به نام او

شهنام فیضیان 810100197
عرفان دارایی 810100139

آزمون نرم‌افزار
گزارش بخش تئوری پروژه 4

آدرس مخزن https://github.com/ShahnamFeyzian/Software-Testing-Course

شناسه آخرین کامیت 51772107c9825ecda06207852f277e600b9bdb37

سوال اول

WebMvcTest

1. برای تست بخش‌های وب و کنترلرها در Spring استفاده می‌شود.
2. فقط لایه وب (Web Layer) برنامه، شامل کنترلرها، فیلترها و تنظیمات MVC را بالا می‌آورد.
3. وابستگی‌های دیگر مثل سرویس‌ها یا لایه داده (Repository) را بارگذاری نمی‌کند، مگر اینکه با mock مشخص شوند.
4. مناسب برای تست‌های سریع و متمرکز روی کنترلرها و درخواست‌های HTTP.
5. config کمتری دارد (فقط مربوط به لایه وب)

SpringBootTest

1. کل Application Context را بارگذاری می‌کند.
2. همه اجزای اپلیکیشن شامل سرویس‌ها، ریپازیتوری‌ها و سایر وابستگی‌ها را تست می‌کند.
3. مناسب برای تست‌های End-to-End یا تست کل سیستم.

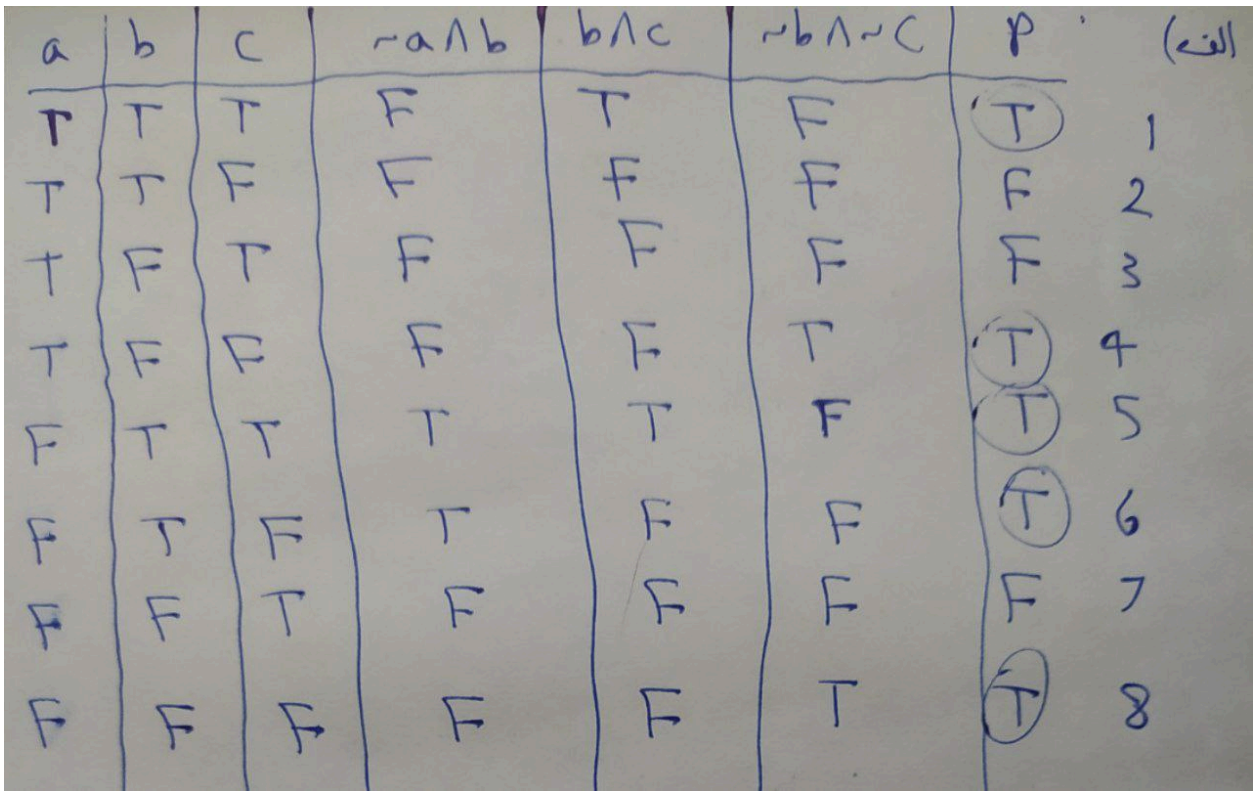
4. معمولاً زمان بیشتری برای اجرا نیاز دارد، چون تمام برنامه راه‌اندازی می‌شود.
5. شامل config های بیشتری است (کل پروژه)

خلاصه تفاوت:

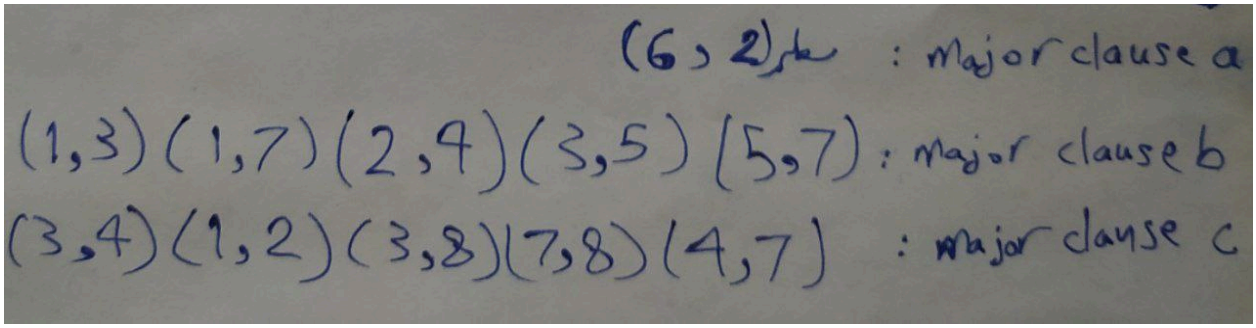
- WebMvcTest: فقط برای تست لایه وب و کنترلرها.
- SpringBootTest: برای تست کل برنامه و همه لایه‌ها.

سوال دوم

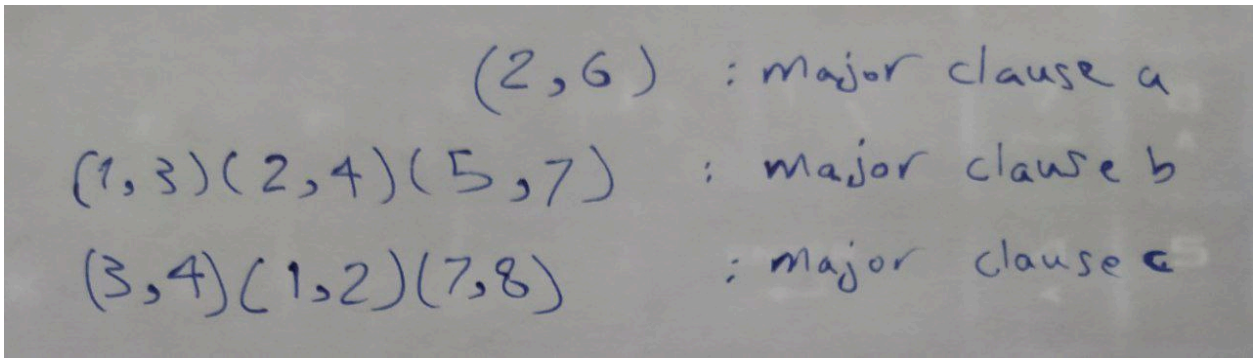
الف)



ب)

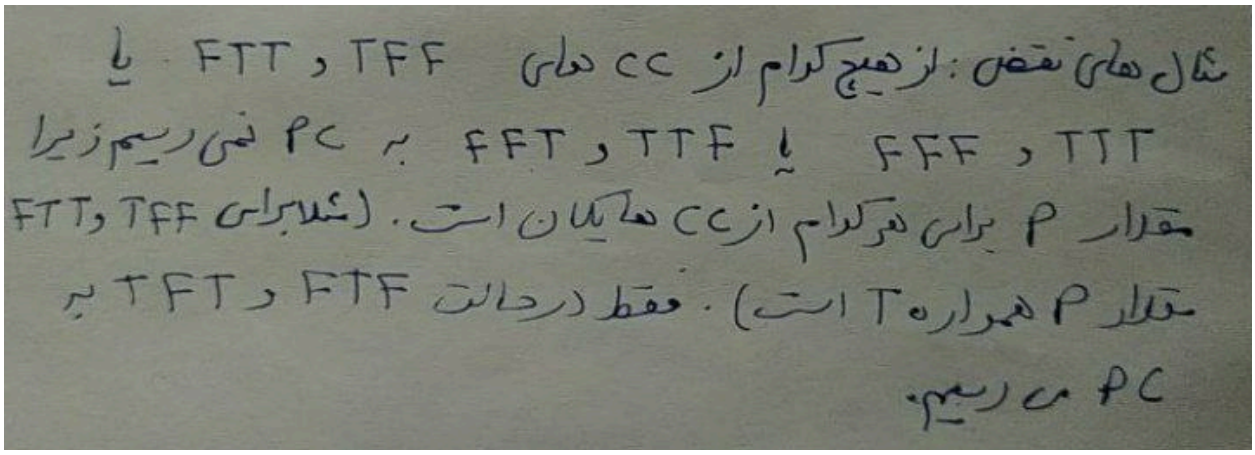


پ)



خیر بخش ب زیر مجموعه این بخش نیست و برعکس این بخش زیرمجموعه بخش ب است. زیرا در cacc ، در واقع minor clause ها می توانند با هم برابر باشند یا نباشند ، اما در RACC حتما باید با هم برابر باشند. پس RACC حالت های کمتری شامل می شود.

ت)



سوال سوم

برای ورودی های مختلف:

discountRate:

D1: مقدار منفی

D2: مقدار 0

D3: مقدار بین 0 و 1

D4: مقدار 1

D5: مقدار بزرگتر از یک

Price:

P1: مقدار منفی

P2: مقدار 0

P3: مقدار مثبت

minPurchase:

M1: مقدار منفی

M2: مقدار 0

M3: مقدار مثبت

شرط if بر اساس این مقادیر پوشش داده می شود. اما برای else if باید characteristic برای مقایسه Price و minPurchase داشته باشیم:

C1: minPurchase > price

C2: minPurchase <= price

پوشش های pair wise ممکن :(از نوشتن نا ممکن ها صرف نظر کردیم)

P1, D1, M1, C2	P1, D2, M2, C1	P1, D3, M3, C1	P1, D4, M1, C1	P1, D5, M1, C2
P2, D1, M2, C2	P2, D2, M3, C1	P2, D3, M1, C2	P2, D4, M2, C2	P2, D5, M2, C2
P3, D1, M3, C1	P3, D2, M1, C2	P3, D3, M2, C2	P3, D4 or D3 , M3, C2	P3, D5, M3, C1

یک حالت دیگر هم P3 , D3 or D4 , C3 , D1 که منجر به درست بودن else if میشود.

```
import org.junit.jupiter.api.Test;
import java.util.List;
import static org.junit.jupiter.api.Assertions.*;

public class CalculateDiscountedPriceTest {
    private final List<Double> P = List.of(-100.0 , 0.0, 200.0 , 150.0);
    private final List<Double> D = List.of(-0.4, 0.0, 0.5, 1.0, 1.6);
    private final List<Double> M = List.of(-300.0 , 0.0, 170.0);
    @Test
    void testInvalidInputsRanges() {
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(0), D.get(0),
M.get(0))); // P1 , D1 , M1
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(1), D.get(0),
M.get(1))); // P2 , D1 , M2
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(2), D.get(0),
M.get(2))); // P3 , D1 , M3
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(0), D.get(1),
M.get(1))); // P1 , D2 , M2
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(1), D.get(1),
M.get(2))); // P2 , D2 , M3
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(2), D.get(1),
M.get(0))); // P3 , D2 , M1
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(0), D.get(2),
```

```
M.get(2))); // P1 , D3 , M3
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(1), D.get(2),
M.get(0))); // P2 , D3 , M1
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(2), D.get(2),
M.get(1))); // P3 , D3 , M2
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(0), D.get(3),
M.get(0))); // P1 , D4 , M1
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(1), D.get(3),
M.get(1))); // P2 , D4 , M2
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(0), D.get(4),
M.get(0))); // P1 , D5 , M1
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(1), D.get(4),
M.get(1))); // P2 , D5 , M2
        assertEquals("Invalid input",
                DiscountCalculator.calculateDiscountedPrice(P.get(2), D.get(4),
M.get(2))); // P3 , D5 , M3
    }
    @Test
    void testValidInputsDiscountApplied(){
        assertEquals("100.0" ,
DiscountCalculator.calculateDiscountedPrice(P.get(2),
                D.get(2), M.get(2))); // p3, d3, m3, c2
        assertEquals("0.0" ,
DiscountCalculator.calculateDiscountedPrice(P.get(2),
                D.get(3), M.get(2))); // p3, d4, m3, c2
    }
    @Test
    void testValidInputsDiscountNotApplied(){
        assertEquals("150.0" ,
DiscountCalculator.calculateDiscountedPrice(P.get(3),
                D.get(2), M.get(2))); // p3, d3, m3, c1
    }
}
```