Lab 2 Report

Name 陳宇宏
Student ID 110598067
Date 2022/04/17

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	1	4/16
2	Learn **ISP** and JUnit	1	4/16
3	Design test cases for the selected methods	2	4/16
4	Implement test cases	6	4/17

5	Perform tests	0.2	4/17
…			
n	Complete Lab2 report	1	4/17

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

都寫在 **excel** 檔案裡

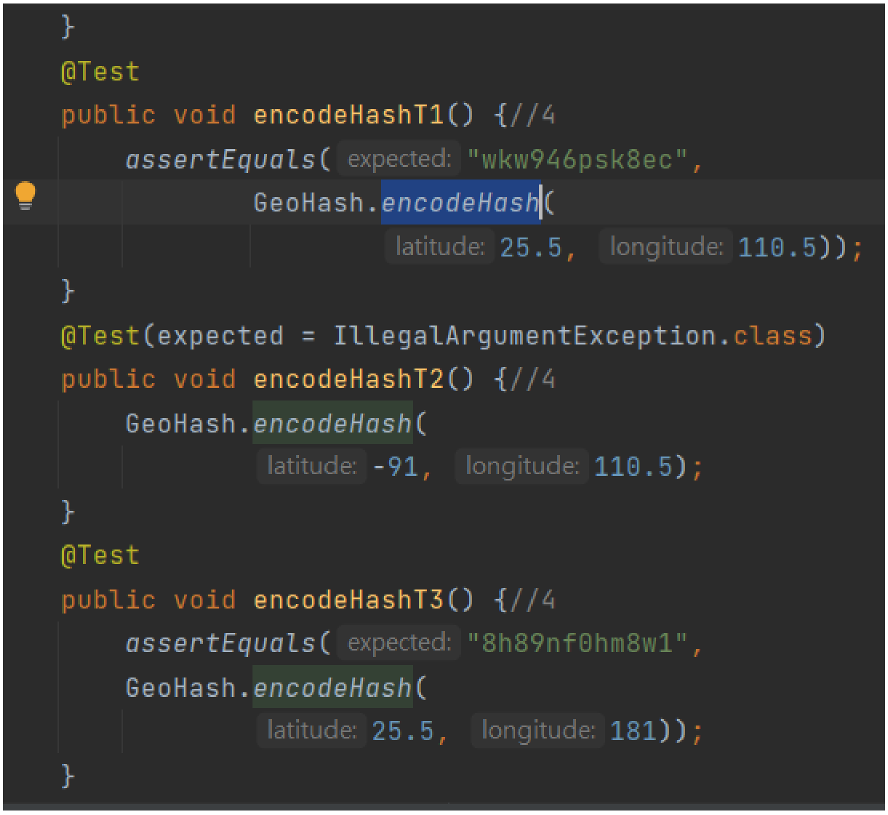
The Excel file of test cases…

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).

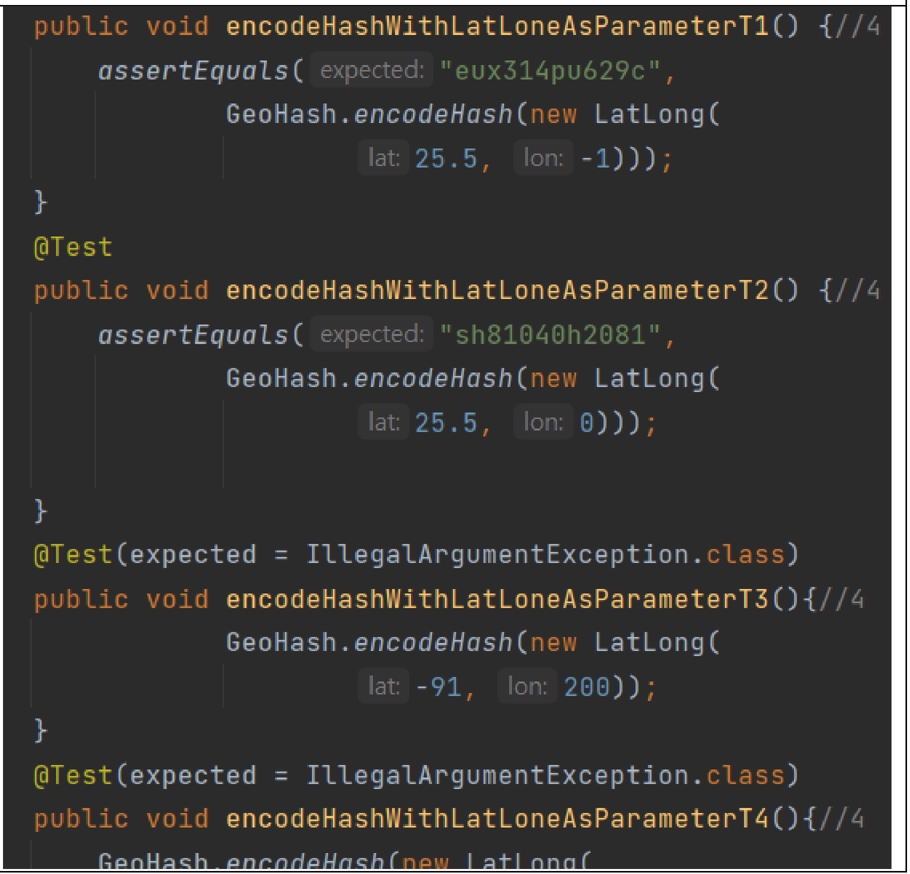
No.	Test method	Source code
1	+encodeHash(double,double):String	

<table>
<tr>
<td></td>
<td></td>
<td>



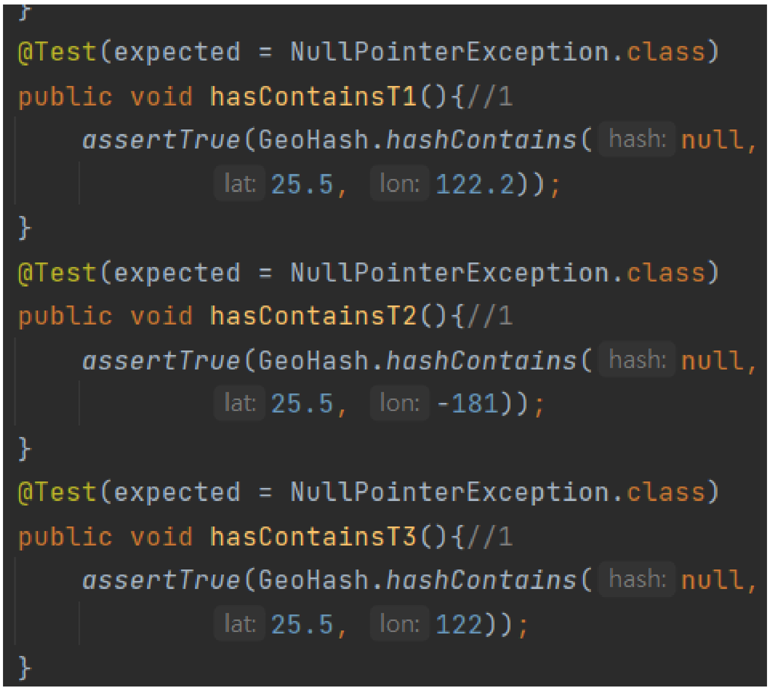
```
}
@Test
public void encodeHashT1() {//4
    assertEquals( expected: "wkw946psk8ec",
            GeoHash.encodeHash(
                    latitude: 25.5,  longitude: 110.5));
}
@Test(expected = IllegalArgumentException.class)
public void encodeHashT2() {//4
    GeoHash.encodeHash(
            latitude: -91,  longitude: 110.5);
}
@Test
public void encodeHashT3() {//4
    assertEquals( expected: "8h89nf0hm8w1",
    GeoHash.encodeHash(
            latitude: 25.5,  longitude: 181));
}
```

</td>
</tr>
<tr>
<td>2</td>
<td>+ encodeHash(LatLong):String</td>
<td>



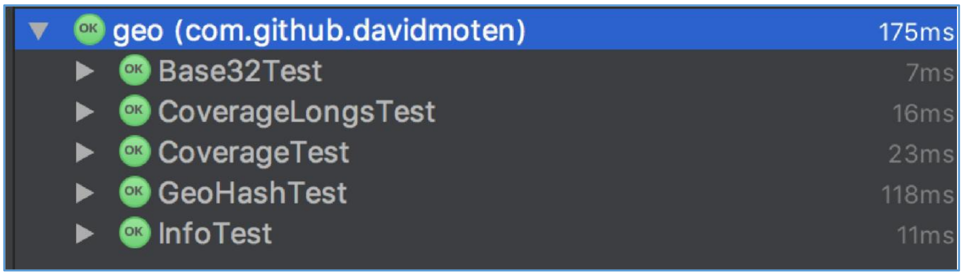
```
public void encodeHashWithLatLoneAsParameterT1() {//4
    assertEquals( expected: "eux314pu629c",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.5,  lon: -1)));
}
@Test
public void encodeHashWithLatLoneAsParameterT2() {//4
    assertEquals( expected: "sh81040h2081",
            GeoHash.encodeHash(new LatLong(
                    lat: 25.5,  lon: 0)));

}
@Test(expected = IllegalArgumentException.class)
public void encodeHashWithLatLoneAsParameterT3(){//4
            GeoHash.encodeHash(new LatLong(
                    lat: -91,  lon: 200));
}
@Test(expected = IllegalArgumentException.class)
public void encodeHashWithLatLoneAsParameterT4(){//4
    GeoHash.encodeHash(new LatLong(
```

</td>
</tr>
</table>

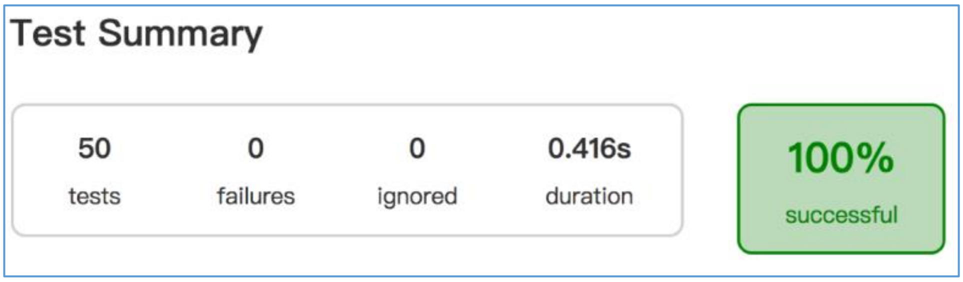
3	+hashContains(String, double, double):boolean	
		}
		@Test(expected = NullPointerException.class)
		public void hasContainsT1(){//1
		    assertTrue(GeoHash.hashContains(hash: null,
		            lat: 25.5, lon: 122.2));
		}
		@Test(expected = NullPointerException.class)
		public void hasContainsT2(){//1
		    assertTrue(GeoHash.hashContains(hash: null,
		            lat: 25.5, lon: -181));
		}
		@Test(expected = NullPointerException.class)
		public void hasContainsT3(){//1
		    assertTrue(GeoHash.hashContains(hash: null,
		            lat: 25.5, lon: 122));
		}

4 Test Results

4.1 JUnit test result snapshot

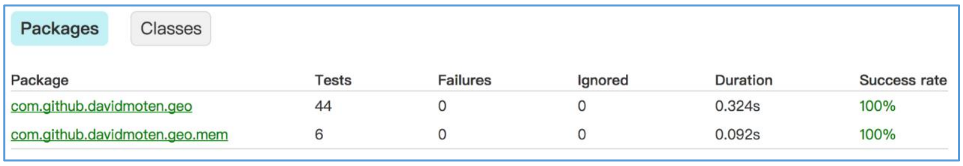


```
▼ OK geo (com.github.davidmoten)    175ms
    ▶ OK Base32Test                   7ms
    ▶ OK CoverageLongsTest           16ms
    ▶ OK CoverageTest                23ms
    ▶ OK GeoHashTest                118ms
    ▶ OK InfoTest                    11ms
```



Test Summary

50	0	0	0.416s	100%
tests	failures	ignored	duration	successful

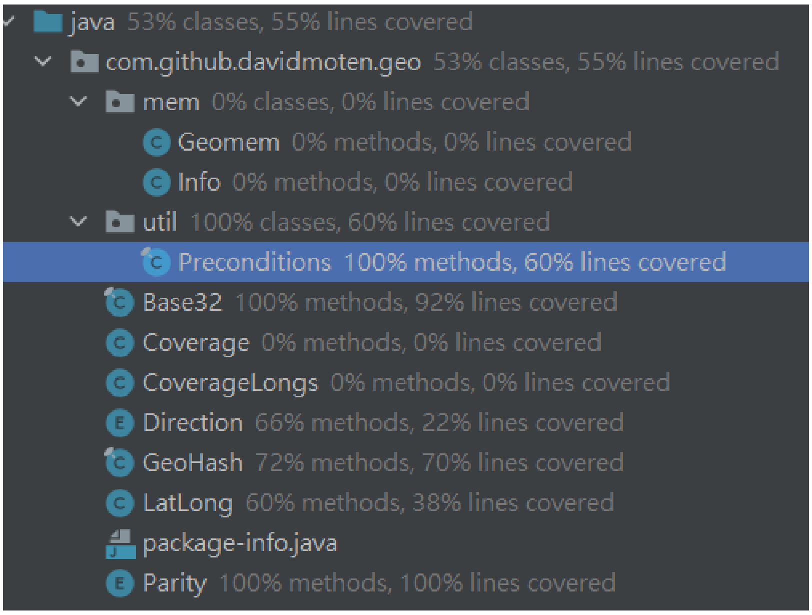


Packages Classes

Package	Tests	Failures	Ignored	Duration	Success rate
com.github.davidmoten.geo	44	0	0	0.324s	100%
com.github.davidmoten.geo.mem	6	0	0	0.092s	100%

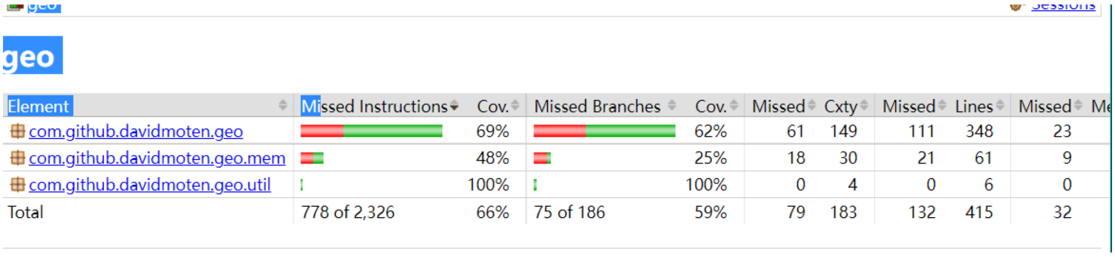
4.2 Code coverage snapshot

- 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

- Total coverage

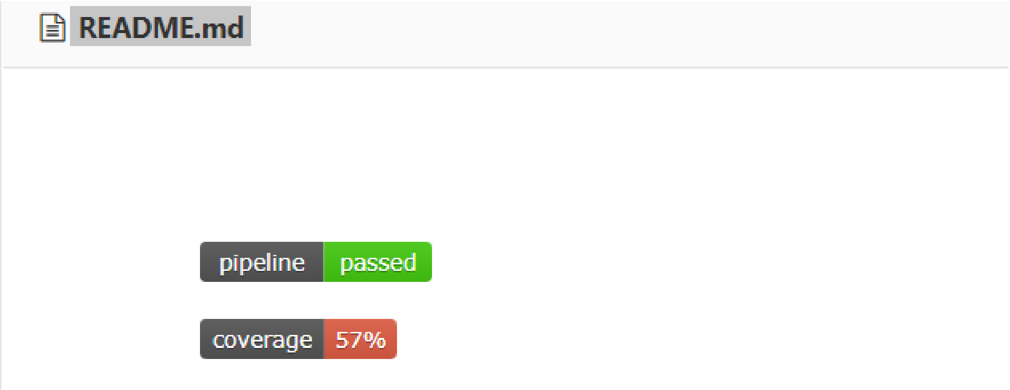


geo

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Me
com.github.davidmoten.geo		69%		62%	61	149	111	348	23	
com.github.davidmoten.geo.mem		48%		25%	18	30	21	61	9	
com.github.davidmoten.geo.util		100%		100%	0	4	0	6	0	
Total	778 of 2,326	66%	75 of 186	59%	79	183	132	415	32	

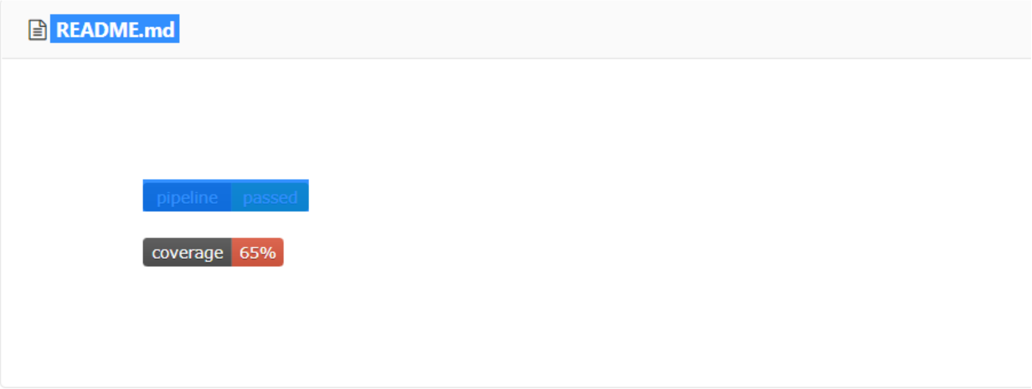
4.3 CI result snapshot (3 iterations for CI)

- CI#1



README.md

- CI#2

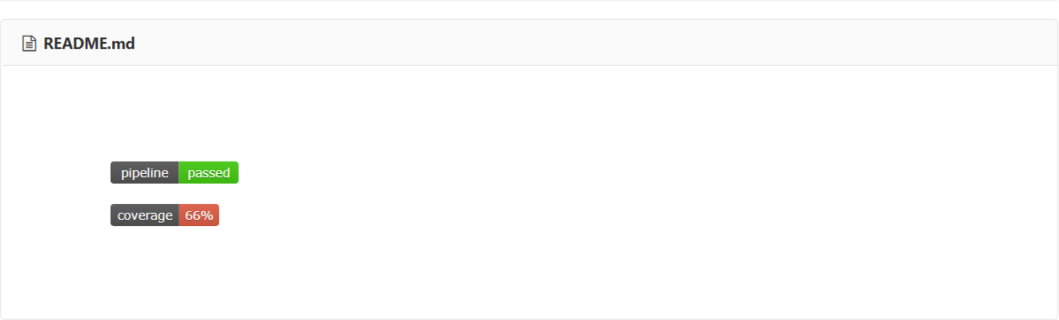


README.md

pipeline passed

coverage 65%

- CI#3

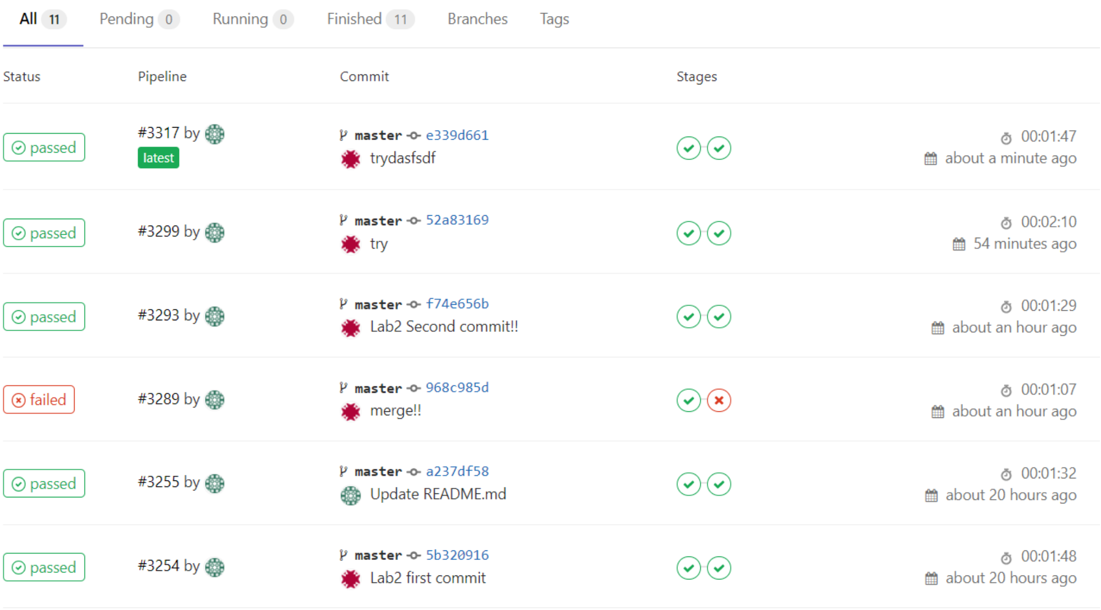


README.md

pipeline passed

coverage 66%

- CI Pipeline



All 11 Pending 0 Running 0 Finished 11 Branches Tags

Status	Pipeline	Commit	Stages	
✓ passed	#3317 by latest	master e339d661 trydasfsdf	✓ ✓	00:01:47 about a minute ago
✓ passed	#3299 by	master 52a83169 try	✓ ✓	00:02:10 54 minutes ago
✓ passed	#3293 by	master f74e656b Lab2 Second commit!!	✓ ✓	00:01:29 about an hour ago
☒ failed	#3289 by	master 968c985d merge!!	✓ ☒	00:01:07 about an hour ago
✓ passed	#3255 by	master a237df58 Update README.md	✓ ✓	00:01:32 about 20 hours ago
✓ passed	#3254 by	master 5b320916 Lab2 first commit	✓ ✓	00:01:48 about 20 hours ago

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 70%. Thus, the test requirements described in Section 1 are satisfied. Some lessons learned in this Lab are …