Github Repository: https://github.com/PashaBarahimi/Software-Testing-Course-Projects

Last Commit: `b08d828c4b96d013ca6c3dcabfd099d380198527`

بخش اول

1. نتیجه اجرای تست Mutation

```
================================================================================
- Statistics
================================================================================
>> Line Coverage (for mutated classes only): 49/49 (100%)
>> Generated 29 mutations Killed 28 (97%)
>> Mutations with no coverage 0. Test strength 97%
>> Ran 39 tests (1.34 tests per mutation)
```

تعداد Mutant-های ساخته شده

همانطور که در تصویر مشاهده می‌شود، 29 عدد Mutant طی پردازش Pitest ساخته شده‌اند.

تعداد Mutant-های کشته شده

28 عدد از 29 تا Mutant ساخته شده توسط تست‌ها kill شده‌اند که باعث می‌شود به Mutation Coverage برابر با 97 درصد برسیم.

تعداد Mutant-های زنده مانده

تنها 1 عدد از Mutant-ها زنده مانده‌اند که در ادامه (در بخش تحلیل) به توضیح چرایی آن می‌پردازیم.

گزارش Pitest

Pit Test Coverage Report

Project Summary

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	100%	49/49	97%	28/29	97%	28/29

Breakdown by Package

Name	Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
domain	2	100%	49/49	97%	28/29	97%	28/29

Report generated by PIT 1.15.2

Enhanced functionality available at arcmutate.com

Pit Test Coverage Report

Package Summary

domain

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	100%	49/49	97%	28/29	97%	28/29

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
Engine.java	100%	40/40	95%	20/21	95%	20/21
Order.java	100%	9/9	100%	8/8	100%	8/8

Report generated by PIT 1.15.2

Order.java

```
1   package domain;
2
3   import lombok.Getter;
4   import lombok.Setter;
5
6   @Getter
7   @Setter
8   public class Order {
9  1    int id;
10 1    int customer;
11 1    int price;
12 1    int quantity;
13
14      @Override
15      public boolean equals(Object obj) {
16 1        if (obj instanceof Order order) {
17 2            return id == order.id;
18          }
19 1        return false;
20      }
21  }
```

Mutations

```
9   1. replaced int return with 0 for domain/Order::getId → KILLED
10  1. replaced int return with 0 for domain/Order::getCustomer → KILLED
11  1. replaced int return with 0 for domain/Order::getPrice → KILLED
12  1. replaced int return with 0 for domain/Order::getQuantity → KILLED
16  1. negated conditional → KILLED
17  1. negated conditional → KILLED
    2. replaced boolean return with true for domain/Order::equals → KILLED
19  1. replaced boolean return with true for domain/Order::equals → KILLED
```

```
59      int getCustomerFraudulentQuantity(Order order) {
60
61          var averageOrderQuantity = getAverageOrderQuantityByCustomer(order.customer);
62
63 2        if (order.quantity > averageOrderQuantity) {
64 2            return order.quantity - averageOrderQuantity;
65          }
66
67          return 0;
68      }
69
```

Mutations

18	1. negated conditional → KILLED
19	1. Replaced integer addition with subtraction → KILLED
20	1. Changed increment from 1 to -1 → KILLED
24	1. negated conditional → KILLED
28	1. replaced int return with 0 for domain/Engine::getAverageOrderQuantityByCustomer → KILLED 2. Replaced integer division with multiplication → KILLED
32	1. negated conditional → KILLED
40	1. negated conditional → KILLED
44	1. negated conditional → KILLED
48	1. negated conditional → KILLED
49	1. Replaced integer subtraction with addition → KILLED
51	1. negated conditional → KILLED 2. Replaced integer subtraction with addition → KILLED
56	1. replaced int return with 0 for domain/Engine::getQuantityPatternByPrice → KILLED
63	1. changed conditional boundary → SURVIVED 2. negated conditional → KILLED
64	1. replaced int return with 0 for domain/Engine::getCustomerFraudulentQuantity → KILLED 2. Replaced integer subtraction with addition → KILLED
71	1. negated conditional → KILLED
76	1. negated conditional → KILLED
81	1. replaced int return with 0 for domain/Engine::addOrderAndGetFraudulentQuantity → KILLED

تحلیل گزارش Pitest

همانطوز که مشاهده می‌شود، تنها Mutant-ای که kill نشده است، اپراتور Conditionals Boundary است که روی شرط if در تابع زیر اعمال شده است:

```
int getCustomerFraudulentQuantity(Order order) {
    var averageOrderQuantity =
getAverageOrderQuantityByCustomer(order.customer);

    if (order.quantity > averageOrderQuantity) {
        return order.quantity - averageOrderQuantity;
    }

    return 0;
}
```

اتفاقی که توسط انجام این Mutation رخ داده، این است که اپراتور > در شرط if به اپراتور ≥ تبدیل شده است. دلیل kill نشدن این Mutant این است که اگر تستی را در نظر بگیریم که در آن، مقدار `order.quantity` با مقدار `averageOrderQuantity` برابر باشد، مقدار return شده در داخل if با مقدار return شده در بیرون if که مقدار 0 است، برابر خواهد بود. در واقع در این حالت خروجی در هر دو حالت یکسان بوده و به اصطلاح، Equivalent Mutation داریم. به همین دلیل، امکان kill شدن این Mutant با هیچ تستی وجود ندارد.

2. رابطه Mutation Coverage بالا در میان خطر Refactoring

هدف از Refactoring، بهبود ساختار کد و طراحی، خوانایی بالاتر، دیباگ سریع‌تر، و توسعه سریع‌تر است. اما نکته مهم در خصوص Refactoring این است که رفتار خارجی کد نباید در حین آن تغییر کند و باید عینا مانند رفتار قبل از Refactoring باشد.

از طرفی، Refactoring هم بر روی کد اصلی قابل انجام است و هم بر روی تست‌ها. زمانی که این کار را بر روی کد اصلی اعمال می‌کنیم، در صورتی که تست‌های خوبی داشته باشیم، می‌توان تقریبا مطمئن بود که رفتار کد در حین انجام این کار، تغییر نمی‌کند زیرا در صورت تغییر، تست‌ها fail می‌شوند و متوجه تغییر رفتار می‌شویم. اما زمانی که می‌خواهیم عمل Refactoring را بر روی خود تست‌ها اعمال کنیم، هیچ Safety Net-ای وجود ندارد که مطمئن باشیم رفتار تست‌ها عینا مانند پیش از Refactor باشد. به همین دلیل، می‌توانیم از Mutation Testing برای اطمینان حاصل کردن از ثبات رفتار تست‌ها استفاده کنیم. این مورد به این صورت این که Mutation Coverage پس از انجام Refactoring باید دقیقا مانند پیش از Refactoring باشد. در این حالت، تا حد خوبی می‌توانیم مطمئن باشیم که Refactoring به درستی انجام شده است.

منبع

بخش دوم

برای ایجاد pipeline، فایل زیر را در آدرس github/workflow/maven.yml. قرار دادیم که هر دو پروژه را در زمان پوش شدن کد، تست کند. در صورتی که تست‌ها پاس نشوند، pipeline هم fail خواهد شد.

```yaml
name: Java CI with Maven

on:
  push:
    branches:
      - master
  pull_request:
    branches:
      - master

jobs:
  test:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Set up JDK 19
      uses: actions/setup-java@v3
      with:
        java-version: '19'
        distribution: 'oracle'
        cache: 'maven'

    - name: Run Baloot 1 tests
      run: cd Baloot1 && mvn test -B -f pom.xml

    - name: Run Baloot 2 tests
      run: cd Baloot2 && mvn test -B -f pom.xml
```

نتیجه اجرای pipeline در تصویر زیر قابل مشاهده است: