به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

آزمون نرم افزار

گزارش کار شماره 5

محمد پویا افشاری (810198577)
مصطفی ابراهیمی (810199575)

شناسه

بخش اول

- **PIT**

PIT به کمک mutation operator یا mutator روی byte code (به source code تغییر اعمال نمی کند) ایجاد می‌کند. در نتیجه خروجی HTML تولید کرده که چه میزان تست کشته شده (در صورت تغییر کد شروط تست ما قادر به خطا گرفتن ان شده است) و چه تعداد زنده مانده است (یعنی تست نتوانسته آن را تشخیص بدهد).
همچنین No coverage یعنی مشابه Lived که تستی نبوده که آن بخش mutation ساخته شده را کاور کند.
Time out به معنی آن است که در infinite loop گیر کرده (مشابه حذف counter از لوپ)
Non viable به معنای آن است که به وسیله byte code لود JVM صورت نگرفته و PIT سعی می‌کند شمار این موارد رو حداقل و تست های کشته شده را حداکثر کند.

- **اضافه کردن PIT**

به فایل pom.xml قطعه کد زیر را اضافه می کنیم:

و با دستور زیر PIT را اجرا می کنیم:

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

- **گزارش PIT**

با اضافه کردن Mutation PIT به تست دو کلاس بر اساس گزارش دریافت شده توانستیم Mutation Coverage زیر را بدست بیاوریم:

Pit Test Coverage Report

Package Summary

domain

Number of Classes	Line Coverage		Mutation Coverage		Test Strength	
2	100%	51/51	97%	29/30	97%	29/30

Breakdown by Class

Name	Line Coverage		Mutation Coverage		Test Strength	
Engine.java	100%	42/42	95%	21/22	95%	21/22
Order.java	100%	9/9	100%	8/8	100%	8/8

Report generated by PIT 1.15.2

تست ها توانسته اند تمام خط ها رو پوشش دهند. همچنین از ۳۰ mutation ایجاد شده ۲۹ آن را کشته اند.

```
64      int getCustomerFraudulentQuantity(Order order) {
65
66          var averageOrderQuantity = getAverageOrderQuantityByCustomer(order.customer);
67
68 2        if (order.quantity > averageOrderQuantity) {
69 2            return order.quantity - averageOrderQuantity;
70          }
71
72          return 0;
73      }
74
75      public int addOrderAndGetFraudulentQuantity(Order order) {
76 1          if (orderHistory.contains(order)) {
77              return 0;
78          }
79
```

تنها mutation که زنده ماند را در تصویر بالا مشاهده میکند که کشتن آن به خاطر ماهیت کد امکان پذیر نیست.
با اعمال mutation علامت < تبدیل به =< میشود، که میخواهد حساسیت تست ها رو برروی شرایط مرزی چک کند.
در حالت عادی وقتی دو متغیر داخل if با هم برابر باشند، برنامه داخل if نمیرود و ۰ برمیگرداند.
در حالتی که mutation رخ دهد و دو متغیر با هم برابر باشند، برنامه داخل if میرود و تفاضل آنها که مساوری ۰ است را برمیگرداند.
چون خروجی ها یکسان شده است، تست پاس میشود و mutation کشته نمی شود.

خط سبز کم رنگ: توسط تست پوشش داده شده است.
خط سبز پر رنگ: توسط تست پوشش داده شده است و mutation آن کشته شده است.
خط قرمز کم رنگ: توسط تست پوشش داده نشده است.
خط قرمز پر رنگ: mutation آن کشته نشده است.

- **تاثیر Coverage mutation بالا بر خطر Refactoring**

به طور کلی میتوان گفت Coverage بالا در Mutation به این معنا خواهد بود که در طول زمان Refactoring تست های ما به سرعت خطا ها پیدا خواهند کرد. برعکس در صورتی که Coverage Mutation کم باشد یعنی در صورت تغییر کد احتمال دریافت نکردن آن توسط تست ها بالا خواهد بود.
حال Refactoring به معنای ان اگر باشد که در طول انجام این عمل خطای انسانی نداشته باشیم و در صورتی که تست های Mutation با Effectiveness بالا نوشته باشیم احتمالا به مشکل نخواهیم خورد ولی در صورتی که تست‌های نوشته شده کارایی بالا نداشته و صرفا Coverage داشته باشند با هر تغییر در Source که Logic را حفظ کند خطا را دریافت خواهیم کرد و احتمالا نیاز هست چندین تست را تغییر بدهیم.
پس در نتیجه Mutation Coverage بالا به طور کلی ریسک Refactoring اضافه نمی کند بلکه در صورتی که کارایی و دقت در هنگام اضافه کردن تست ها همراه باشد در هنگام Refactor چون منطق برنامه حفظ می‌شود درصوتی که تست های متناظر آن در صورت نیاز تغییر کند (تغییر نام متغییر ها و...) همان کارایی را حفظ خواهد کرد.

یک حالت منفی این خواهد بود که ما کد برنامه را Refactor کنیم ولی آن را در Test اعمال نکنیم. بنابر این نیاز است که بعد از ایجاد Mutant ها چک کنیم که به وسیله‌ی Class test متناظر کد پیدا می‌شوند یا خیر بنابر این به ابزار اتوماتیگ نیاز خواهیم داشت که Mutator های متناظر جدید را ایجاد کند (مثلا PIT):
لینک زیر که مقاله Uncle Bob است توضیح کامل تری می‌دهد.
http://blog.cleancoder.com/uncle-bob/2016/06/10/MutationTesting.html

بخش دوم