Lab 2 Report

Name: 許哲維
Student ID: 111598066
Date: 2023.03.29

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 73% (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 12 methods that were chosen in Lab1** and **3 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	3	2023.03.16
2	Learn **ISP** and JUnit	5	2023.03.19
3	Design test cases for the selected methods	5	2023.03.22
4	Implement test cases	6	2023.03.25
5	Perform tests	6	2023.03.27
6	Complete Lab2 report	2	2023.03.29

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 85% of all test cases must pass) and *the statement coverage should have achieved at least 73%*.

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(long i, int length)		{75324,4}, {-75324,4}, {75324,8}, {-75324,8}	{"29jw"}, {"-29jw"}, {"000029jw"}, {"-000029jw"}
2	Base32	encodeBase32(long i)		{75324}, {-75324}	{"0000000029jw"}, {"-0000000029jw"}
3	Base32	decodeBase32(String		{"29jw"}, {"-29jw"},	{75324}, {-75324},

		hash)		{""}	{0}
4	Base32	getCharIndex(char ch)		{‘0’}, {‘A’}	{0}, {"not a base32 character: 0"}
5	LatLong	add(double deltaLat, double deltaLon)		{0,0}, {1,0}, {0,1}, {-1,0}, {0,-1}, {1,1}, {1,-1}, {-1,1}, {-1,-1}	{lat,lon}, {lat+1,lon}, {lat,lon+1}, {lat-1,lon}, {lat,lon-1}, {lat+1,lon+1}, {lat+1,lon-1}, {lat-1,lon+1}, {lat-1,lon-1}
6	LatLong	toString()		{0,0}, {1,0}, {0,1}, {-1,0}, {0,-1}, {1,1}, {1,-1}, {-1,1}, {-1,-1}	{"LatLong [lat=0, lon=0]"}, {"LatLong [lat=1, lon=0]"}, {"LatLong [lat=0, lon=1]"}, {"LatLong [lat=-1, lon=0]"}, {"LatLong [lat=0, lon=-1]"}, {"LatLong [lat=1, lon=1]"}, {"LatLong [lat=1, lon=-1]"}, {"LatLong [lat=-1, lon=1]"}, {"LatLong [lat=-1, lon=-1]"}
7	LatLong	getLat()		{0}, {1}, {-1}	{0.00}, {1.00}, {-1.00}
8	LatLong	getLon()		{0}, {1}, {-1}	{0.00}, {1.00}, {-1.00}
9	Direction	opposite()		{Direction.TOP}, {Direction.BOTTOM}, {Direction.LEFT}, {Direction.RIGHT}	{Direction.BOTTOM}, {Direction.TOP}, {Direction.RIGHT}, {Direction.LEFT}

10	GeoHash	hashLength ToCoverBou ndingBox(d ouble topLeftLat, double topLeftLon, double bottomRigh tLat, double bottomRigh tLon)		{90,180,90, 180}	{12}
11	GeoHash	neighbours(String hash)		{"29jw"}, {"-29jw"}, {""}	{75324}, {-75324}, {"adjacent has no meaning for a zero length hash that covers the whole world"}
12	GeoHash	adjacentHas h(String hash, Direction direction)		{"29jw",Dir ection.BOT TOM,1}	{“29jt”}
13	GeoHash	right(String hash)		{"29jw"}, {"-29jw"}, {""}	{"29jy"}, {"-29jx"}, {"adjacent has no meaning for a zero length hash that covers the whole world"}
14	GeoHash	left(String hash)		{"29jw"}, {"-29jw"}, {""}	{"29jq"}, {"-29jt"}, {"adjacent has no meaning for a zero length hash that covers the whole world"}
15	GeoHash	top(String hash)		{"29jw"}, {"-29jw"}, {""}	{"29jx"}, {"-29jy"}, {"adjacent has no meaning for a zero length hash that covers the whole world"}

16	GeoHash	bottom(String hash)		{"29jw"}, {"-29jw"}, {""}	{"29jt"}, {"-29jq"}, {"adjacent has no meaning for a zero length hash that covers the whole world"}

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	encodeBase32(long i, int length)、encodeBase32(long i)	no usages 許晉維
		@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: 75324, length: 8);
		assertEquals(expected: "000029jw", encode);
		encode = Base32.encodeBase32(i: -75324, length: 8);
		assertEquals(expected: "-000029jw", encode);
		encode = Base32.encodeBase32(i: 75324);
		assertEquals(expected: "0000000029jw", encode);
		encode = Base32.encodeBase32(i: -75324);
		assertEquals(expected: "-0000000029jw", encode);
		}

2	decodeBase32(String hash)	no usages 許晉維
		@Test
		public void decodeBase32() throws Exception {
		long decode = Base32.decodeBase32(hash: "29jw");
		assertEquals(expected: 75324, decode);
		decode = Base32.decodeBase32(hash: "-29jw");
		assertEquals(expected: -75324, decode);
		decode = Base32.decodeBase32(hash: "");
		assertEquals(expected: 0, decode);
		}

3	getCharIndex(char ch)	no usages 許哲維 +1
		@Test(expected = IllegalArgumentException.class)
		public void getCharIndex() throws Exception {
		assertEquals(expected: 0,Base32.getCharIndex('0'));
		Base32.getCharIndex('A');
		}

4	opposite()	no usages 許哲維
		@Test
		public void opposite() throws Exception {
		direction = direction.opposite();
		assertEquals(expected, direction);
		}

5	add(double deltaLat, double deltaLon)	no usages 許哲維
		@Test
		public void add() throws Exception {
		latlong = latlong.add(lon,lat);
		assertEquals(expected: lat+lon,latlong.getLat(), delta: 0.001);
		assertEquals(expected: lon+lat,latlong.getLon(), delta: 0.001);
		}

4 Test Results

4.1 JUnit test result snapshot

Test Summary

84	0	0	0.043s
tests	failures	ignored	duration

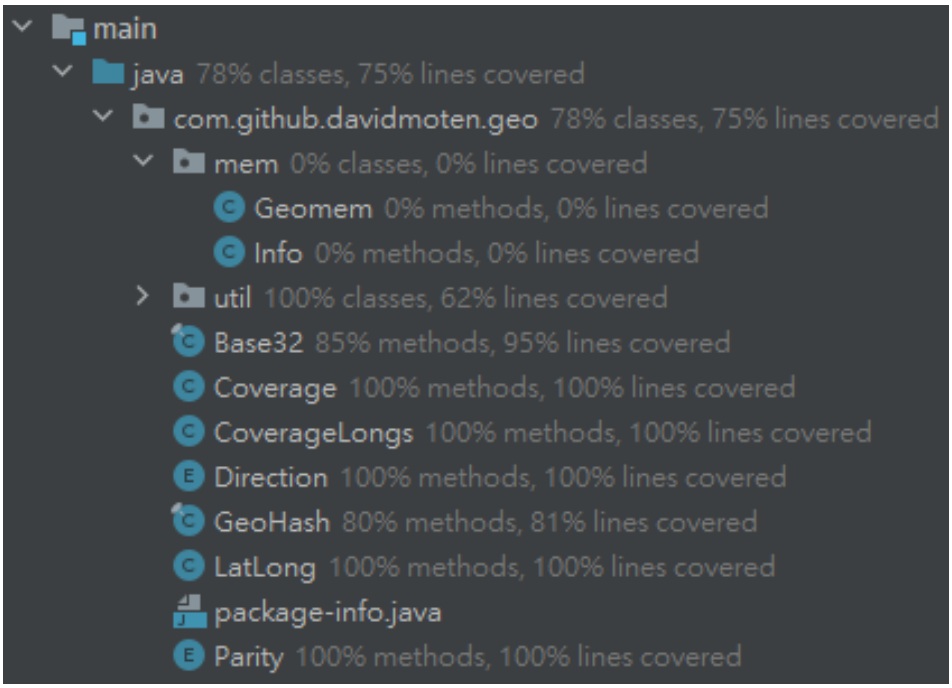
100% successful

Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	84	0	0	0.043s	100%

4.2 Code coverage snapshot

- Coverage of each selected method



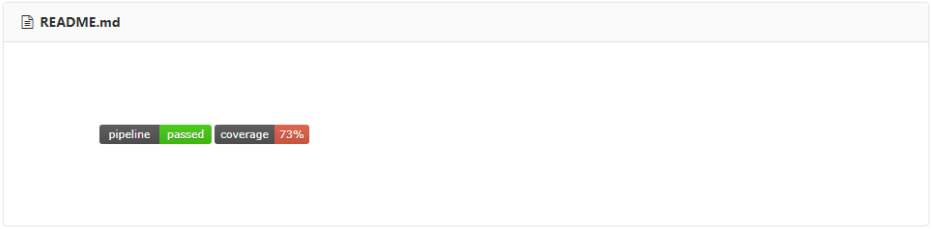
- Total coverage

geo

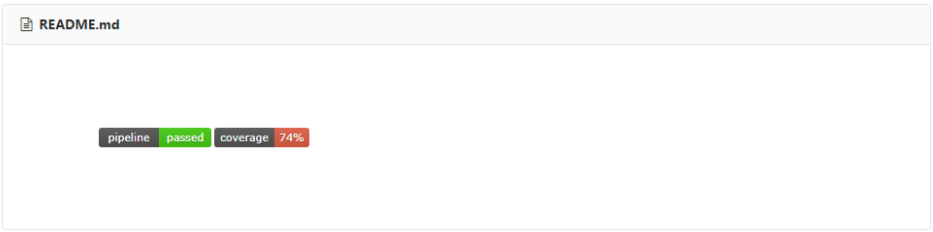
Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo		90%		80%	33	149	31	348	5	68	0	10
com.github.davidmoten.geo.mem		0%		0%	30	30	61	61	20	20	3	3
com.github.davidmoten.geo.util		68%		75%	1	4	1	6	0	2	0	1
Total	523 of 2,326	77%	53 of 186	71%	64	183	93	415	25	90	3	14

4.3 CI result snapshot (3 iterations for CI)

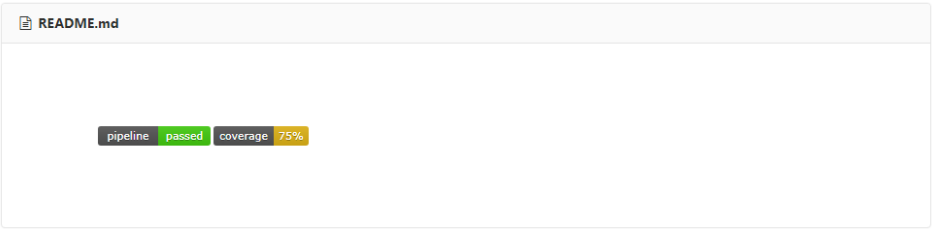
- CI#1



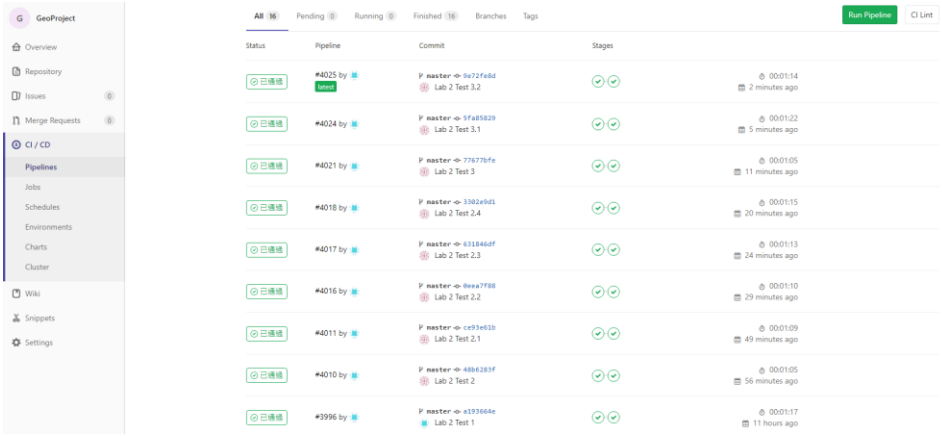
- CI#2



- 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 75%.** Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab is that through the ISP design we were able to avoid missing some test cases to achieve a higher coverage rate, as seen by the report we improved from 73% to 75%.