Assignment 12

Rasika Mohod
rmohod@gmu.edu
G01044774

Consider the PIT mutation tool. Apply PIT to a program of your choosing and generate tests to kill the PIT mutants. Analyze the PIT mutation operators and determine whether or not PIT mutation subsumes branch (edge) coverage (with either the default or the optional operators).

I have used **Pitclipse, PIT mutation tool/plugin for Eclipse IDE** for this assignment. The program on which tool was run is simple class with two methods to compute if the given number if even or odd. The code for the program is given as follows:

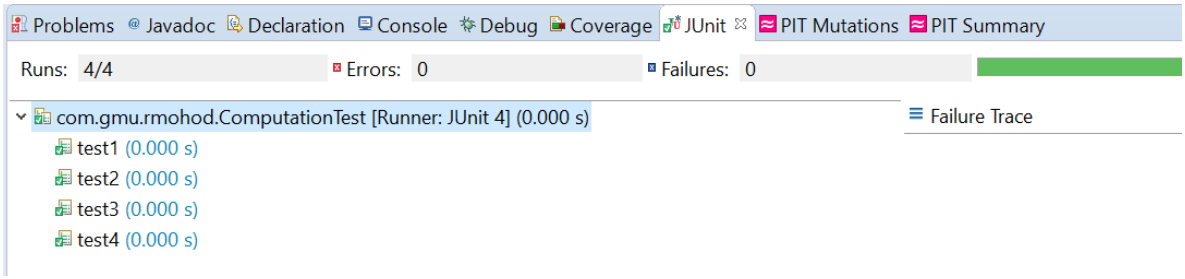
Computation.java:

```
package com.gmu.rmohod;
/**
 * Class to compute if a number is even or odd
 *
 * @author Rasika
 *
 */
public class Computation {

	int number;

	Computation(int x)
	{
		number = x;
	}

	boolean isEven()
	{
		if(number%2==0)
			return true;
		else
			return false;
	}

	boolean isOdd()
	{
		if(number%2==1)
			return true;
		else
			return false;
	}

}
```

ComputationTest.java:

```java
package com.gmu.rmohod;

import static org.junit.Assert.*;
import org.junit.Test;

/**
 * JUnit Test class for Computation class
 *
 * @author Rasika
 *
 */
public class ComputationTest {

	@Test
	public void test1() {
		Computation com = new Computation(5);

		assertEquals(true,com.isOdd());
	}

	@Test
	public void test2() {
		Computation com = new Computation(4);

		assertEquals(false,com.isOdd());
	}

	@Test
	public void test3() {
		Computation com = new Computation(5);

		assertEquals(false,com.isEven());
	}

	@Test
	public void test4() {
		Computation com = new Computation(4);

		assertEquals(true,com.isEven());
	}

}
```

JUnit Output:



Console output after running PIT Mutation tool:

```
================================================================================
===
- Timings
================================================================================
===
> scan classpath : < 1 second
> coverage and dependency analysis : < 1 second
> build mutation tests : < 1 second
> run mutation analysis : < 1 second
--------------------------------------------------------------------------------
---
> Total  : 1 seconds
--------------------------------------------------------------------------------
---
================================================================================
===
- Statistics
================================================================================
===
>> Generated 8 mutations Killed 8 (100%)
>> Ran 10 tests (1.25 tests per mutation)
================================================================================
===
- Mutators
================================================================================
===
> org.pitest.mutationtest.engine.gregor.mutators.MathMutator
>> Generated 2 Killed 2 (100%)
> KILLED 2 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
--------------------------------------------------------------------------------
---
> org.pitest.mutationtest.engine.gregor.mutators.ReturnValsMutator
>> Generated 4 Killed 4 (100%)
> KILLED 4 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
--------------------------------------------------------------------------------
---
> org.pitest.mutationtest.engine.gregor.mutators.NegateConditionalsMutator
>> Generated 2 Killed 2 (100%)
> KILLED 2 SURVIVED 0 TIMED_OUT 0 NON_VIABLE 0
> MEMORY_ERROR 0 NOT_STARTED 0 STARTED 0 RUN_ERROR 0
> NO_COVERAGE 0
--------------------------------------------------------------------------------
---
Sending results: PitResults [htmlResultFile=C:\workspace
eclipse\.metadata\.plugins\org.pitest.pitclipse.core\html_results\20170424115
5\index.html, projects=[SWE637Assignment12]]
Closing server
Closed
```

PIT Mutations:

In all 8 mutants were generated by the tool.

Problems | Javadoc | Declaration | Console | Debug | Coverage | JUnit | PIT Mutations | PIT Summary

- KILLED (8)
  - SWE637Assignment12 (8)
    - com.gmu.rmohod (8)
      - com.gmu.rmohod.Computation (8)
        - 14: Replaced integer modulus with multiplication
        - 14: negated conditional
        - 15: replaced return of integer sized value with (x == 0 ? 1 : 0)
        - 17: replaced return of integer sized value with (x == 0 ? 1 : 0)
        - 22: Replaced integer modulus with multiplication
        - 22: negated conditional
        - 23: replaced return of integer sized value with (x == 0 ? 1 : 0)
        - 25: replaced return of integer sized value with (x == 0 ? 1 : 0)

PIT Summary:

All of the 8 mutants were successfully killed by my 4 test cases.

Given below is the complete summary of PIT Test coverage on my program.

Problems | Javadoc | Declaration | Console | Debug | Coverage | JUnit | PIT Mutations | PIT Summary

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
1	100%	9/9	100%	8/8

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
com.gmu.rmohod	1	100%	9/9	100%	8/8

Report generated by PIT 1.1.9

Problems | Javadoc | Declaration | Console | Debug | Coverage | JUnit | PIT Mutations | PIT Summary

Pit Test Coverage Report

Package Summary

com.gmu.rmohod

Number of Classes	Line Coverage		Mutation Coverage	
1	100%	9/9	100%	8/8

Breakdown by Class

Name	Line Coverage		Mutation Coverage	
Computation.java	100%	9/9	100%	8/8

Report generated by PIT 1.1.9

Problems | Javadoc | Declaration | Console | Debug | Coverage | JUnit | PIT Mutations | PIT Summary

Computation.java

```
1   package com.gmu.rmohod;
2
3   public class Computation {
4
5           int number;
6
7           Computation(int x)
8           {
9                   number = x;
10          }
11
12          boolean isEven()
13          {
14 2                if(number%2==0)
15 1                        return true;
16                  else
17 1                        return false;
18          }
19
20          boolean isOdd()
21          {
22 2                if(number%2==1)
23 1                        return true;
24                  else
25 1                        return false;
26          }
27
28  }
```

Mutations

```
14  1. Replaced integer modulus with multiplication → KILLED
    2. negated conditional → KILLED
15  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
17  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
22  1. Replaced integer modulus with multiplication → KILLED
    2. negated conditional → KILLED
23  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
25  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
```

Active mutators

- INCREMENTS_MUTATOR
- CONDITIONALS_BOUNDARY_MUTATOR
- RETURN_VALS_MUTATOR
- VOID_METHOD_CALL_MUTATOR
- INVERT_NEGS_MUTATOR
- MATH_MUTATOR
- NEGATE_CONDITIONALS_MUTATOR

Tests examined

- com.gmu.rmohod.ComputationTest.test3(com.gmu.rmohod.ComputationTest) (0 ms)
- com.gmu.rmohod.ComputationTest.test4(com.gmu.rmohod.ComputationTest) (0 ms)
- com.gmu.rmohod.ComputationTest.test1(com.gmu.rmohod.ComputationTest) (16 ms)
- com.gmu.rmohod.ComputationTest.test2(com.gmu.rmohod.ComputationTest) (0 ms)

Report generated by PIT 1.1.9

Test Coverage:

Problems | Javadoc | Declaration | Console | Debug | Coverage | JUnit | PIT Mutations | PIT Summary

ComputationTest (1) (Apr 24, 2017 11:46:36 AM)

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
SWE637Assignment12	100.0 %	76	0	76
src	100.0 %	76	0	76
com.gmu.rmohod	100.0 %	76	0	76
Computation.java	100.0 %	25	0	25
Computation	100.0 %	25	0	25
Computation(int)	100.0 %	6	0	6
isEven()	100.0 %	9	0	9
isOdd()	100.0 %	10	0	10
ComputationTest.java	100.0 %	51	0	51
ComputationTest	100.0 %	51	0	51
test1()	100.0 %	12	0	12
test2()	100.0 %	12	0	12
test3()	100.0 %	12	0	12
test4()	100.0 %	12	0	12

Merits of the PIT tool:

- This tool gives detailed description of all the mutants generated.
- It also provides the type of each mutator generated in the code which helps to write test accordingly to kill the respective muatant.
- The test coverage analysis provides a clear and complete view of test coverage in general and also the mutation test coverage by the written test cases.
- Overall this tool was very simple to use and implement in the code.