گزارش‌کار تمرین کامپیوتری پنجم آزمون نرم افزار

محمدهادی بابالو – 810199380 سنا ساری نوایی – 810199435

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	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 می‌باشد.

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

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

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

تعداد mutant های زنده مانده برابر 1 می‌باشد. در یکی از mutantها، اپراتور < به ≥ تبدیل می‌شود. در صورتی که order.quantity برابر با averageOrderQuantity باشد، مقدار return شده داخل و خارج شرط با یکدیگر برابر است و هردو 0 می‌باشند. 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 نمی‌دهد.

```
58
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      }
```

محمدهادی بابالو – 810199380 سنا ساری نوایی – 810199435

تاثیر mutation coverage بالا در میزان خطر refactoring

Refactoring به معنای بهبود ساختار کد بدون تغییر در عملکرد اصلی و رفتار خارجی آن می‌باشد. Refactoring تنها برای کد اصلی استفاده نمی‌شود و در تست نیز می‌توان از آن بهره برد. به بررسی هر یک می‌پردازیم:

- Refactoring بر روی کد اصلی: اگر تست‌های خوبی داشته باشیم، پس از هر تغییر کوچک در کد، می‌توانیم تست‌ها را اجرا کنیم و مطمئن باشیم رفتار آن تغییر نمی‌کند زیرا در صورت تغییر رفتار کد، تست‌ها fail می‌شوند.
- Refactoring بر روی تست: در اینجا مانند قبل نمی‌توانیم مطمئن باشیم که refactoring منجر به تغییر رفتار کد نمی‌شود زیرا تستی برای آن نداریم. برای اطمینان یافتن از اینکه رفتار تست در refactoring دچار تغییر نمی‌شود، از mutation testing استفاده می‌کنیم. اگر mutation coverage قبل از refactoring تست‌ها با بعد آن برابر باشد، این اطمینان را به ما می‌دهد که رفتار تست‌هایمان 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 می‌باشد و تعداد کل mutant های این کلاس برابر 21 می‌باشد.

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

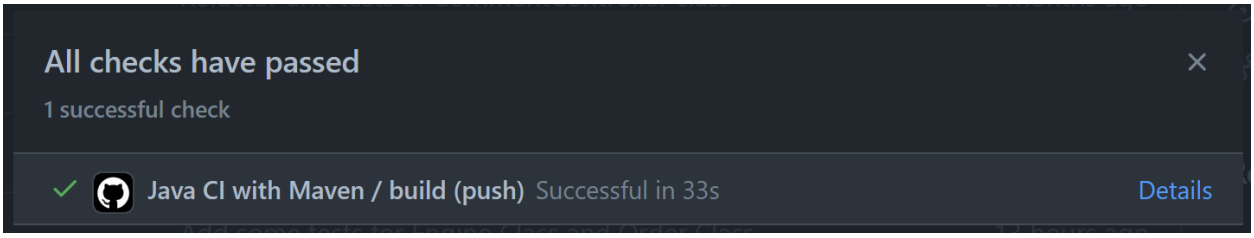
Mutation های کلاس Engine در آمده است. در بخش "تعداد mutant های زنده مانده پس از آزمون‌ها (lived ها)" به دلیل kill نشدن mutant مشخص شده پرداختیم.

Mutations

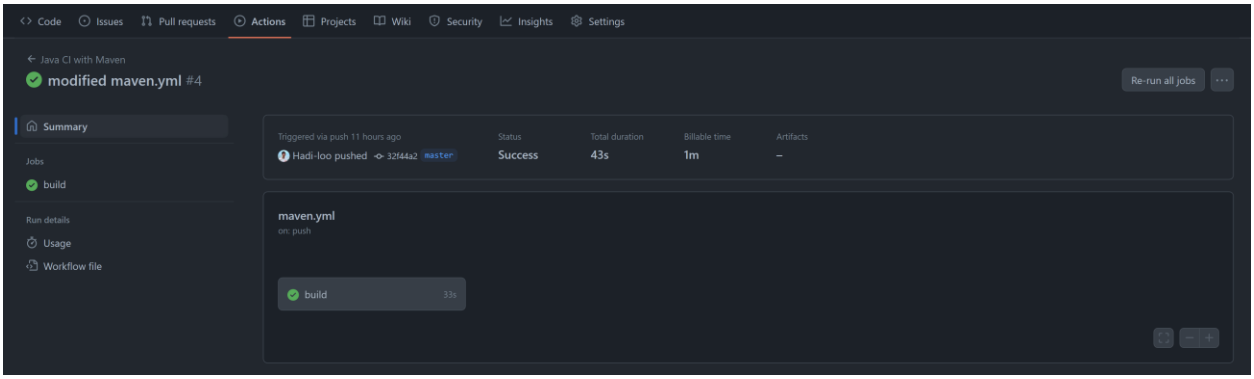
18	1. negated conditional → KILLED
19	1. Replaced integer addition with subtraction → KILLED
20	1. Changed increment from 1 to -1 → KILLED
24	1. negated conditional → KILLED
28	1. replaced int return with 0 for domain/Engine::getAverageOrderQuantityByCustomer → KILLED 2. Replaced integer division with 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.



گزارش‌کار تمرین کامپیوتری پنجم آزمون نرم افزار

محمدهادی بابالو – 810199380 سنا ساری نوایی – 810199435