Lab 1 Report

Name 陳宇宏
Student ID 110598067
Date 2022/3/1

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	Study GeoProject	1	3/10
2	Learn JUnit	0.5	3/11
3	Design test cases for the selected methods	2	3/12
4	Implement test cases	2	3/12
5	Perform test	1	3/12
6	Complete Lab1 report	1	3/12

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 40%*.

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):String	Test if it returns the base 32 encoding of the given length from a Long geohash.	75324, 4	“29jw”
2	Base32	+encodeBase32(long i):String	Test if returns the base 32 encoding of length GeoHash.MAX_HASH_LENGTH from a Long geohash.	15324	“000000000fyw”
3	Base32	+decodeBase32(String): long	Test if it returns the conversion of a base32 geohash to a long.	"29jw"	75324
4	GeoHash	+encodeHash(double,double):String	Test if it returns a geohash of length MAX_HASH_LENGTH (12) for the given WGS84 point (latitude,longitude).	25.033821717782278, 121.56459135583758	"wsqqqm28s695"
5	GeoHash	+encodeHash(double,double,int):String	Test if it returns a geohash of given length for the given WGS84 point (latitude,longitude).	25.033821717782278, 121.56459135583758,8	“wsqqqm28”
6	GeoHas	+ encodeHash(LatLon	Test if it returns a	new LatLong(25.0338217177	"wsqqqm28s695"

	h	g):String	geohash of of length MAX_HASH_LENGTH (12) for the given WGS84 point.	82278, 121.56459135583758)	
7	GeoHash	+ encodeHash(LatLong,int):String	Test if it returns a geohash of given length for the given WGS84 point.	new LatLong(25.033821717782278, 121.56459135583758), 4	"wsqq"
8	GeoHash	+hashContains(String, double, double):boolean	Test if it returns true if and only if the bounding box corresponding to the hash contains the given lat and long.	"wsqqqm28s",25.0338077545166, 121.564621925354	true
				"wsqqqm28s",25.03385066986084, 121.564621925354	true
				"wsqqqm28s",25.03385066986084, 121.56457901000977	true
				"wsqqqm28s",25.0338077545166, 121.56457901000977	true
				"wsqqqm28s",25.0338077545165, 121.564621925354	false
				"wsqqqm28s",25.03385066986085, 121.564621925354	false
				"wsqqqm28s",25.03385066986084, 122.56457901000977	false
				"wsqqqm28s",25.0338077545166, 120.56457901000977	false
9	GeoHash	+adjacentHash (String, Direction) :String	Test if it returns the adjacent hash in given Direction.	"wsqqhp8hk1mn",Direction.TOP	"wsqqhp8hk1mp"
10	GeoHas	+adjacentHash (String,	Test if it returns the	"wsqqhp8hk1mn",Direction.TOP,3	“wsqqhp8hk1t1”

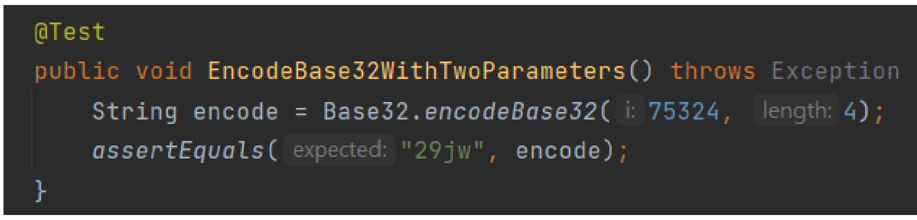
	h	Direction,int) :String	adjacent hash N steps in the given Direction.		
11	GeoHash	+right(String):String	Test if it returns the adjacent hash to the right (east).	"wsqqhjwyuvdu"	"wsqqhjwyuveh"
12	GeoHash	+bottom(String):String	Test if it returns the adjacent hash to the bottom (south).	"wsqqhjwyuvdu"	"wsqqhjwyuvdg"
13	GeoHash	+decodeHash(String):String	Test if it returns a latitude,longitude pair as the centre of the given geohash.	"wsqqhjwyuvdu"	new LatLong(24.99233888, 121.47432117)
14	GeoHash	+hashLengthToCoverBoundingBox(double, double, double, double):int	Test if it returns the maximum length of hash that covers the bounding box.	24.67693180092837, 121.73868842362948,22.09087140332405, 120.75488854089994	1
				36.633950257465244, 138.20176404334265,-1.2383111850902806, 103.60559538049397	0
				29.240501036359017, 107.95600702691077,29.230257104690256, 107.92737197784561	3
15	GeoHash	+neighbours(String hash): List<String>	Test if it 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.	"wsqqhjwyuvdu"	Arrays.asList("wsqqhjwyuvds", "wsqqhjwyuveh", "wsqqhjwyuvdv", "wsqqhjwyuvdg", "wsqqhjwyuvdt", "wsqqhjwyuvde", "wsqqhjwyuvej", "wsqqhjwyuve

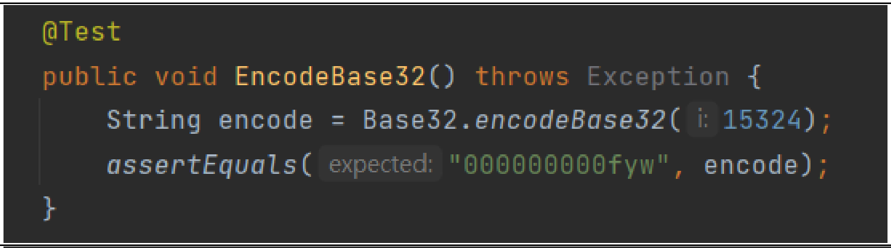
					5")

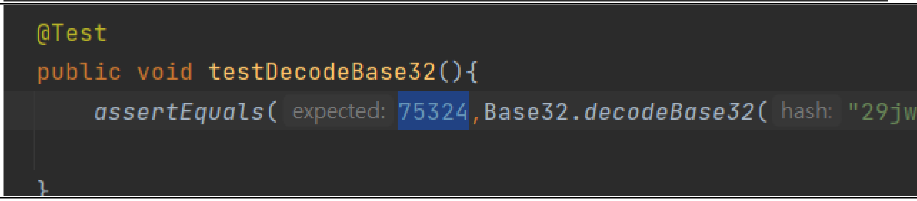
3 Test Implementation

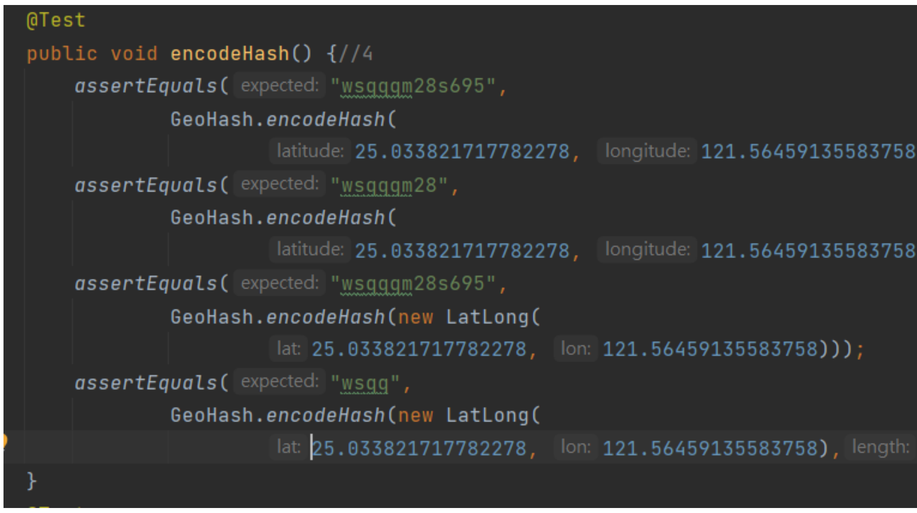
The design of test cases specified in Section 2 was implemented using JUnit 4. The test scripts of 3 selected test cases are given below. The rest of test script implementations can be found in the link (or JUnit files).

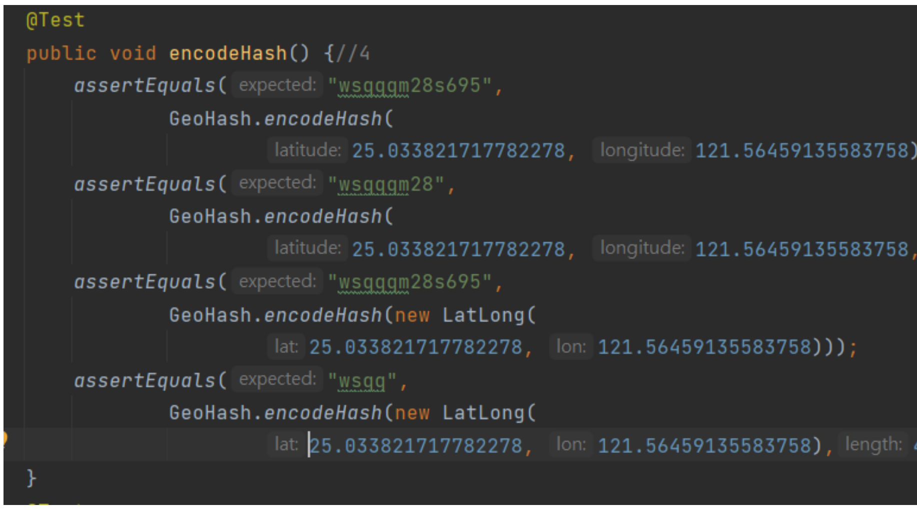
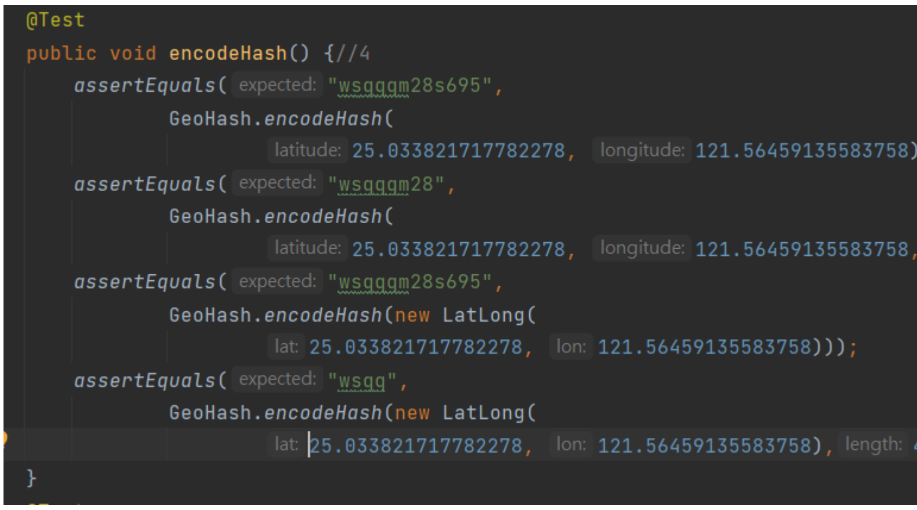
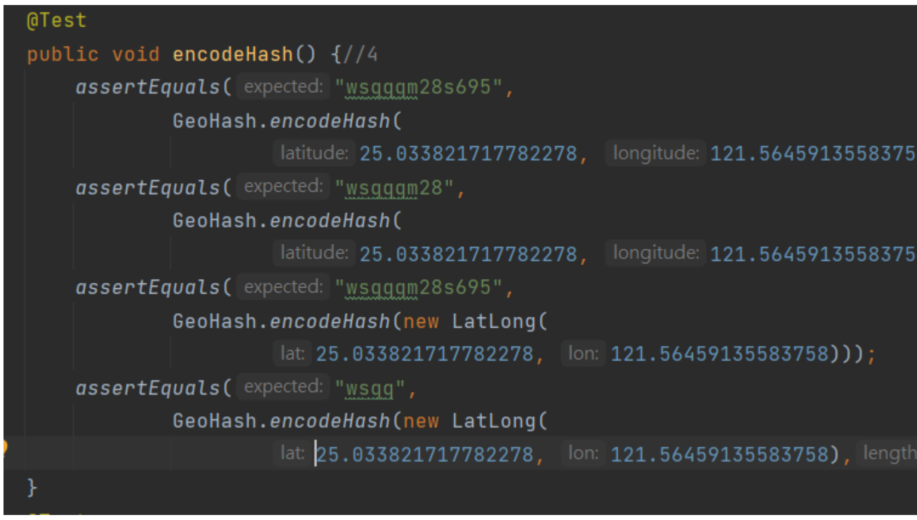
No.	Test method	Source code

1	+encodeBase32(long i,
	
	int length):String
		
		`@Test`
		`public void EncodeBase32WithTwoParameters() throws Exception`
		`    String encode = Base32.encodeBase32( i: 75324,  length: 4);`
		`    assertEquals( expected: "29jw", encode);`
		`}`

2	+encodeBase32(long i):String	
		`@Test`
		`public void EncodeBase32() throws Exception {`
		`    String encode = Base32.encodeBase32( i: 15324);`
		`    assertEquals( expected: "000000000fyw", encode);`
		`}`

3	+decodeBase32(String): long	
		`@Test`
		`public void testDecodeBase32(){`
		`    assertEquals( expected: 75324,Base32.decodeBase32( hash: "29jw`
		
		`}`

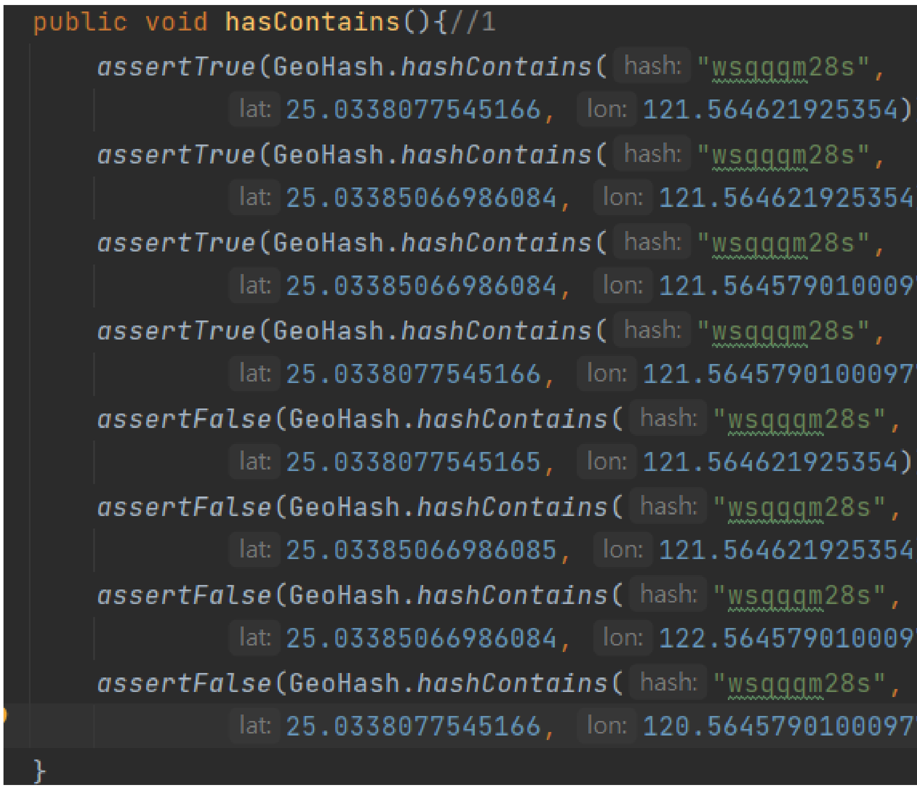
4	+encodeHash(double,double):String	
		`@Test`
		`public void encodeHash() {//4`
		`    assertEquals( expected: "wsqqqm28s695",`
		`            GeoHash.encodeHash(`
		`                    latitude: 25.033821717782278,  longitude: 121.56459135583758`
		`    assertEquals( expected: "wsqqqm28",`
		`            GeoHash.encodeHash(`
		`                    latitude: 25.033821717782278,  longitude: 121.56459135583758`
		`    assertEquals( expected: "wsqqqm28s695",`
		`            GeoHash.encodeHash(new LatLong(`
		`                    lat: 25.033821717782278,  lon: 121.56459135583758)));`
		`    assertEquals( expected: "wsqq",`
		`            GeoHash.encodeHash(new LatLong(`
		`                    lat: 25.033821717782278,  lon: 121.56459135583758), length:`
		`}`

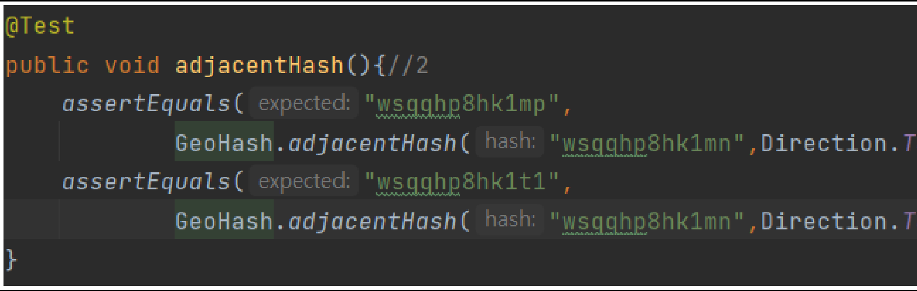
5	+encodeHash(double,double,int):String	
6	+ encodeHash(LatLong):String	
7	+ encodeHash(LatLong,int):String	

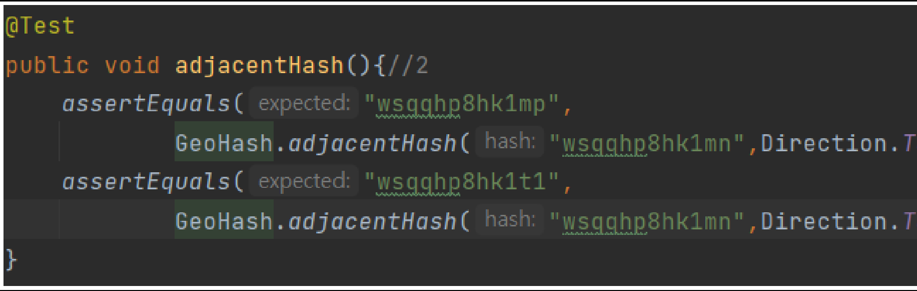
```
@Test
public void encodeHash() {//4
    assertEquals( expected: "wsqqgm28s695",
            GeoHash.encodeHash(
                    latitude: 25.033821717782278,  longitude: 121.56459135583758)
    assertEquals( expected: "wsqqgm28",
            GeoHash.encodeHash(
                    latitude: 25.033821717782278,  longitude: 121.56459135583758,
    assertEquals( expected: "wsqqgm28s695",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.033821717782278,  lon: 121.56459135583758)));
    assertEquals( expected: "wsqq",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.033821717782278,  lon: 121.56459135583758),  length:
}
```

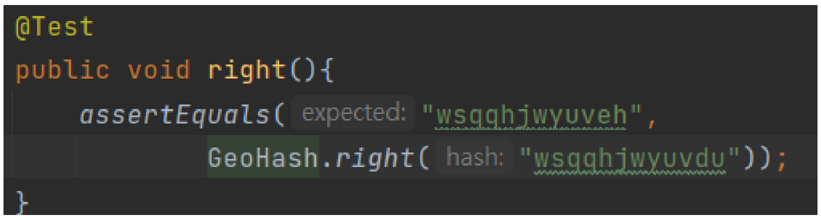
```
@Test
public void encodeHash() {//4
    assertEquals( expected: "wsqqgm28s695",
            GeoHash.encodeHash(
                    latitude: 25.033821717782278,  longitude: 121.56459135583758)
    assertEquals( expected: "wsqqgm28",
            GeoHash.encodeHash(
                    latitude: 25.033821717782278,  longitude: 121.56459135583758,
    assertEquals( expected: "wsqqgm28s695",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.033821717782278,  lon: 121.56459135583758)));
    assertEquals( expected: "wsqq",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.033821717782278,  lon: 121.56459135583758),  length:
}
```

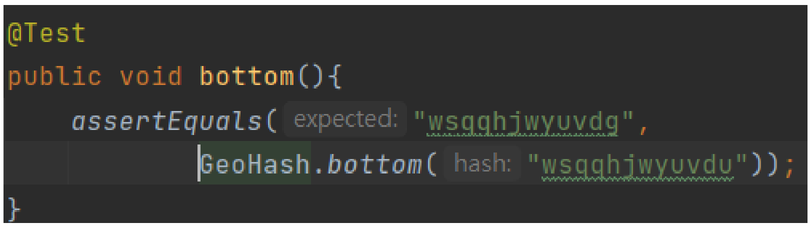
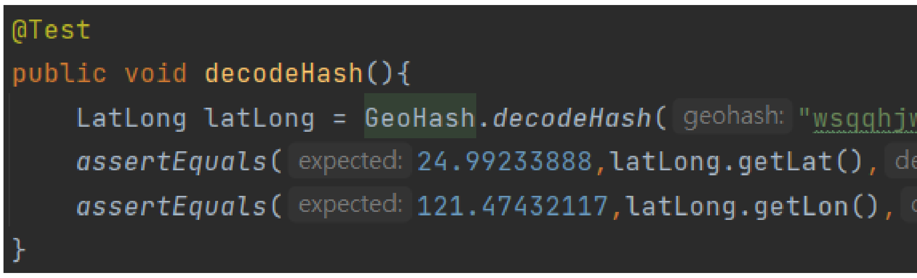
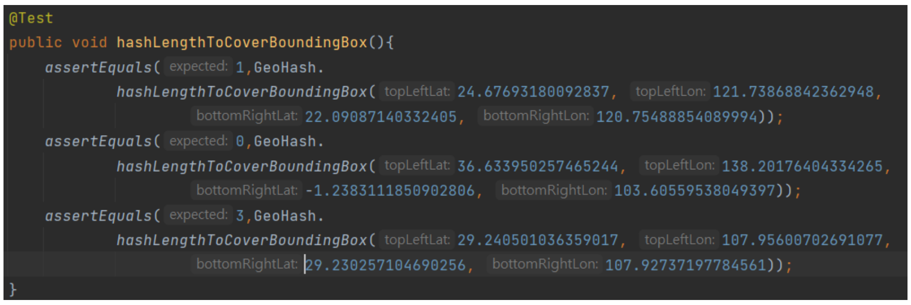
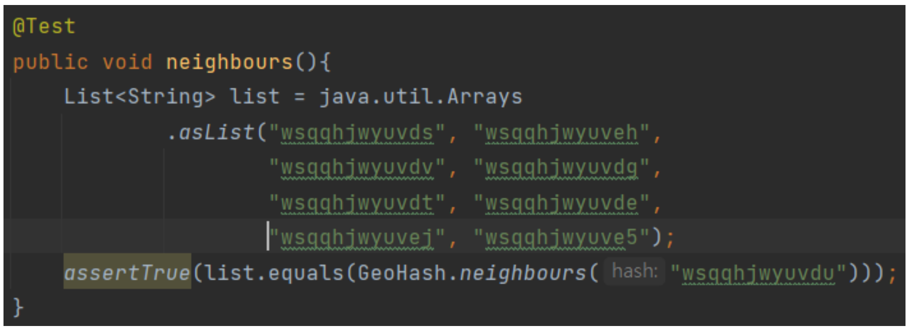
```
@Test
public void encodeHash() {//4
    assertEquals( expected: "wsqqgm28s695",
            GeoHash.encodeHash(
                    latitude: 25.033821717782278,  longitude: 121.5645913558375
    assertEquals( expected: "wsqqgm28",
            GeoHash.encodeHash(
                    latitude: 25.033821717782278,  longitude: 121.5645913558375
    assertEquals( expected: "wsqqgm28s695",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.033821717782278,  lon: 121.56459135583758)));
    assertEquals( expected: "wsqq",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.033821717782278,  lon: 121.56459135583758),  length
}
```

8	+hashContains(String, double, double):boolean	
		`public void hasContains(){//1`
		`assertTrue(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.0338077545166, lon: 121.564621925354)`
		`assertTrue(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.03385066986084, lon: 121.564621925354`
		`assertTrue(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.03385066986084, lon: 121.564579010009`
		`assertTrue(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.0338077545166, lon: 121.5645790100097`
		`assertFalse(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.0338077545165, lon: 121.564621925354)`
		`assertFalse(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.03385066986085, lon: 121.564621925354`
		`assertFalse(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.03385066986084, lon: 122.564579010009`
		`assertFalse(GeoHash.hashContains( hash: "wsqqqm28s",`
		`lat: 25.0338077545166, lon: 120.5645790100097`
		`}`

9	+adjacentHash (String, Direction) :String	
		`@Test`
		`public void adjacentHash(){//2`
		`assertEquals( expected: "wsqqhp8hk1mp",`
		`GeoHash.adjacentHash( hash: "wsqqhp8hk1mn",Direction.T`
		`assertEquals( expected: "wsqqhp8hk1t1",`
		`GeoHash.adjacentHash( hash: "wsqqhp8hk1mn",Direction.T`
		`}`

10	+adjacentHash (String, Direction,int) :String	
		`@Test`
		`public void adjacentHash(){//2`
		`assertEquals( expected: "wsqqhp8hk1mp",`
		`GeoHash.adjacentHash( hash: "wsqqhp8hk1mn",Direction.T`
		`assertEquals( expected: "wsqqhp8hk1t1",`
		`GeoHash.adjacentHash( hash: "wsqqhp8hk1mn",Direction.T`
		`}`

11	+right(String):String	
		`@Test`
		`public void right(){`
		`assertEquals( expected: "wsqqhjwyuveh",`
		`GeoHash.right( hash: "wsqqhjwyuvdu"));`
		`}`

12	+bottom(String):String	
13	+decodeHash(String):String	
14	+hashLengthToCoverBoundingBox(double, double, double, double):int	
15	+neighbours(String hash): List<String>	

```
@Test
public void bottom(){
    assertEquals( expected: "wsqghjwyuvdg",
            GeoHash.bottom( hash: "wsqghjwyuvdu"));
}
```

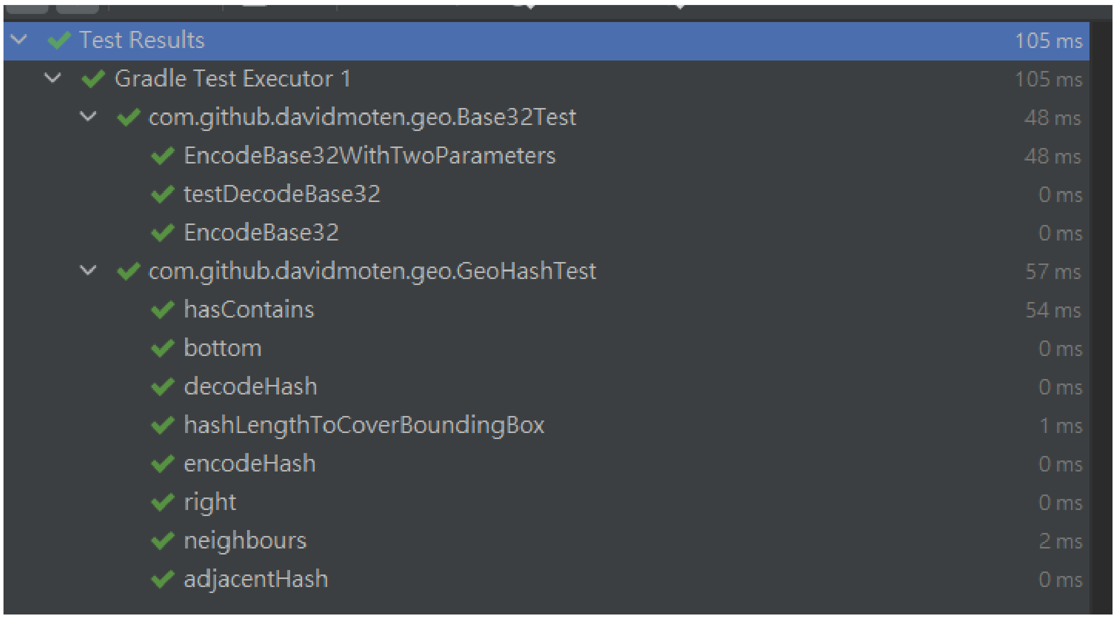
```
@Test
public void decodeHash(){
    LatLong latLong = GeoHash.decodeHash( geohash: "wsqghjw
    assertEquals( expected: 24.99233888,latLong.getLat(), de
    assertEquals( expected: 121.47432117,latLong.getLon(), 
}
```

```
@Test
public void hashLengthToCoverBoundingBox(){
    assertEquals( expected: 1,GeoHash.
            hashLengthToCoverBoundingBox( topLeftLat: 24.67693180092837, topLeftLon: 121.73868842362948,
                    bottomRightLat: 22.09087140332405, bottomRightLon: 120.75488854089994));
    assertEquals( expected: 0,GeoHash.
            hashLengthToCoverBoundingBox( topLeftLat: 36.633950257465244, topLeftLon: 138.20176404334265,
                    bottomRightLat: -1.2383111850902806, bottomRightLon: 103.60559538049397));
    assertEquals( expected: 3,GeoHash.
            hashLengthToCoverBoundingBox( topLeftLat: 29.240501036359017, topLeftLon: 107.95600702691077,
                    bottomRightLat: 29.230257104690256, bottomRightLon: 107.92737197784561));
}
```

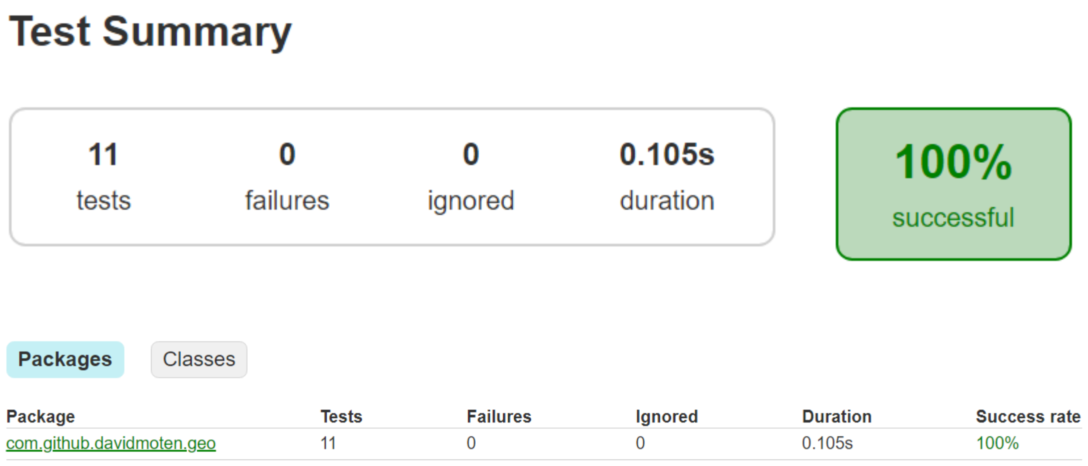
```
@Test
public void neighbours(){
    List<String> list = java.util.Arrays
            .asList("wsqghjwyuvds", "wsqghjwyuveh",
                    "wsqghjwyuvdv", "wsqghjwyuvdg",
                    "wsqghjwyuvdt", "wsqghjwyuvde",
                    "wsqghjwyuvej", "wsqghjwyuve5");
    assertTrue(list.equals(GeoHash.neighbours( hash: "wsqghjwyuvdu")));
}
```

4 Test Results

4.1 JUnit test result snapshot



```
Test Results                                   105 ms
  Gradle Test Executor 1                       105 ms
    com.github.davidmoten.geo.Base32Test        48 ms
      EncodeBase32WithTwoParameters             48 ms
      testDecodeBase32                           0 ms
      EncodeBase32                               0 ms
    com.github.davidmoten.geo.GeoHashTest       57 ms
      hasContains                               54 ms
      bottom                                     0 ms
      decodeHash                                 0 ms
      hashLengthToCoverBoundingBox               1 ms
      encodeHash                                 0 ms
      right                                      0 ms
      neighbours                                 2 ms
      adjacentHash                               0 ms
```



Test Summary

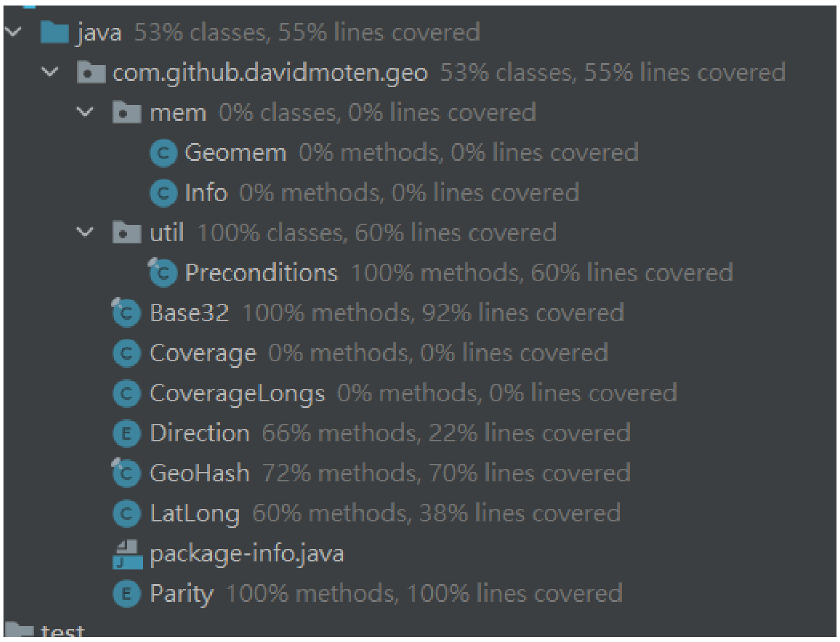
11	0	0	0.105s	100%
tests	failures	ignored	duration	successful

Packages Classes

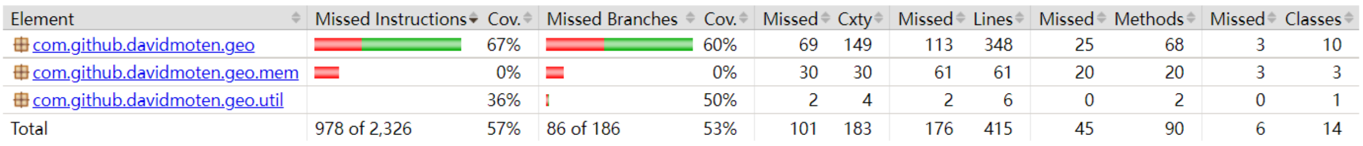
Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	11	0	0	0.105s	100%

4.2 Code coverage snapshot

- Coverage of each selected method


java 53% classes, 55% lines covered
com.github.davidmoten.geo 53% classes, 55% lines covered
mem 0% classes, 0% lines covered
Geomem 0% methods, 0% lines covered
Info 0% methods, 0% lines covered
util 100% classes, 60% lines covered
Preconditions 100% methods, 60% lines covered
Base32 100% methods, 92% lines covered
Coverage 0% methods, 0% lines covered
CoverageLongs 0% methods, 0% lines covered
Direction 66% methods, 22% lines covered
GeoHash 72% methods, 70% lines covered
LatLong 60% methods, 38% lines covered
package-info.java
Parity 100% methods, 100% lines covered
test

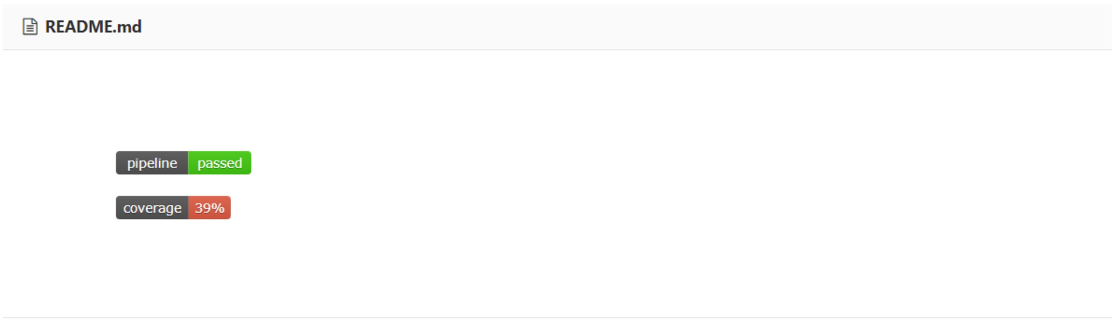
- Total coverage



Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
com.github.davidmoten.geo	🟥🟩	67%	🟥🟩	60%	69	149	113	348	25	68	3	10
com.github.davidmoten.geo.mem	🟥	0%	🟥	0%	30	30	61	61	20	20	3	3
com.github.davidmoten.geo.util		36%	🟥	50%	2	4	2	6	0	2	0	1
Total	978 of 2,326	57%	86 of 186	53%	101	183	176	415	45	90	6	14

4.3 CI result snapshot (3 iterations for CI)

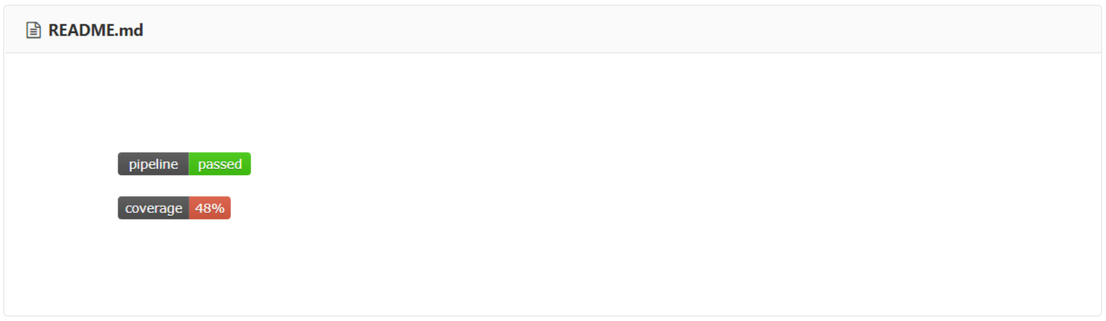
- CI#1


README.md

pipeline passed

coverage 39%

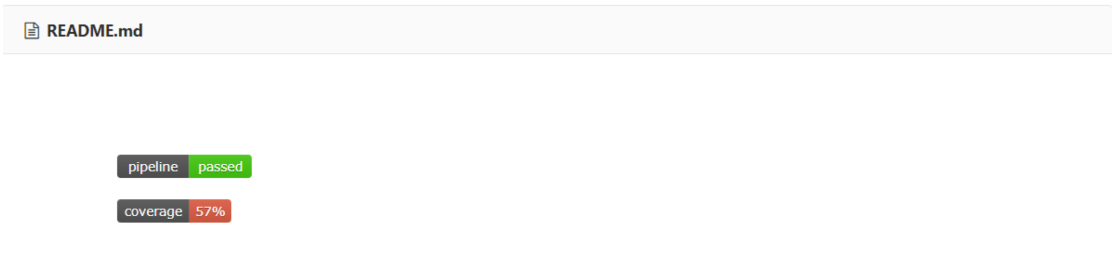
- CI#2


README.md

pipeline passed

coverage 48%

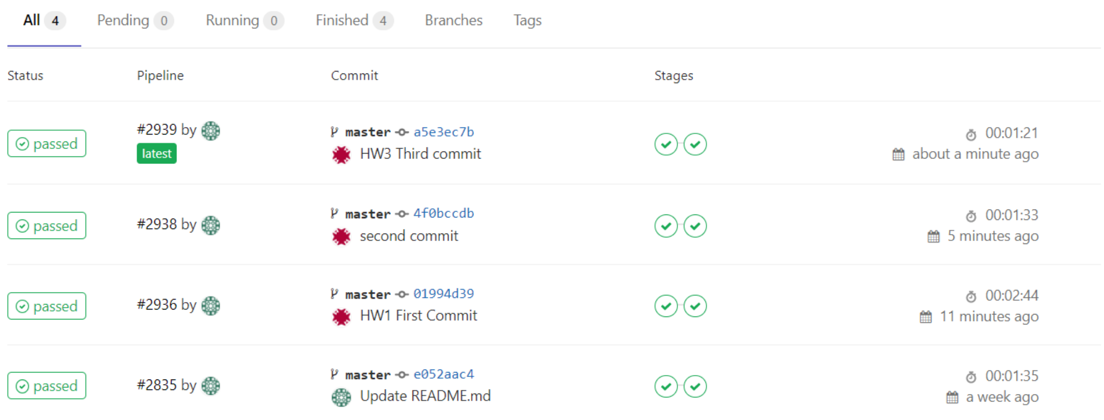
- CI#3


README.md

pipeline passed

coverage 57%

- CI Pipeline


All 4 Pending 0 Running 0 Finished 4 Branches Tags

Status	Pipeline	Commit	Stages	
passed	#2939 by latest	master a5e3ec7b HW3 Third commit	✓ ✓	00:01:21 about a minute ago
passed	#2938 by	master 4f0bccdb second commit	✓ ✓	00:01:33 5 minutes ago
passed	#2936 by	master 01994d39 HW1 First Commit	✓ ✓	00:02:44 11 minutes ago
passed	#2835 by	master e052aac4 Update README.md	✓ ✓	00:01:35 a week ago

5 Summary

In Lab 1, **15 test cases have been designed and implemented using JUnit**. 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 57%.** Thus, the test requirements described in Section 1 are satisfied.

為了能更完整的測出這個程式，在測試之前，也學習了 GeoProject 的使用方法和背景知識。這次作業完成而 3 次的 CI，總共測 15 個 methods，且 **total statement coverage 為 57%，**符合此次作業要求。對於 +hashContains(String, double, double):boolean，也增加了邊界測試。