

شناسه آخرین کامیت

<https://github.com/Hadi-loo/Software-Testing-Course/commit/32f44a23a994f560490f78550595204e4f2fdc0e>

بخش اول

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 | 2 | 100% 49/49 | 97% 28/29 | 97% 28/29 |

Report generated by [PIT](#) 1.15.2

تعداد mutant های ساخته شده

با توجه به عکس بالا، تعداد mutant های ساخته شده برابر 29 می باشد.

تعداد mutant های کشته شده توسط آزمون های نوشته شده (killed ها)

تعداد mutant های کشته شده برابر 28 می باشد که در مقایسه با تعداد کل mutant ها، مقدار mutation coverage برابر 97% را به ما می دهد.

تعداد mutant های زنده مانده پس از آزمون ها (lived ها)

تعداد mutant های زنده مانده برابر 1 می باشد. در یکی از mutant ها، اپراتور $>$ به \geq تبدیل می شود. در صورتی که order.quantity برابر با averageOrderQuantity باشد، مقدار return شده داخل و خارج شرط با یکدیگر برابر است و هر دو 0 می باشند. به عبارتی خروجی در هر دو حالت یکسان بوده و equivalent mutant داریم؛ یعنی جهشی پیدا کردیم که راهی برای نوشتن تست برای آن نیست زیرا رفتار برنامه را تغییر نمی دهد.

```
58
59     int getCustomerFraudulentQuantity(Order order) {
60
61         var averageOrderQuantity = getAverageOrderQuantityByCustomer(order.customer);
62
63         if (order.quantity > averageOrderQuantity) {
64             return order.quantity - averageOrderQuantity;
65         }
66
67         return 0;
68     }
```

تاثیر mutation coverage بالا در میزان خطر refactoring

Refactoring به معنای بهبود ساختار کد بدون تغییر در عملکرد اصلی و رفتار خارجی آن می باشد. Refactoring تنها برای کد اصلی استفاده نمی شود و در تست نیز می توان از آن بهره برد. به بررسی هر یک می پردازیم:

- Refactoring بر روی کد اصلی: اگر تست های خوبی داشته باشیم، پس از هر تغییر کوچک در کد، می توانیم تست ها را اجرا کنیم و مطمئن باشیم رفتار آن تغییر نمی کند زیرا در صورت تغییر رفتار کد، تست ها fail می شوند.
- Refactoring بر روی تست: در اینجا مانند قبل نمی توانیم مطمئن باشیم که refactoring منجر به تغییر رفتار کد نمی شود زیرا تستی برای آن نداریم. برای اطمینان یافتن از اینکه رفتار تست در refactoring دچار تغییر نمی شود، از mutation testing استفاده می کنیم. اگر mutation coverage قبل از refactoring تست ها با بعد آن برابر باشد، این اطمینان را به ما می دهد که رفتار تست هایمان تغییر نکرده است.

تحلیل گزارش PIT

نتیجه کلی PIT به صورت زیر است. همانطور که مشخص است، در کل 29 عدد mutant داریم که 28 تا از آنها kill شده اند. تمامی mutant های کلاس Order که 8 تا بودند، kill شدند اما یکی از mutant های کلاس Engine همچنان زنده می باشد و تعداد کل mutant های این کلاس برابر 21 می باشد.

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 |

عکس زیر مربوط به کلاس Order است که تمامی mutant ها kill شده اند.

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 |
| 17 | 2. replaced boolean return with true for domain/Order::equals → KILLED |
| 19 | 1. replaced boolean return with true for domain/Order::equals → KILLED |

گزارش کار تمرین کامپیوتری پنجم آزمون نرم افزار

سنا ساری نوایی - 810199435

محمدهادی بابالو - 810199380

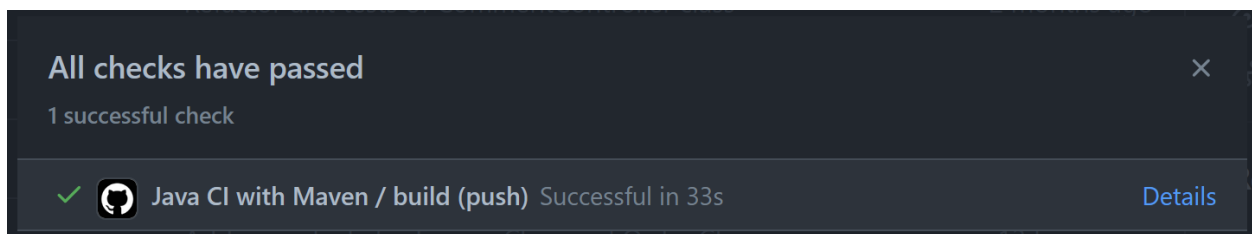
Mutation های کلاس Engine در آمده است. در بخش "تعداد mutant های زنده مانده پس از آزمون ها (lived ها)" به دلیل kill نشدن mutant مشخص شده پرداختیم.

Mutations

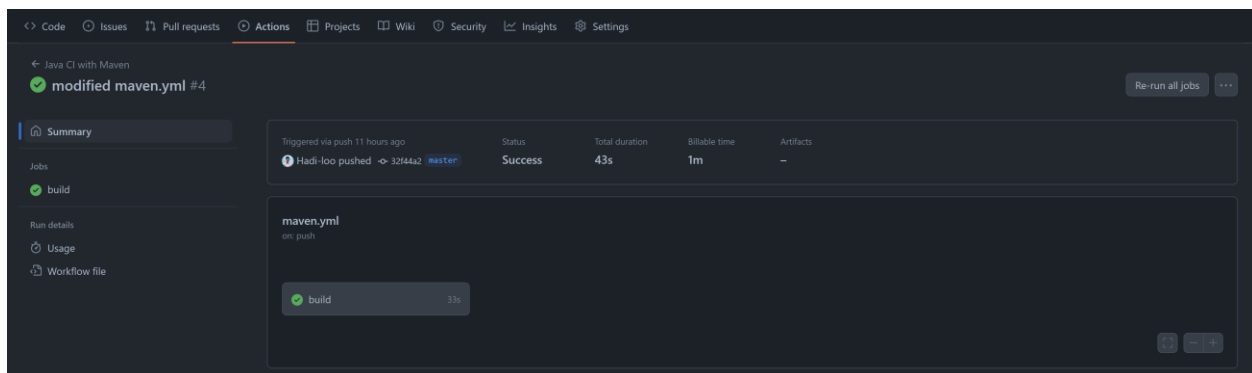
| | |
|----|--|
| 18 | 1. negated conditional → KILLED |
| 19 | 1. Replaced integer addition with subtraction → KILLED |
| 20 | 1. Changed increment from 1 to -1 → KILLED |
| 24 | 1. negated conditional → KILLED |
| 28 | 1. replaced int return with 0 for domain/Engine::getAverageOrderQuantityByCustomer → 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 |

بخش دوم

پس از اضافه کردن GitHub Pipeline با افزودن فایل maven.yml و پوش کردن آن می‌توانیم ببینیم که pipeline ما برای build کردن پروژه و اجرای تست‌های آن راه افتاده است و در حال کار کردن است.



همچنین می‌توانیم جزئیات اجرا شدن pipeline و ریز مراحل آن را هم مشاهده کنیم.



گزارش کار تمرین کامپیوتری پنجم آزمون نرم افزار

سنا ساری نوایی - 810199435

محمدهادی بابالو - 810199380

The screenshot shows the GitHub Actions interface for a workflow named 'Java CI with Maven'. The workflow file is 'modified maven.yml #4'. The 'build' job is highlighted, showing it succeeded 11 hours ago in 33s. The job steps are listed on the right:

- Set up job (2s)
- Run actions/checkout@v3 (1s)
- Set up JDK 19 (9s)
- Build with Maven (15s)
- Post Set up JDK 19 (3s)
- Post Run actions/checkout@v3 (0s)
- Complete job (0s)

The screenshot shows the logs for the 'Build with Maven' job. The logs indicate that the build was successful and that the tests passed. The output shows the following information:

```
762 [INFO] -----
763 [INFO] T E S T S
764 [INFO] -----
765 [INFO] Running domain.EngineTest
766 [INFO] Tests run: 18, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.186 s -- in domain.EngineTest
767 [INFO] Running domain.OrderTest
768 [INFO] Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.040 s -- in domain.OrderTest
769 [INFO]
770 [INFO] Results:
771 [INFO]
772 [INFO] Tests run: 25, Failures: 0, Errors: 0, Skipped: 0
773 [INFO]
774 [INFO]
775 [INFO] --- jacoco-maven-plugin:0.8.10:report (report) @ CAS ---
776 [INFO] Loading execution data file /home/runner/work/Software-Testing-Course/Software-Testing-Course/CAS/target/jacoco.exec
777 [INFO] Analyzed bundle 'CAS' with 2 classes
778 [INFO]
```

The screenshot shows the logs for the 'Post Set up JDK 19' job. The logs indicate that the build was successful and that the tests passed. The output shows the following information:

```
813 [INFO] -----
814 [INFO] BUILD SUCCESS
815 [INFO] -----
816 [INFO] Total time: 13.567 s
817 [INFO] Finished at: 2023-12-14T23:38:56Z
818 [INFO] -----
```