## ریپو گیت‌هاب

**Repository:** `https://github.com/elahekhodaverdi/SWT-Fall03`
**Commit Hash:** abd418e5e7f2911d7d387ef032a61838f6546e74

## بخش اول

نتیجه اجرای mutation test:

```
- Statistics
>> Line Coverage (for mutated classes only): 50/50 (100%)
>> Generated 29 mutations Killed 28 (97%)
>> Mutations with no coverage 0. Test strength 97%
>> Ran 37 tests (1.28 tests per mutation)
```

# Pit Test Coverage Report

**Project Summary**

| Number of Classes | Line Coverage | | Mutation Coverage | | Test Strength | |
|---|---|---|---|---|---|---|
| 2 | 100% | 50/50 | 97% | 28/29 | 97% | 28/29 |

**Breakdown by Package**

| Name | Number of Classes | Line Coverage | | Mutation Coverage | | Test Strength | |
|---|---|---|---|---|---|---|---|
| domain | 2 | 100% | 50/50 | 97% | 28/29 | 97% | 28/29 |

Report generated by PIT 1.17.2

با توجه به آمار بالا:

- 29 عدد mutant ساخته شده است.
- از تعداد بالا 28تای آن‌ها توسط تست‌ها kill شده‌اند و 97% mutant coverage داریم. یکی از این mutantها kill نمی‌شود که در ادامه به پیدا کردن و علت آن می‌پردازیم.

- یک mutant زنده مانده است.

گزارش Pitest نهایی داده شده به صورت زیر است. همانطور که مشخص است mutant کشته نشده مربوط به کلاس TransactionEngine است.

# Pit Test Coverage Report

## Package Summary

### domain

| Number of Classes | Line Coverage | | Mutation Coverage | | Test Strength | |
|---|---|---|---|---|---|---|
| 2 | 100% | 50/50 | 97% | 28/29 | 97% | 28/29 |

### Breakdown by Class

| Name | Line Coverage | | Mutation Coverage | | Test Strength | |
|---|---|---|---|---|---|---|
| Transaction.java | 100% | 9/9 | 100% | 4/4 | 100% | 4/4 |
| TransactionEngine.java | 100% | 41/41 | 96% | 24/25 | 96% | 24/25 |

Report generated by PIT 1.17.2

با توجه به عکس زیر تنها mutant-ای که kill نمی‌شود در شرط مربوط به detectFraudulent است.

```
59      int detectFraudulentTransaction(Transaction txn) {
60          var averageAmount = getAverageTransactionAmountByAccount(txn.accountId);
61
62 4        if (txn.isDebit && txn.amount > 2 * averageAmount) {
63 3            return txn.amount - 2 * averageAmount;  // Excessive debit, marked as suspicious
64          }
65
66          return 0;
67      }
```

علت آن این است که در حالتی که علامت < به =< تبدیل شود و در تست amount = 2*averageAmount باشد آنگاه نتیجه حالت صحیح سیستم و حالت mutate شده یکسان است و نتیجه تغییر نخواهد کرد (برای هر دو حالت نتیجه 0 خواهد بود). لذا در این حالت Equivalent Mutation داریم و امکان kill شدن این mutation وجود ندارد.

**تاثیر mutation coverage بالا در میزان خطر refactoring:**

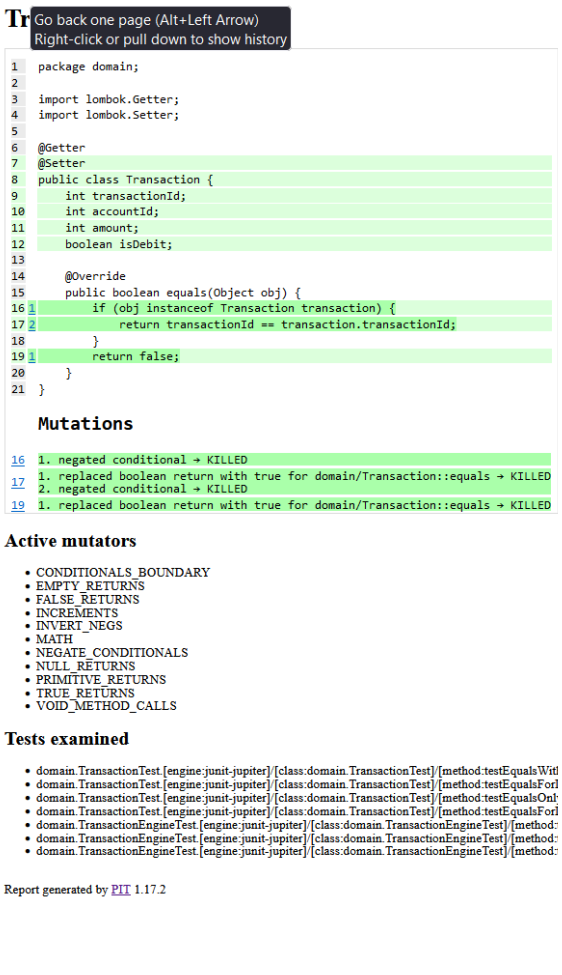
در فرایند refactoring هدف بهبود ساختار کد، افزایش خوانایی، دیباگ سریع‌تر و تسهیل در توسعه است. نکته اصلی در اینجا حفظ رفتار خارجی کد است؛ به این معنا که کد باید همانند قبل عمل کند و تغییری در عملکرد آن ایجاد نشود. Refactoring می‌تواند هم روی کد اصلی و هم روی تست‌ها اعمال شود. وقتی این تغییرات روی کد اصلی انجام می‌شود، وجود تست‌های مناسب به ما این اطمینان را می‌دهد که رفتار کد در

طول این فرآیند ثابت باقی می‌ماند. در صورت بروز هر تغییری در رفتار، تست‌ها شکست می‌خورند (fail می‌شوند) و ما متوجه مشکل می‌شویم.

از سوی دیگر، وقتی قصد داریم Refactoring را روی خود تست‌ها انجام دهیم، چنین اطمینانی وجود ندارد. دلیل این امر نبود Safety Net در این نوع تغییرات است. برای رفع این مشکل می‌توان از Mutation Testing استفاده کرد که به ما کمک می‌کند مطمئن شویم تست‌ها همچنان قابل اعتماد هستند.

در نهایت، میزان Mutation Coverage باید پیش و پس از Refactoring در یک سطح باقی بماند. این موضوع نشان می‌دهد که Refactoring به درستی انجام شده و رفتار کد و تست‌ها ثابت مانده است.

گزارش کامل مربوط به Pitest:



Tr

Go back one page (Alt+Left Arrow)
Right-click or pull down to show history

```
1  package domain;
2
3  import lombok.Getter;
4  import lombok.Setter;
5
6  @Getter
7  @Setter
8  public class Transaction {
9      int transactionId;
10     int accountId;
11     int amount;
12     boolean isDebit;
13
14     @Override
15     public boolean equals(Object obj) {
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

17 1. replaced boolean return with true for domain/Transaction::equals → KILLED
2. negated conditional → KILLED

19 1. replaced boolean return with true for domain/Transaction::equals → KILLED

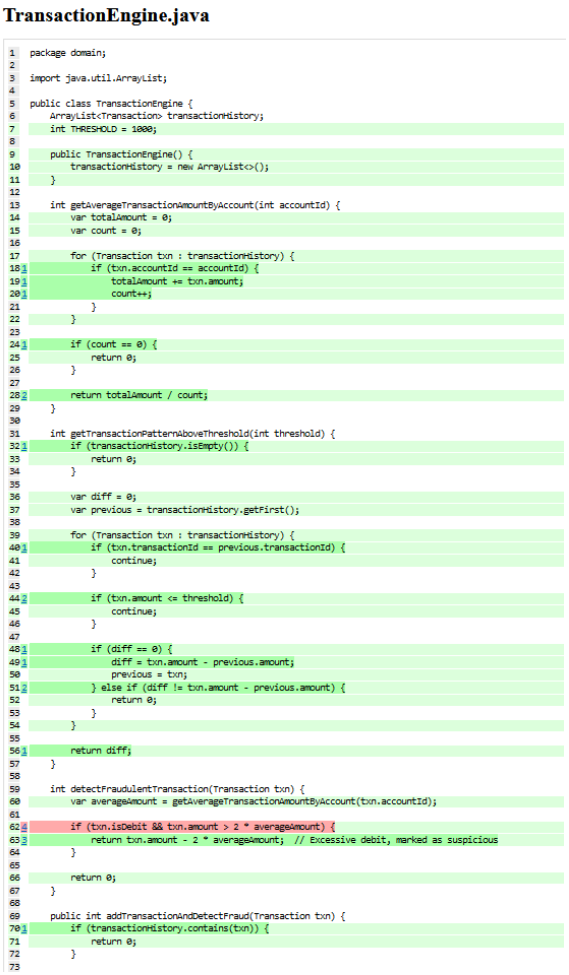
**Active mutators**

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

**Tests examined**

- domain.TransactionTest.[engine:junit-jupiter]/[class:domain.TransactionTest]/[method:testEqualsWitl
- domain.TransactionTest.[engine:junit-jupiter]/[class:domain.TransactionTest]/[method:testEqualsForl
- domain.TransactionTest.[engine:junit-jupiter]/[class:domain.TransactionTest]/[method:testEqualsOnl
- domain.TransactionTest.[engine:junit-jupiter]/[class:domain.TransactionTest]/[method:testEqualsForl
- domain.TransactionEngineTest.[engine:junit-jupiter]/[class:domain.TransactionEngineTest]/[method:
- domain.TransactionEngineTest.[engine:junit-jupiter]/[class:domain.TransactionEngineTest]/[method:
- domain.TransactionEngineTest.[engine:junit-jupiter]/[class:domain.TransactionEngineTest]/[method:

Report generated by PIT 1.17.2



**TransactionEngine.java**

```
1  package domain;
2
3  import java.util.ArrayList;
4
5  public class TransactionEngine {
6      ArrayList<Transaction> transactionHistory;
7      int THRESHOLD = 1000;
8
9      public TransactionEngine() {
10         transactionHistory = new ArrayList<>();
11     }
12
13     int getAverageTransactionAmountByAccount(int accountId) {
14         var totalAmount = 0;
15         var count = 0;
16
17         for (Transaction txn : transactionHistory) {
18 1           if (txn.accountId == accountId) {
19 1               totalAmount += txn.amount;
20 1               count++;
21            }
22        }
23
24 1       if (count == 0) {
25            return 0;
26        }
27
28 2       return totalAmount / count;
29     }
30
31     int getTransactionPatternAboveThreshold(int threshold) {
32 1       if (transactionHistory.isEmpty()) {
33            return 0;
34        }
35
36        var diff = 0;
37        var previous = transactionHistory.getFirst();
38
39        for (Transaction txn : transactionHistory) {
40 1           if (txn.transactionId == previous.transactionId) {
41                continue;
42            }
43
44 2           if (txn.amount <= threshold) {
45                continue;
46            }
47
48 1           if (diff == 0) {
49 1               diff = txn.amount - previous.amount;
50                previous = txn;
51 2           } else if (diff != txn.amount - previous.amount) {
52                return 0;
53            }
54        }
55
56 1       return diff;
57     }
58
59     int detectFraudulentTransaction(Transaction txn) {
60         var averageAmount = getAverageTransactionAmountByAccount(txn.accountId);
61
62 2       if (txn.isDebit && txn.amount > 2 * averageAmount) {
63 2           return txn.amount - 2 * averageAmount;  // Excessive debit, marked as suspicious
64        }
65
66        return 0;
67     }
68
69     public int addTransactionAndDetectFraud(Transaction txn) {
70 1       if (transactionHistory.contains(txn)) {
71            return 0;
72        }
73
```

```
56 1        return diff;
57      }
58
59      int detectFraudulentTransaction(Transaction txn) {
60          var averageAmount = getAverageTransactionAmountByAccount(txn.accountId);
61
62 4        if (txn.isDebit && txn.amount > 2 * averageAmount) {
63 3            return txn.amount - 2 * averageAmount;  // Excessive debit, marked as suspicious
64          }
65
66          return 0;
67      }
68
69      public int addTransactionAndDetectFraud(Transaction txn) {
70 1        if (transactionHistory.contains(txn)) {
71              return 0;
72          }
73
74          var fraudScore = detectFraudulentTransaction(txn);
75 1        if (fraudScore == 0) {
76              fraudScore = getTransactionPatternAboveThreshold(THRESHOLD);
77          }
78
79          transactionHistory.add(txn);
80 1        return fraudScore;
81      }
82  }
```

## 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 integer division with multiplication → KILLED<br>2. replaced int return with 0 for domain/TransactionEngine::getAverageTransactionAmountByAccount → KILLED |
| 32 | 1. negated conditional → KILLED |
| 40 | 1. negated conditional → KILLED |
| 44 | 1. negated conditional → KILLED<br>2. changed conditional boundary → KILLED |
| 48 | 1. negated conditional → KILLED |
| 49 | 1. Replaced integer subtraction with addition → KILLED |
| 51 | 1. negated conditional → KILLED<br>2. Replaced integer subtraction with addition → KILLED |
| 56 | 1. replaced int return with 0 for domain/TransactionEngine::getTransactionPatternAboveThreshold → KILLED |
| 62 | 1. Replaced integer multiplication with division → KILLED<br>2. negated conditional → KILLED<br>3. negated conditional → KILLED<br>4. changed conditional boundary → SURVIVED Covering tests |
| 63 | 1. Replaced integer subtraction with addition → KILLED<br>2. replaced int return with 0 for domain/TransactionEngine::detectFraudulentTransaction → KILLED<br>3. Replaced integer multiplication with division → KILLED |
| 70 | 1. negated conditional → KILLED |
| 75 | 1. negated conditional → KILLED |
| 80 | 1. replaced int return with 0 for domain/TransactionEngine::addTransactionAndDetectFraud → KILLED |

## Active mutators

- CONDITIONALS_BOUNDARY
- EMPTY_RETURNS
- FALSE_RETURNS
- INCREMENTS
- INVERT_NEGS
- MATH
- NEGATE_CONDITIONALS
- NULL_RETURNS
- PRIMITIVE_RETURNS
- TRUE_RETURNS
- VOID_METHOD_CALLS

## بخش دوم

در این بخش با استفاده از فایل yaml زیر کانفیگ رانر برای github action را انجام دادیم.

```yaml
name: Java CI with Maven CA5
on:
  push:
    branches: [ "main" ]
  pull_request:
    branches: [ "main" ]

jobs:
  build-and-test:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v4

    - name: Set up JDK 21
      uses: actions/setup-java@v4
      with:
        java-version: '21'
        distribution: 'oracle'
        cache: 'maven'

    - name: Build and Test with Maven
      run: mvn clean test -B -e -f ./Fesadyab/pom.xml
```

به این منظور این فایل در آدرس زیر قرار دادیم:

```
repo/
├── Fesadyab/
│   ├── pom.xml
│   └── src/
├── Mizdooni/
│   ├── pom.xml
│   └── src/
└── .github/
    └── workflows/
        └── maven.yml
```

در این کانفیگ بیان شده در صورتی که push یا merge ای روی برنچ main رخ داد با پیاده سازی محیط ubuntu در رانر و راه اندازی جاوا در آن دستور زیر برای اجرا تست ها ران شود.

```
mvn clean test -B -e -f ./Fesadyab/pom.xml
```

توجه شود برای اینکه بتوان با maven از junit بهره برد ما plugin زیر را در فایل **pom.xml** افزودیم:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>3.5.2</version>
    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.11.3</version>
        </dependency>
    </dependencies>
</plugin>
<plugin>
```

خروجی اجرا به صورت زیر است: