n8,n 7,n8,n7n, 8,n7,n9,n 10,n12,n 13}	inputs:(int = 75324)	"29jw"
3	Base32	padLeftW ithZerosT oLength( String s, int length)	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/podleft htozerole	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/podleft htozerole	P1:{n1,n 2,n3,n4,n 5,n4,n6}	inputs:(s ="29jw",I ength=4)	"29jw"

			ngth.jpg	ngth.jpg			
4	GeoHash	fromLong ToString( long hash)	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/fromlon gtostring. jpg	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/fromlon gtostring. jpg	P1:{n1,n 2a,n3}	inputs:(h ash=13)	"invalid long geohash 13"
5	GeoHash	encodeH ashToLo ng(doubl e latitude, double longtitud e, int length)	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/encode hashtolo ng.jpg	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/encode hashtolo ng.jpg	P1:{n1,n 2,n3,n4,n 5,n6,n13, n14,n8,n 9,n10,n1 2,n14,n1 5}	inputs:(lat itude= 37.42199 ,longtitud e=- 122.0840 57 ,length=1 2	"0x9c225c2 6a14d8aL"
6	GeoHash	adjacent Hash(Stri ng hash,Dire ction direction,i nt strps)	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/adjace nthash.jp	https://co urse.sela b.ml/stv- gitlab/11 1598027/ GeoProje ct/blob/m aster/Lab Report/L ab3/Phot o/adjace nthash.jp	P1:{n1,n 2,n3}	Inputs=(h ash="",dir ection=Di rection.B OTTOM)	"adjacent has no meaning for a zero length hash that covers the whole world"

The details of the design are given below: Lab3 (Graph Coverage test case design).xlsx

# 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.

No.	Test method	Source test code
1	adjacentHash(Stri ng hash, Direction direction)	Pfest public void adjacentHash() {  try( GeeHash.adjacentHash() himb(", Direction.BOTTOM); } patch (ItlegalArgumentException e){     ossertEquata(e.getHessage(), which "adjacent has no meaning for a zero length hash that covers the whole world"); }  /**  **String str_adjhash = GeeHash.adjacentHosh(, hash: "20je", Direction.BOTTOM, Negro 1);  assertEquata((mounted "20ji", str_adjhash);  str_adjhash = GeeHash.adjacentHosh(hash: "20je", Direction.BOTTOM, Negro 0);  assertEquata((mounted "20ji", str_adjhash);
		<pre>str_adhash = Geokash.adjacontkash( habr '20jin', Birection.80TTOM, steps -1); assertEquals( expected, "20jin', str_adjhash);</pre>

```
fromLongToString
(long hash)

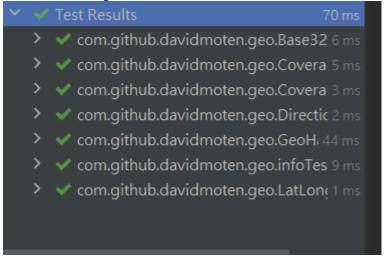
2

try{
    GeoHash.fromLongToString( hash 13);
}-catch (IllegalArgumentException e)
{
    assertEquals(e.getHessage(), actual "invalid long geohash 13");
}
} try{
    GeoHash.fromLongToString( hash 9);
}-catch (IllegalArgumentException e)
{
    assertEquals(e.getHessage(), actual "invalid long geohash 8");
}
String s = GeoHash.fromLongToString( hash 1);
    assertEquals(e.getHessage(), actual "invalid long geohash 8");
}
eTest
public void testEncodeHashToLong() {
    try{
        double latitude = 37.42199;
        double latitude = -122.084657;
        int length = 12;
        GeoHash.encodeHashToLong(latitude,lontitude,length);
}-catch (IllegalArgumentException e)
{
        assertEquals(e.getHessage(), actual "0x9c225c26a14d8al");
}

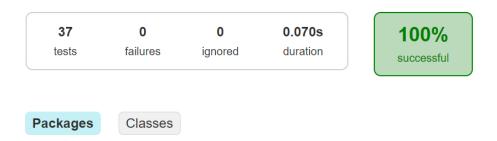
try{
        double latitude = 180.8;
        double lontitude = -122.084657;
        int length = 12'
        int leng
```

### 4 Test Results

4.1 JUnit test result snapshot

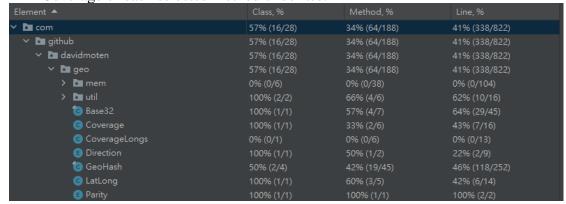


# **Test Summary**

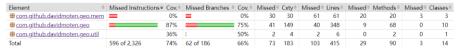


## 4.2 Code coverage snapshot

• Coverage of each selected method under test



Total coverage



# 4.3 CI result snapshot (3 iterations for CI)

• CI#1

README.md

pipeline passed coverage 73%
------------------------------

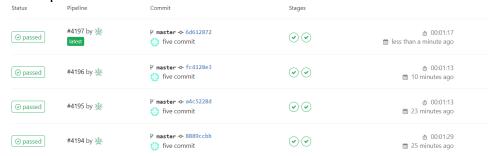
• CI#2

pipeline	passed	coverage	74%

• CI#3

pipeline passed coverage 76%

## • 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 100% to 100% in Lab3.

		Lab1 (or Lab2)		Lab3	
No.	Test method	statement coverage	branch coverage	statement coverage	branch coverage
1	encodeBase32(St ring hash)	100%	100%	100%	100%
2	getCharIndex(cha r ch)	100%	100%	100%	100%
3	adjacentHash(Stri ng hash, Direction direction)	100%	100%	100%	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 100% and 100%, respectively. Thus, the test requirements described in Section 1 are satisfied.

Lab3 The Coverage Comparison 中三次Lab的statement coverage 和branch coverage 都是 100%,但是在test case上皆比Lab01和Lab02來的少。