به نام خدا

تمرین کامپیوتری پنجم – تست نرم افزار

اميرحسين راحتى 810100144

على ممتحن 810100213

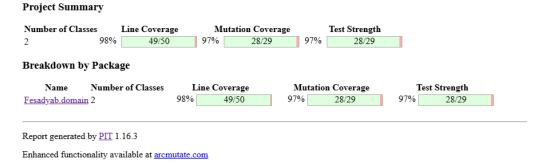
در این تمرین ، ابتدا pitest را به پروژه اضافه میکنیم . چون در این تمرین از استاندارد Junit5 استفاده میکنیم ، باید فایل pom به صورت زیر کانفیگ شود:

سپس برای بررسی Mutation coverage از دستور زیر استفاده میکنیم:

mvn test-compile org.pitest:pitest-maven:mutationCoverage

خروجی pitest به صورت زیر خواهد بود:

Pit Test Coverage Report



Pit Test Coverage Report

Package Summary

Fesadyab.domain

Number of Classes		Line Coverage Mutation Coverage		Test Strength		
2	98%	49/50	97%	28/29	97%	28/29

Breakdown by Class

Name	L	ine Coverage	Mu	tation Coverage	Test Strength	
Transaction.java	89%	8/9	100%	4/4	100%	4/4
TransactionEngine.java	100%	41/41	96%	24/25	96%	24/25

Report generated by PIT 1.16.3

در کلاس Transaction داریم:

Transaction.java

```
package Fesadyab.domain;
3 import lombok.Getter;
4 import lombok.Setter;
6
   @Getter
7
   @Setter
8 public class Transaction {
9
   int transactionId;
10
   int accountId;
11 int amount;
12 boolean isDebit;
13
       @Override
14
       public boolean equals(Object obj) {
15
16 1
            if (obj instanceof Transaction transaction) {
17 2
                return transactionId == transaction.transactionId;
18
19 1
           return false;
20
21 }
    Mutations
16 1. negated conditional → KILLED
1. negated conditional → KILLED
2. replaced boolean return with true for Fesadyab/domain/Transaction::equals → KILLED
19 1. replaced boolean return with true for Fesadyab/domain/Transaction::equals → KILLED
```

فقط متغیر isDebit کاور نشده است و چون در متد های این کلاس از آن استفاده نشده است.

در کلاس TransactionEngine داریم :

```
if (txn.transactionId == previous.transactionId) {
             42
             43
             44 2
                             if (txn.amount <= threshold) {
                                 continue;
             47
                             if (diff == 0) {
             48 1
             49 1
                                 diff = txn.amount - previous.amount;
                             } else if (diff != txn.amount - previous.amount) {
             52
                                 return 0;
             54
             56 1
             57
             58
             59
                     int detectFraudulentTransaction(Transaction txn) {
                         var averageAmount = getAverageTransactionAmountByAccount(txn.accountId);
             62 4
                         if (txn.isDebit && txn.amount > 2 * averageAmount) {
                             return txn.amount - 2 * averageAmount; // Excessive debit, marked as suspicious
             633
             64
             67
                     }
             68
                     public int addTransactionAndDetectFraud(Transaction txn) {
                         if (transactionHistory.contains(txn)) {
             71
             72
             73
                         var fraudScore = detectFraudulentTransaction(txn);
                         if (fraudScore == 0) {
                             fraudScore = getTransactionPatternAboveThreshold(THRESHOLD);
             77
             78
             79
                         transactionHistory.add(txn);
             80 1
             81
             82 }
58
59
          int detectFraudulentTransaction(Transaction txn) {
60
              var average\mount = get\verageTransaction\mountRv\ccount(tyn accountId):
61

    detectFraudulentTransaction : changed conditional boundary → SURVIVED

62 <u>4</u>
               detectFraudulentTransaction : Replaced integer multiplication with division → KILLED
63 3
           3. detectFraudulentTransaction : negated conditional → KILLED 4. detectFraudulentTransaction : negated conditional → KILLED
64
```

فقط قسمت قرمز رنگ کاور نشده است . دلیل آن این است که در حالتی که averageAmout صفر است ، با تغییر علامت < به =< ، نتیجه فرقی نمیکند و عملا امکان کاور کردن آن در این حالت وجود ندارد .

تاثیر mutation بالا در refactoring : اگر تستهای موجود نتوانند تغییرات کوچکی که در کد اعمال شده را شناسایی کنند، این نشاندهنده ضعف در پوشش تستها است. در زمانRefactoring ، این ضعف می تواند باعث شود تغییرات ناخواسته یا باگها شناسایی نشوند. همچنین در Refactoring ، ممکن است تغییراتی که به نظر بی خطر هستند، منجر به تغییر رفتار کد شوند Mutation Coverage این تغییرات را شناسایی می کند.