Lab 2 Report

Name
Student ID
Date April 22, 2020

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, and (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	2	14th, April
2	Learn **ISP** and JUnit	2	15th, April
3	Design test cases for the selected methods	3	17th, April
4	Implement Base32 test	1	19th, April

	cases		
5	Perform Base32 test	1	19th, April
10	Implement Geomem test cases	3	21th, April
11	Perform Geomem test	3	21th, April
12	Implement GeoHash test cases	1	22th, April
13	Perform GeoHash test	2	22th, April
14	Implement Info test cases	0.5	22th, April
15	Perform Info test	0.5	22th, April
16	Complete Lab2 report	2	22th, April

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	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. 0000000010

		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 IllegalArgumentException if the character is not found in the array.	D. Base32.getCharIndex('-')	D. Throw message: "not a base32 character : -"

2	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("
					A. geoHash.adjacentHash("11w", Direction.BOTTOM, 1) “14y”
					B. normal: “11y”
					------- 	
					out of border:
					➢ top: “gzz”
					➢ bottom: ”11g”
					➢ left: “pbp”
					➢ right: “pr0”

				000",Direction.LEFT);
				//out of border(left)
				➢ adjacentHash = geoHash.adjacentHash("ppp",Direction.RIGHT);
				//out of border(right)	

		C. encodeHash(double latitude, double longitude, int length)	return hash of adjacent hash	C. T1:{latitude=-50,longitude=0,length=5,expect:hp058} T2:{latitude=-50,longitude=-100,length=0,expect:throw:"length must be greater than zero"} T3:{latitude=-100,longitude=500,length=5,expect:throw:"latitude must be between -90 and 90 inclusive"} T4:{latitude=-100,latitude=45,length=0,expect:throw:"length	C. hp058 throw:"length must be greater than zero" throw:"latitude must be between -90 and 90 inclusive"

				must be greater than zero"} T5:{latitude=120,latitude=45,length=3,expect:throw:"latitude must be between -90 and 90 inclusive"} T6:{latitude=120,latitude=45,length=-5,expect:throw:"length must be greater than zero"}	
3	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)]”
4	Geomem	G. find(double topLeftLat, double topLeftLon, double bottomRightLat,double bottomRightLon, long start, long finish)	return info records	T1:{topLeftLat=-1,topLeftLong=100,bottomRightLat=-90,bottomRightLong=170,PRECISION=0.00001,expect:(size=1,lat=-90,lon=150,time=500L,value="X",id="1")} G. find(double topLeftLat, double topLeftLon, double bottomRightLat,double bottomRightLon, long start, long finish)	G. T
		H. createRegionFilter(final double topLeftLat, final double topLeftLon, final double bottomRightLat,	predicate	H. T1:{topLeftLat=-2,topLeftLong=101,assert:True} I. T2:{topLeftLat=0,topLeftLong=101,assert:False} J. T3:{topLeftLat=-2,topLeftLong=99,assert:False} K. T4:{topLeftLat=-1,bottomRightLong=171,assert:False} L. T5:{bottomRightLat= -91, bottomRightLong= 169, assert:False }	H. T,F

		final double bottomRightLon)			

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).

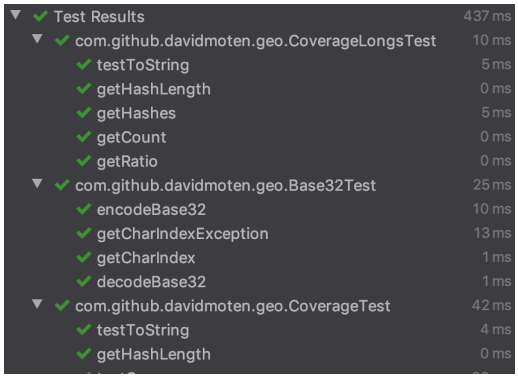
No.	Test method	Source code
1	Optional<R> id()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
2	double lat()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
3	double lon()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
4	long time()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
5	T value()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
6	toString()	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/InfoTest.java
7	encodeBase32(long i, int length)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java
8	encodeBase32(long i)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java
9	decodeBase32(String hash)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java
10	getCharIndex(char ch)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/Base32Test.java

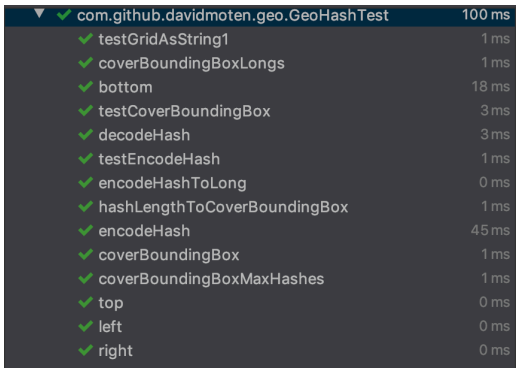
11	adjacentHash(String hash, Direction direction)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java
12	adjacentHash(String hash, Direction direction, int steps)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java
13	encodeHash(double latitude, double longitude, int length)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/GeoHashTest.java
14	find(double topLeftLat, double topLeftLon, double bottomRightLat,double bottomRightLon, long start, long finish)	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/GeomemTest.java
15	createRegionFilter(final double topLeftLat, final double topLeftLon, final double bottomRightLat, final double	https://stv.csie.ntut.edu.tw/rojeanlin/GeoProject/blob/master/src/test/java/com/github/davidmoten/geo/mem/GeomemTest.java

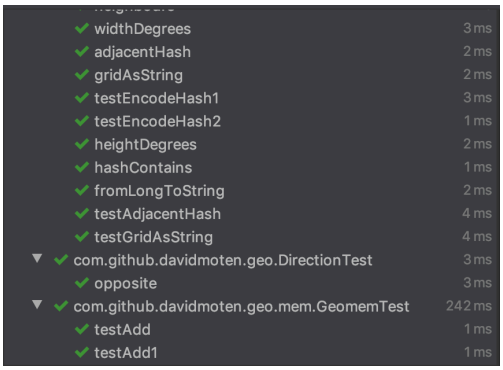
	bottomRightLon)	

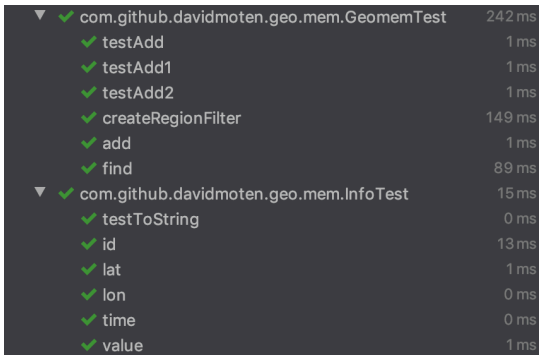
4 Test Results

4.1 JUnit test result snapshot









Test Summary

52	0	0	0.200s
tests	failures	ignored	duration

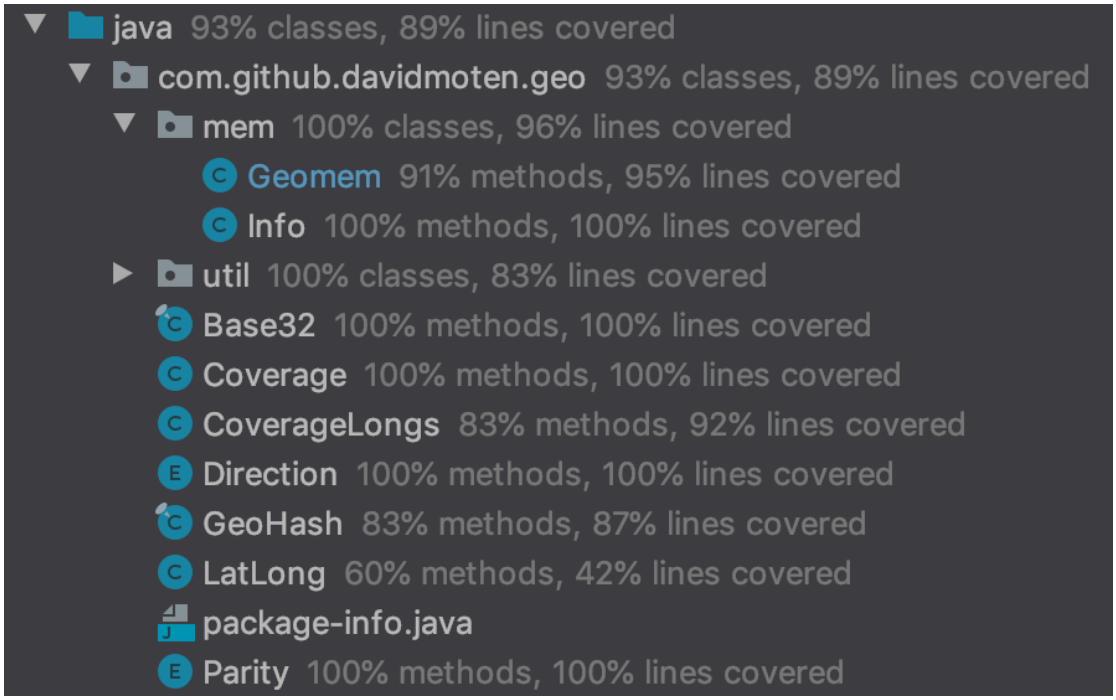
100% successful

Packages | Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	40	0	0	0.060s	100%
com.github.davidmoten.geo.mem	12	0	0	0.140s	100%

4.2 Code coverage snapshot

- Coverage of each selected method

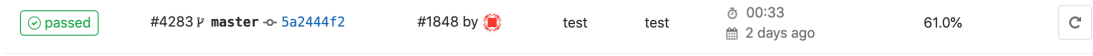


- Total coverage

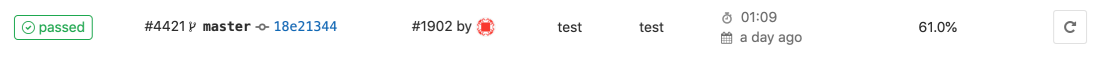
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		87%		82%	33	149	41	348	10	68	0	10
com.github.davidmoten.geo.mem		97%		85%	4	30	2	61	1	20	0	3
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	263 of 2,326	88%	33 of 186	82%	38	183	44	415	11	90	0	14

4.3 CI result snapshot (3 iterations for CI)

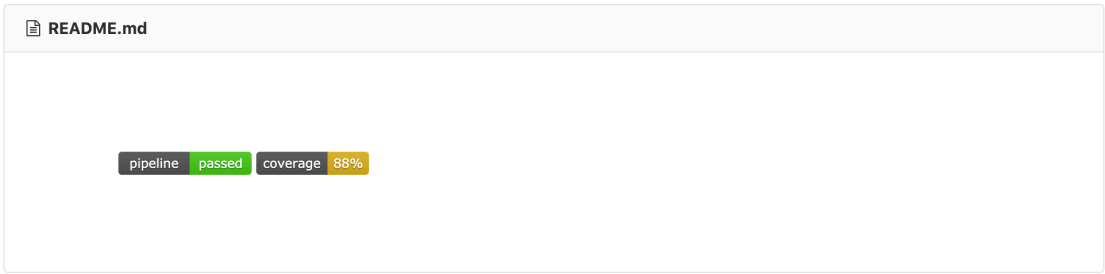
- CI#1



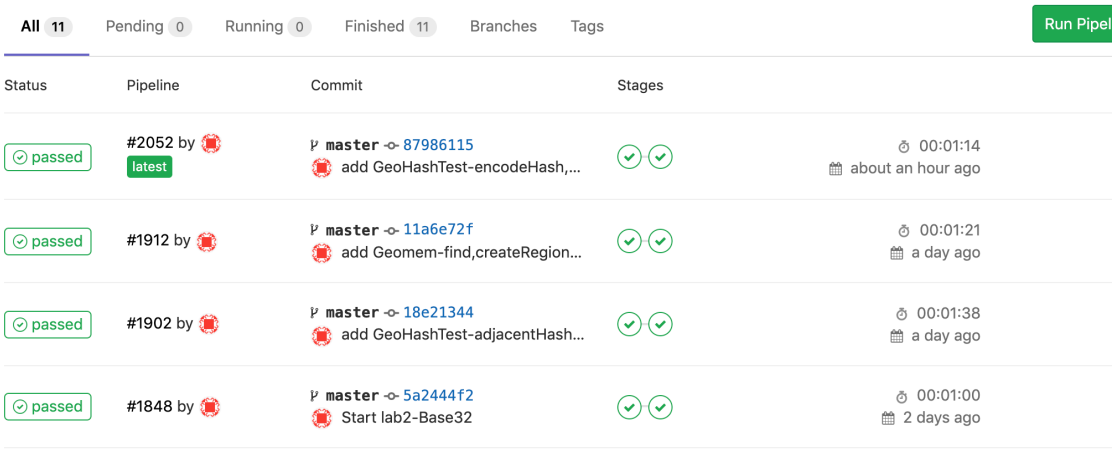
- CI#2



- CI#3



- CI Pipeline



5 Summary

In Lab 2, **15 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 15 test methods are all passed**. **The total statement coverage of the test is 88%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are …