Lab 1 Report

Name
Student ID
Date

1 Test Plan

1.1 Test requirements

The Lab 1 requires to (1) select **15 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 40%*".

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	Environment Setting	1	17th, March
2	Study GeoProject	3	17th, March
3	Learn JUnit	1	17th, March
4	Design test cases for the selected methods	2	18th, March
5	Implement Base32 test cases	1	18th, March
6	Perform Base32 test	1	18th, March

7	Implement Coverage test cases	1	19th, March
8	Perform Coverage test	1	19th, March
9	Implement CoverageLong test cases	1	21th, March
10	Perform CoverageLong test	1	21th, March
11	Implement Direction test cases	0.5	22th, March
12	Perform Direction test	0.5	22th, March
13	Implement GeoHash test cases	1	23th, March
14	Perform GeoHash test	1	23th, March
15	Implement Info test cases		24th, March
16	Perform Info test	1	24th, March
17	Complete Lab1 report	5	24th-25th, March

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 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	A. encodeBase32(long i, int length) B. encodeBase32(long i)	Returns the base 32 encoding of the given length from a geohash	A. Base32.*encodeBase32*(75324, 4) //positive Base32.*encodeBase32*(-122,4) //negative B. Base32.*encodeBase32*((long) 32.0);	A. 29jw -003u B. 000000000 010
		C. decodeBase32(String hash)	Returns the conversion of a base32 geohash to a long	C. Base32.decodeBase32("w") //positive Base32.*decodeBase32*("-j") //negative	C. 28 -17
		D. getCharIndex(char ch)	Throws an IllegalArgument Exception if the character is not found in the array.	D. Base32.getCharIndex('-')	D. Throw message: "not a base32 character : -"

2	Coverage	A. Coverage(Set<String> hashes, double ratio)
		B. Coverage(CoverageLongs coverage)
			How well the coverage is covered by the hashes. Will be >=1. Closer to 1, the close the coverage is to the region in question.	getHashSets.add("22");
				getHashSets.add("33");
				getHashSets.add("44");
				A. Coverage(getHashSets,2.5) //Coverage Test
				---------- 	
				long[] hashes = new long[]{22,33,44}
				B. CoverageLongs coverageLongs = new CoverageLongs(hashes,3,2.5);
				Coverage coverage_long = new Coverage(coverageLongs); //CoverageLongs Test
					A. Hashes = getHashSets
					Ratio = 2.5
					B. getHashLength = 1
					Ratio = 2.5

		C. getHashes()	Returns the hashes which are expected to be all of the same length.	getHashSets.add("aaa") getHashSets.add("bbb") Coverage getHashCoverage = new Coverage(getHashSets,5.5) C. getHashCoverage.getHashes()	C. getHashSets
		D. getRatio()	Returns the measure of how well the hashes cover a region.	Coverage(getHashSets,0.005) D. getRatioCoverage.getRatio()	D. Ratio = 0.005

		E. getHashLength()	Returns the length in characters of the first hash returned by an iterator on the hash set.	Coverage getHashLengthCoverage = new Coverage(getHashSets,2.4)
				E. getHashLengthCoverage.getHashLength() //hashes.size() == 0
				---------- 	
				getHashSets.add("something")
				getHashLengthCoverage.getHashLength() //hashes.size() != 0
					E. HashLength():
					➢ When hash size = 0: 0
					➢ When hash size ≠ 0: 9

		F. toString()	Show Coverage [hashes, ratio]	Coverage testToStringCoverage = new Coverage(getHashSets,6.66)	F. "Coverage [hashes=[

				F. testToStringCoverage .toString()	], ratio=6.66]”
3	CoverageLongs	A. getHashes()	Returns the hashes which are expected to be all of the same length	hash = new long[]{33,22}; CoverageLongs coverageLongsGetHashes = new CoverageLongs(hash, 2, 4.5); A. coverageLongsGetHashes.getHashes()	A. hash = long[]{33,22}
		B. getRatio()	Returns the measure of how well the hashes cover a region	B. coverageLongsGetRatio.getRatio()	B. 3.0
		C. getHashLength()	Returns the length in characters of the first hash returned by an iterator on the hash set //compare[0] with 0000 1111(0x0f)	long[] hashZero = new long[]{}; CoverageLongs coverageLongsGetZeroLength = new CoverageLongs(hashZero,0,3.3); C. coverageLongsGetZeroLength.getHashLength() //length = 0 ---------- hash = new long[]{33,22}; CoverageLongs coverageLongsGetHashLength = new CoverageLongs(hash,count,2.2); coverageLongsGetHashLength.getHashLength() // //length !=0	C. Hash Length: ➢ When length = 0: 0 ➢ When length ≠ 0: 1
		D. getCount()	Count hash element	hash = new long[]{33,22}; CoverageLongs coverageLongsGetCount = new CoverageLongs(hash, count,3.00); D. coverageLongsGetCount.getCount()	D. 2

4	GeoHash	A. adjacentHash (String hash, Direction direction, int steps) Method: adjacentHash( )
		B. adjacentHash(String hash, Direction direction) Method: testAdjacentHash()
			Returns the adjacent hash N steps in the given Direction. A negative N will use the opposite Direction
			--- 	
			Returns the adjacent hash in given Direction
				hash = "11w";
				direction = Direction.TOP;
				A. adjacentHash = geoHash.adjacentHash (hash, direction,-1); //steps<0
				adjacentHash = geoHash.adjacentHash (hash, direction, 5); //steps>0
				B. adjacentHash = geoHash.adjacentHash (hash, direction); //normal
				--- 	
				➢ adjacentHash = geoHash.adjacentHash("zzz", Direction.TOP); //out of border(top)
				➢ adjacentHash = geoHash.adjacentHash("145",Direction.BOTTOM); //out of border(bottom)
				➢ adjacentHash = geoHash.adjacentHash("000",Direction.LEFT); //out of border(left)
				➢ adjacentHash = geoHash.adjacentHash("ppp",Direction.RIGHT); //out of border(right)
					A. geoHash.adjacentHash("11w", Direction.BOTTOM, 1) "14y"
					B. normal: "11y"
					--- 	
					out of border:
					➢ top: "gzz"
					➢ bottom: "11g"
					➢ left: "pbp"
					➢ right: "pr0"

		C. right (String hash)	Returns the adjacent hash to the right (east)	hash = "11w"; C. geoHash.right(hash)	C. "11x"

		D. left (String hash)	Returns the adjacent hash to the left (west)	hash = "11w"; D. geoHash.left(hash)	D. “11t”
		E. top (String hash)	Returns the adjacent hash to the top (north)	hash = "11w"; E. geoHash.top(hash))	E. “11y”
		F. bottom (String hash)	Returns the adjacent hash to the bottom (south)	hash = "11w"; F. geoHash.bottom(hash)	F. “11q”

		G. neighbours() Method: neighbours(String hash)	Returns a list of the 8 surrounding hashes for a given hash in order: left,right,top, bottom,left-top,left-bottom,right-top,right-bottom.	List<String> neighbours = new ArrayList<String>();
				neighbours = geoHash.neighbours(hash);
				List<String> compare = new ArrayList<String>();
				compare.add("11t"); //left
				compare.add("11x"); //right
				compare.add("11y"); //top
				compare.add("11q"); //bottom
				compare.add("11v"); //left-top
				compare.add("11m"); //left-bottom
				compare.add("11z"); //right-top
				compare.add("11r"); //right-bottom
				G. neighbours
					G. compare

5	Direction	A. opposite()	Returns the opposite direction	Direction bottom = Direction.BOTTOM; Direction top = Direction.TOP; Direction left = Direction.LEFT; Direction right = Direction.RIGHT; A. bottom.opposite() top.opposite() left.opposite() right.opposite()	A. Opposite of ➢ BOTTOM: TOP ➢ TOP: BOTTOM ➢ LEFT: RIGHT ➢ RIGHT: LEFT
6	Info	A. id()	id of Info	String a = "a"; Optional<String> id = Optional.of(a); Info info = new Info(25.5, 30.0, 10000, 555, id); A. info.id()	A. Optional.of("a")

		B. lat()	Latitude of Info	String a = "a"; Optional<String> id = Optional.of(a); Info info = new Info(25.5, 30.0, 10000, 555, id); B. info.lat()	B. 25.5
		C. lon()	Longitude of Info	String a = "a"; Optional<String> id = Optional.of(a); Info info = new Info(25.5, 30.0, 10000, 555, id); C. info.lon()	C. 30.0
		D. time()	Time of Info	String a = "a"; Optional<String> id = Optional.of(a); Info info = new Info(25.5, 30.0, 10000, 555, id); D. info.time()	D. 10000
		E. value()	Value of Info	String a = "a"; Optional<String> id = Optional.of(a); Info info = new Info(25.5, 30.0, 10000, 555, id); E. info.value()	E. 555
		F. toString()	Show all information of Info	String a = "a"; Optional<String> id = Optional.of(a); Info info = new Info(25.5, 30.0, 10000, 555, id); F. info.toString()	F. “Info [lat=25.5, lon=30.0, time=10000, value=555, id=Optional .of(a)]”

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()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java
2	decodeBase32()	
3	getCharIndexException()	
4	testCoverage()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/CoverageTest.java
5	getHashes()	
6	getRatio()	
7	getHashLength()	
8	testToString()	
9	getHashes()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/CoverageLongsTest.java
10	getRatio()	
11	getHashLength()	
12	testToString()	
13	getCount()	
14	opposite()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/DirectionTest.java
15	adjacentHash()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java
16	right()	
17	left()	
18	top()	
19	bottom()	
20	testAdjacentHash()	
21	neighbours()	
22	id()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
23	lat()	
24	lon()	
25	time()	
26	value()	
27	testToString()	

4 Test Results

4.1 JUnit test result snapshot

Test Summary

51	0	0	0.225s
tests	failures	ignored	duration

100% successful

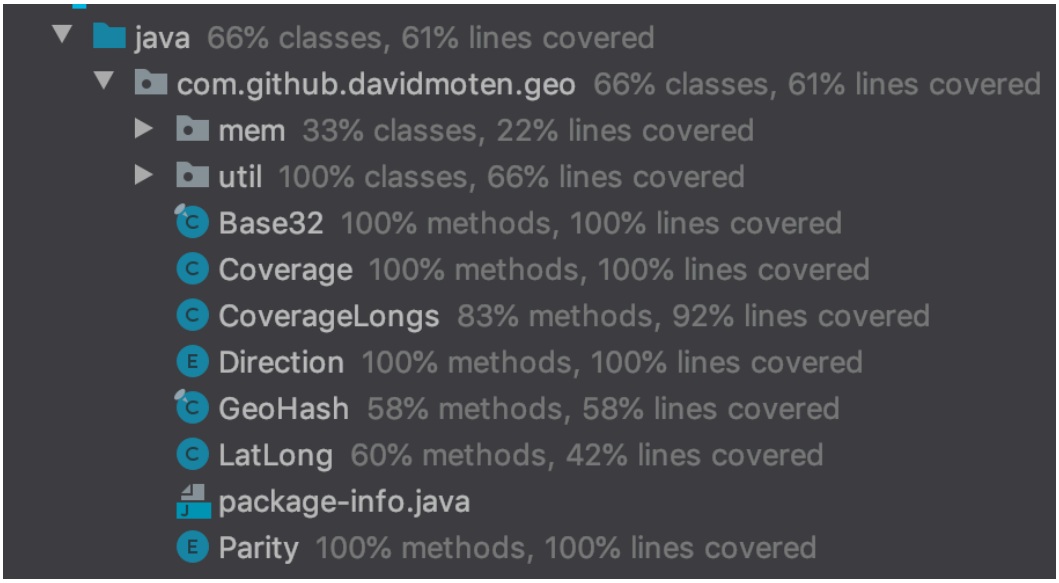
Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	39	0	0	0.168s	100%
com.github.davidmoten.geo.mem	12	0	0	0.057s	100%

geo/build/reports/tests/test/index.html

4.2 Code coverage snapshot

- Coverage of each selected method



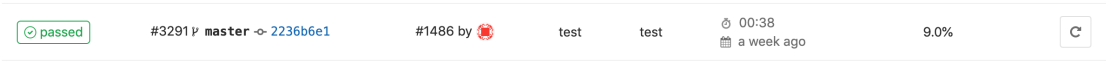
- Total coverage

geo

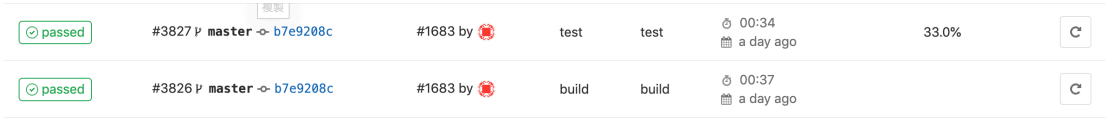
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		68%		55%	62	149	114	348	21	68	2	10
com.github.davidmoten.geo.mem		19%		0%	23	30	48	61	13	20	2	3
com.github.davidmoten.geo.util		36%		50%	2	4	2	6	0	2	0	1
Total	903 of 2,326	61%	94 of 186	49%	87	183	164	415	34	90	4	14

4.3 CI result snapshot (3 iterations for CI)

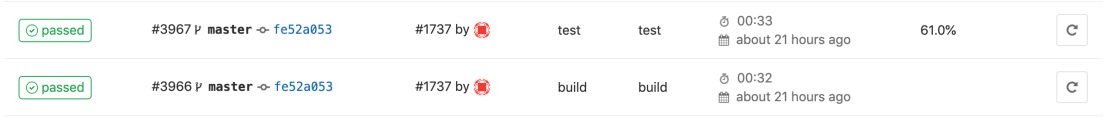
- CI#1



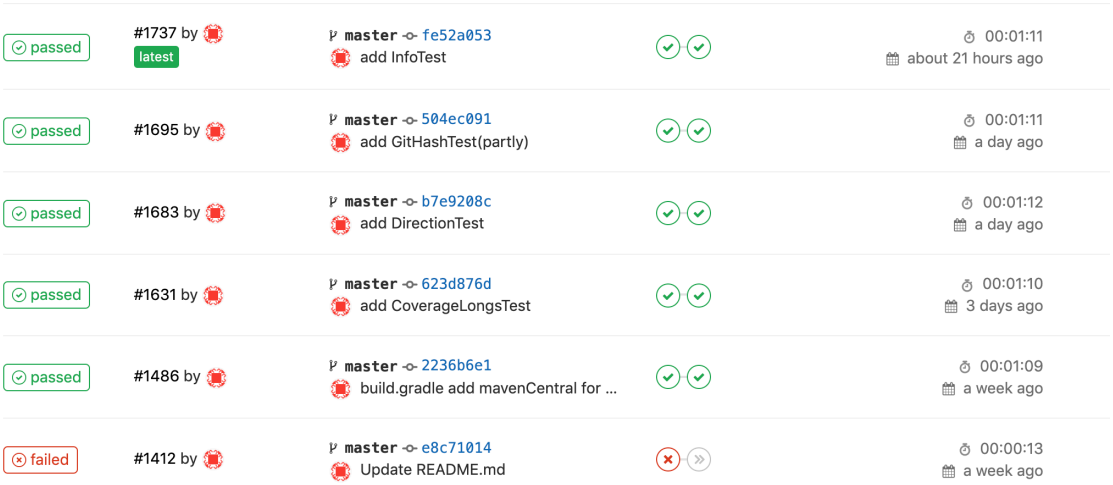
- CI#2



- CI#3



- CI Pipeline



5 Summary

In Lab 1, **27 test cases have been designed and implemented using JUnit**. The test is conducted in 4 CI and **the execution results of the 15 test methods are all passed**. **The total statement coverage of the test is 61%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are …I learnt Junit and GeoHash algorithm in this lab.