PIT and Measuring Mutation Coverage

TAN, Shin Hwei

陈馨慧

Southern University of Science and Technology

Slides adapted from https://pitest.org/ and https://blog.scottlogic.com/2017/09/25/mutation-testing.html

No class for Week 8

- Week 10 is deadline free week so we have extended the deadline for Progress Report to 25 April 2022.
- No lecture will be held on Week 8 due to the 清明holiday.
- There will also be no lab for the labs on Monday and Tuesday (4 April, 5 April)
- Other lab sessions (6 April) will use the uploaded slides for revision. It contains reminder for previous lab assignments, and project.

Lab Part 1: PIT

- Mutation testing tool

Outline

In this lab, you will learn about:

- How to use PIT to measure mutation coverage
- Learn how to increase mutation coverage

Recap: Mutation Testing in Java

- PIT is a tool for Mutation testing
- Used for measure the quality of current test cases
- Available as
  - Command-line tool
  - Ant target
  - Maven plugin

Lab Exercise

- Accept the invitation link: https://classroom.github.com/a/q9vuleVv

Several ways of running PIT

- Run using maven

```
mvn clean install                          #clean and compile
mvn test                                   #run jUnit
mvn org.pitest:pitest-maven:mutationCoverage   #run PIT mutation tests
```

- Run using ant

```
ant
ant test
ant pit
```

- Run using command line

```
java -cp target/classes:target/test-classes:lib/junit-4.10.jar:lib/pitest-0.25-SNAPSHOT.jar \
    org.pitest.mutationtest.MutationCoverageReport \
    --reportDir target/pit-reports \
    --targetClasses pitexample.* \
    --sourceDirs src/main/java,src/test/java
```

Simple example: myMethod

Method under test

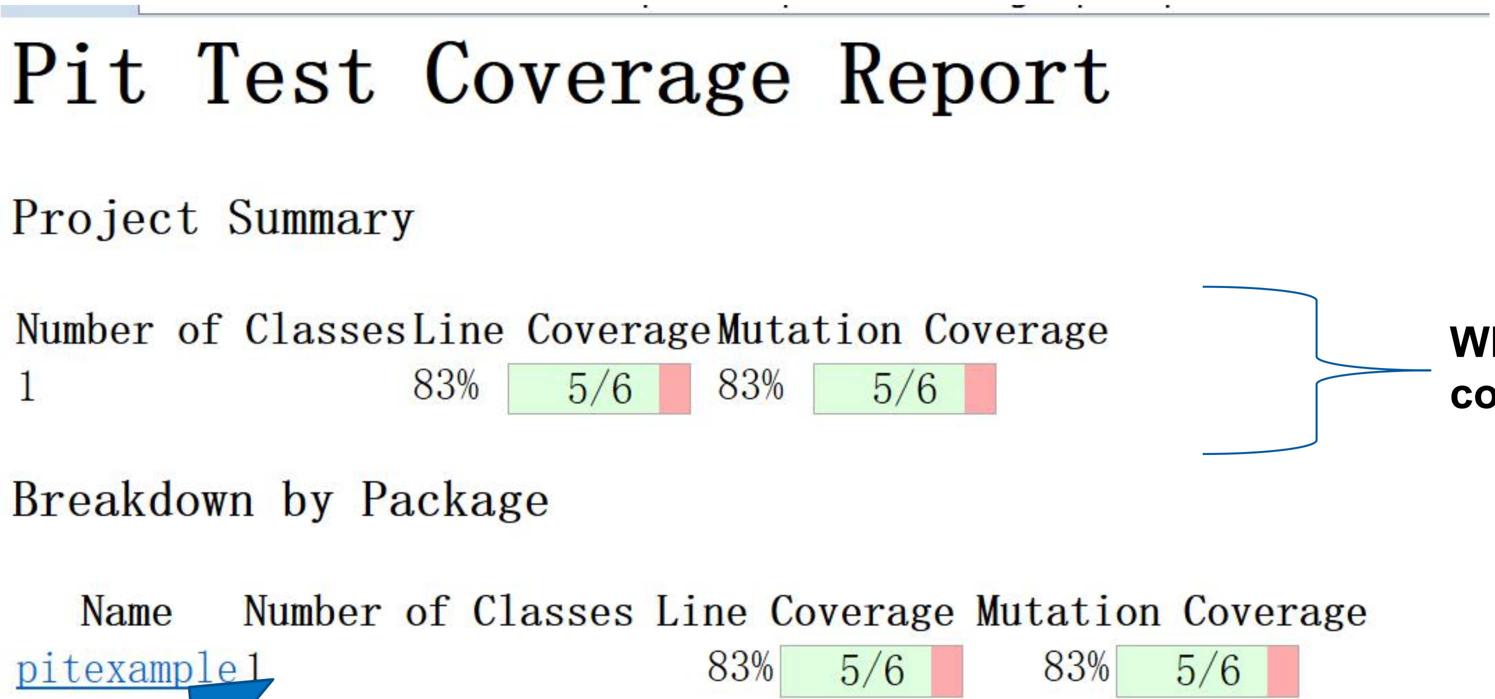
```
public boolean myMethod(int a, boolean flag) {
    if (a > 0) {
        return true;
    }
    if (flag) {
        return true;
    }
    return false;
}
```

Test case

```
@Test public void testMe() {
  MyClass sut = new MyClass();
  assertTrue(sut.myMethod(1, true));
  assertTrue(sut.myMethod(2, true));
  assertTrue(sut.myMethod(1, false));
  assertTrue(sut.myMethod(2, false));
  assertFalse(sut.myMethod(0, false));
}
```

The output of PIT

- Find the mutation score in the index.html in pit-reports folder



Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage	
1	83%	5/6	83%	5/6

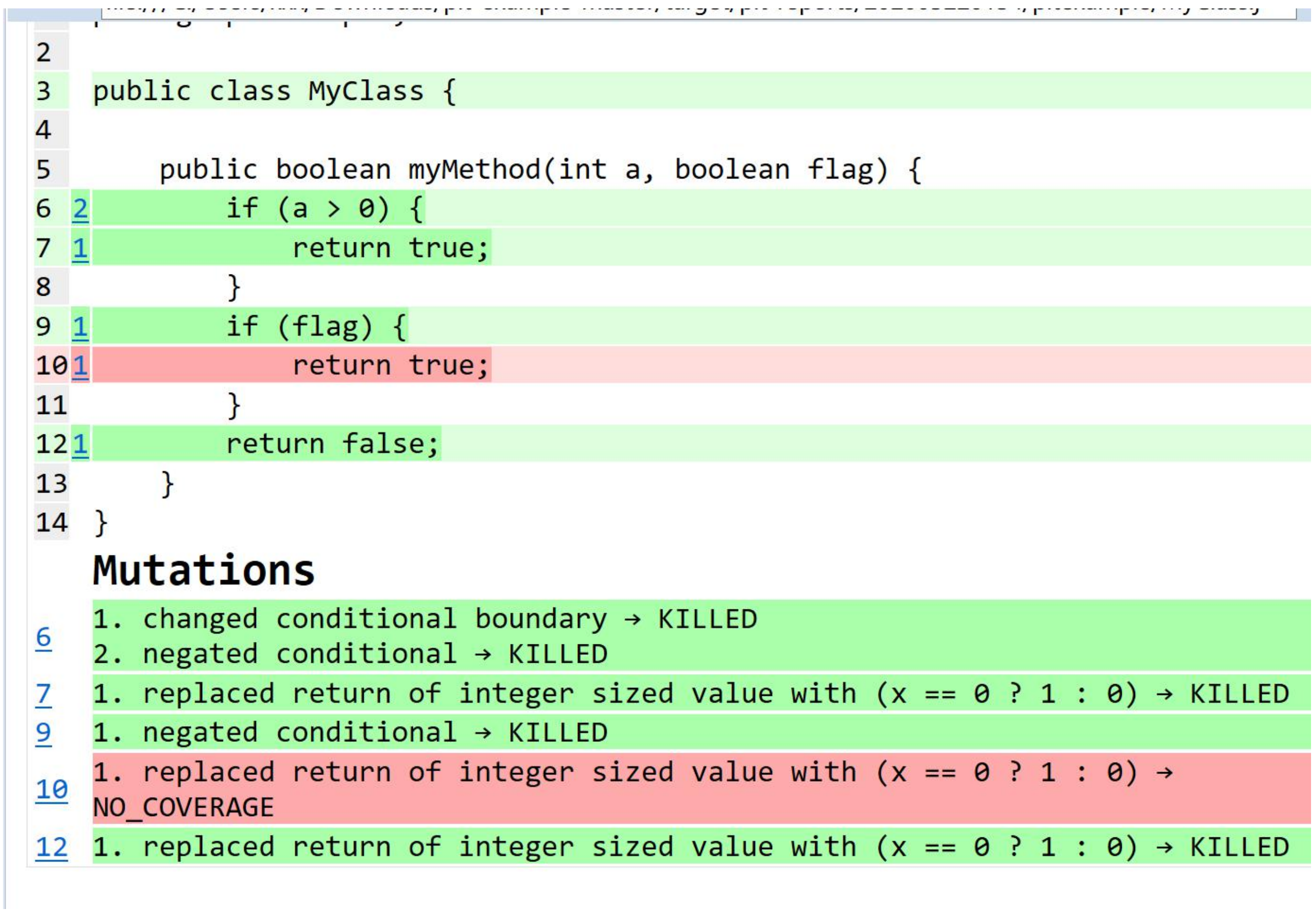
Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage	
pitexample	1	83%	5/6	83%	5/6

Why only 83% mutation coverage?

Click this to check the details

The output of PIT



```
2
3     public class MyClass {
4
5         public boolean myMethod(int a, boolean flag) {
6  2          if (a > 0) {
7  1              return true;
8             }
9  1          if (flag) {
10 1              return true;
11            }
12 1          return false;
13        }
14    }
```

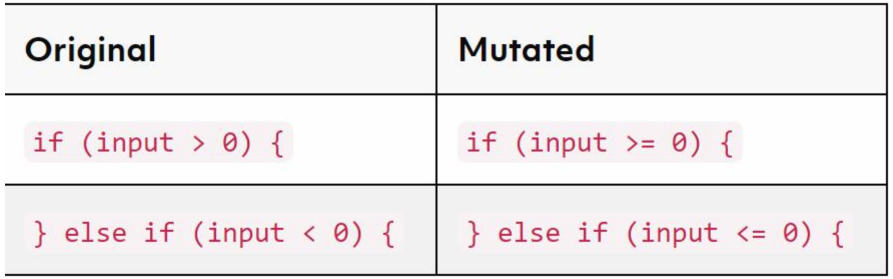
Mutations

```
6   1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
7   1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
9   1. negated conditional → KILLED
10  1. replaced return of integer sized value with (x == 0 ? 1 : 0) →
    NO_COVERAGE
12  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
```

Why are these mutations

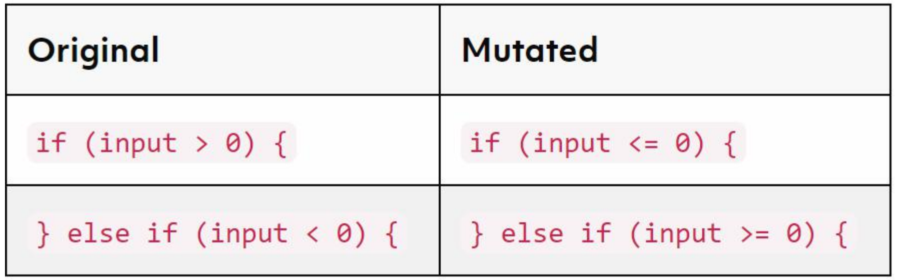
Examples of the Mutations

- **Changed conditional boundary**
  - Changes relational operators to either **add or remove the equals sign**, effectively shifting the boundary by one.



Original	Mutated
`if (input > 0) {`	`if (input >= 0) {`
`} else if (input < 0) {`	`} else if (input <= 0) {`

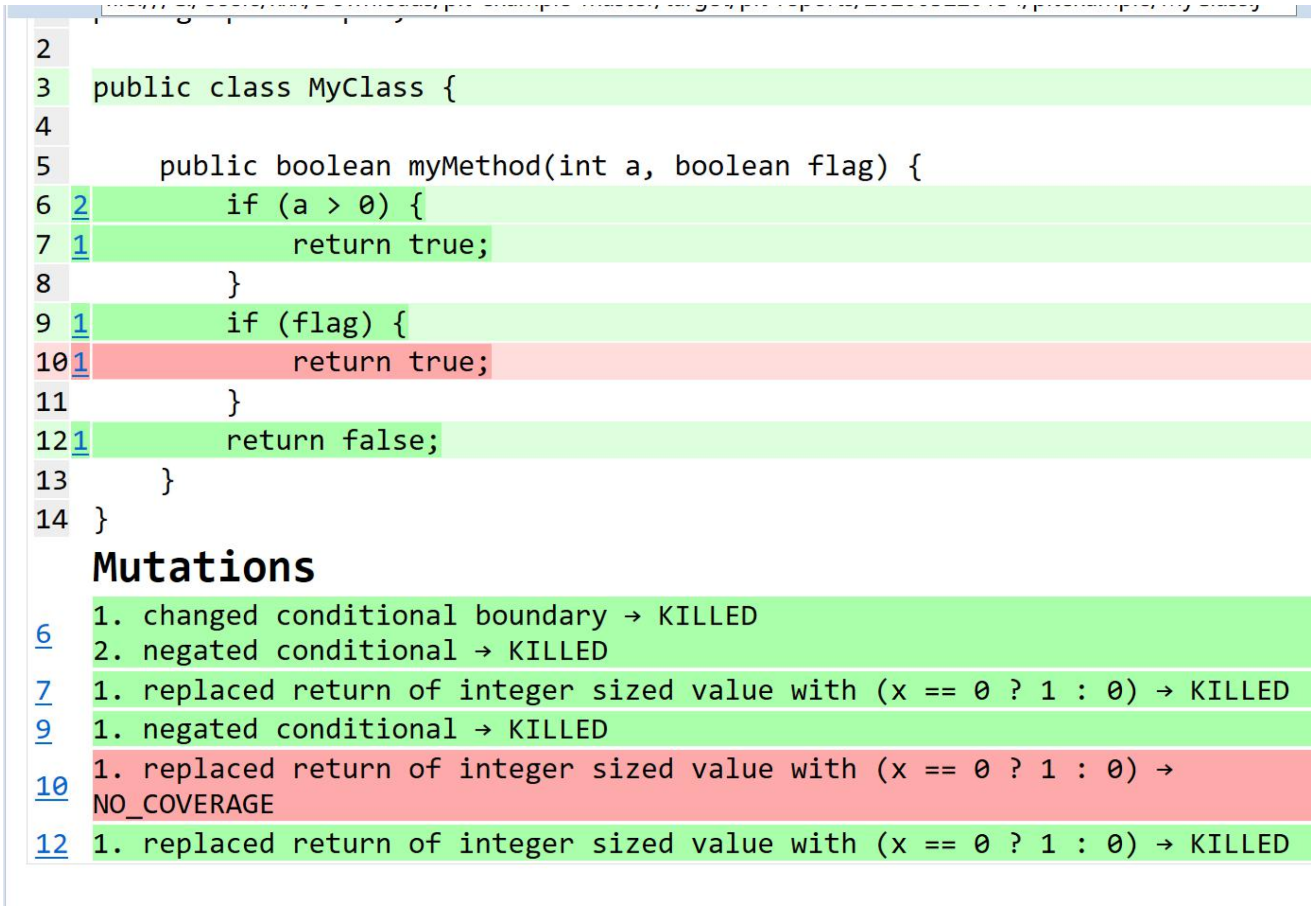
- **Negated conditional**
  - invert the conditional to do the opposite of what it originally did



Original	Mutated
`if (input > 0) {`	`if (input <= 0) {`
`} else if (input < 0) {`	`} else if (input >= 0) {`

From: https://blog.scottlogic.com/2017/09/25/mutation-testing.html

The non-covered mutation



```
2
3      public class MyClass {
4
5          public boolean myMethod(int a, boolean flag) {
6  2           if (a > 0) {
7  1               return true;
8              }
9  1           if (flag) {
10 1               return true;
11             }
12 1           return false;
13         }
14     }
```

Mutations

```
6   1. changed conditional boundary → KILLED
    2. negated conditional → KILLED
7   1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
9   1. negated conditional → KILLED
10  1. replaced return of integer sized value with (x == 0 ? 1 : 0) →
    NO_COVERAGE
12  1. replaced return of integer sized value with (x == 0 ? 1 : 0) → KILLED
```

- **return true->return (x==0? 1: 0)**
  - Same as replacing return true by return false
- Couldn’t detect the case where if(flag==true) return false instead of true
  - **No assertion to check that the case when a>0 and flag true!**
  - Adding assertion to cover this case

Lab Exercise

- Write JUnit test to achieve 100% mutation coverage for both:
  - MyClass.java
  - StockService.java
- Add a README.md with the following information:
  - Name:
  - Student id:
  - JUnit tests for MyClass.java
  - JUnit tests for StockService.java
  - Screenshot of PIT results showing that you achieved 100% mutation coverages for both class

Lab Part 2: Progress report

- Discuss with your groupmates about the progress report
  - Deadline extended to 25 April 2022