به نام خدا

دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر

آزمون نرم افزار

گزارش کار شماره 3

محمد پویا افشاري (810198577)
مصطفی ابراهیمی (810199575)

بخش دوم - گزارش کار

1. سوال اول:

```
public boolean equals(Object obj) {
    if (obj instanceof Order order) {
        return id == order.id;
    }
    return false;
}
```

بله این امکان وجود دارد که تست پوشش 100 درصدی بلاک/ statement داشته باشد اما پوشش شاخه‌ی کمتر از 100 درصد در این حالت داشته باشد. به این علت که اگر در آزمایه بالا statement ها را مشخص بکنیم داریم:

1. **if (obj instanceof Order order) {**
2. **return id == order.id;**

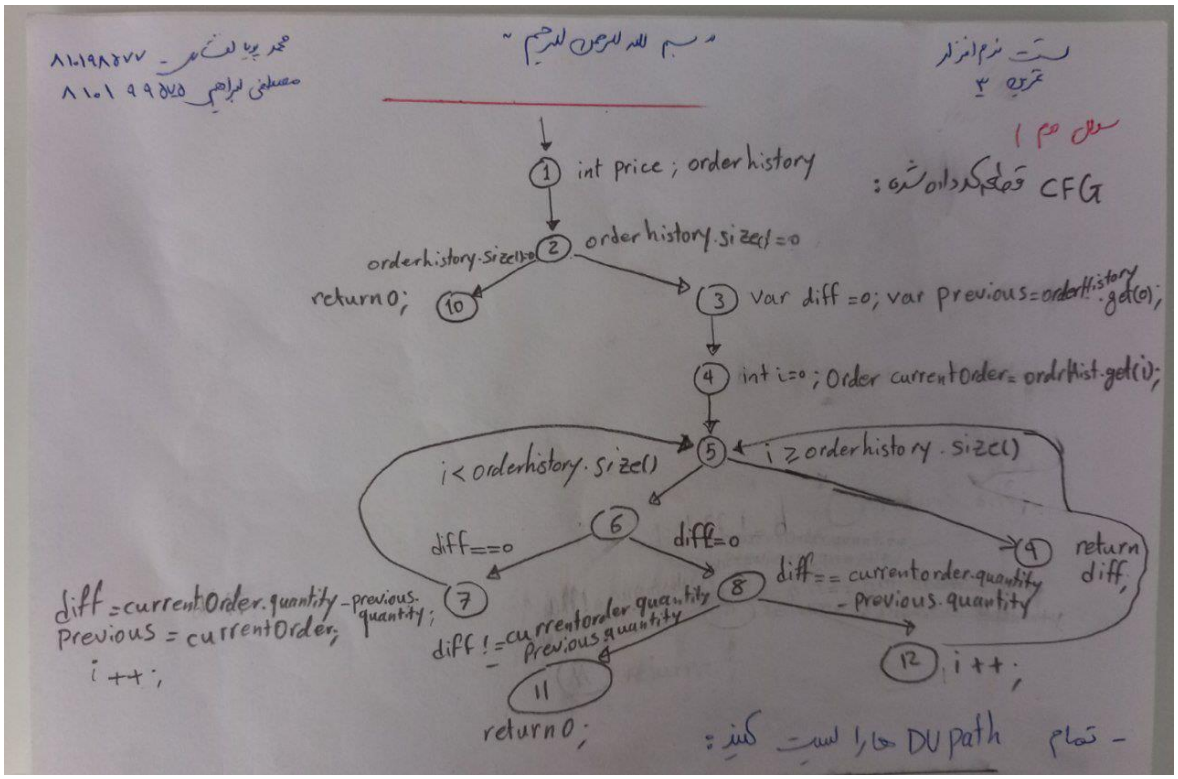
که این یعنی اگر آزمایه ای بنویسیم که صرفا obj به عنوان instance Order شناخته بشود شرط اجرا شده و خروجی true هم برگردانده می‌شود. در این حالت ما statement coverage صد درصد گرفته ایم اما برای دریافت branch coverage صد درصد باید شرط return false را هم بررسی میکردیم که در این حالت بررسی نشده است.

```
public boolean equals(Object obj) {
    var result = false;
    if (obj instanceof Order order) {
        result = id == order.id;
    }
    return result;
}
```

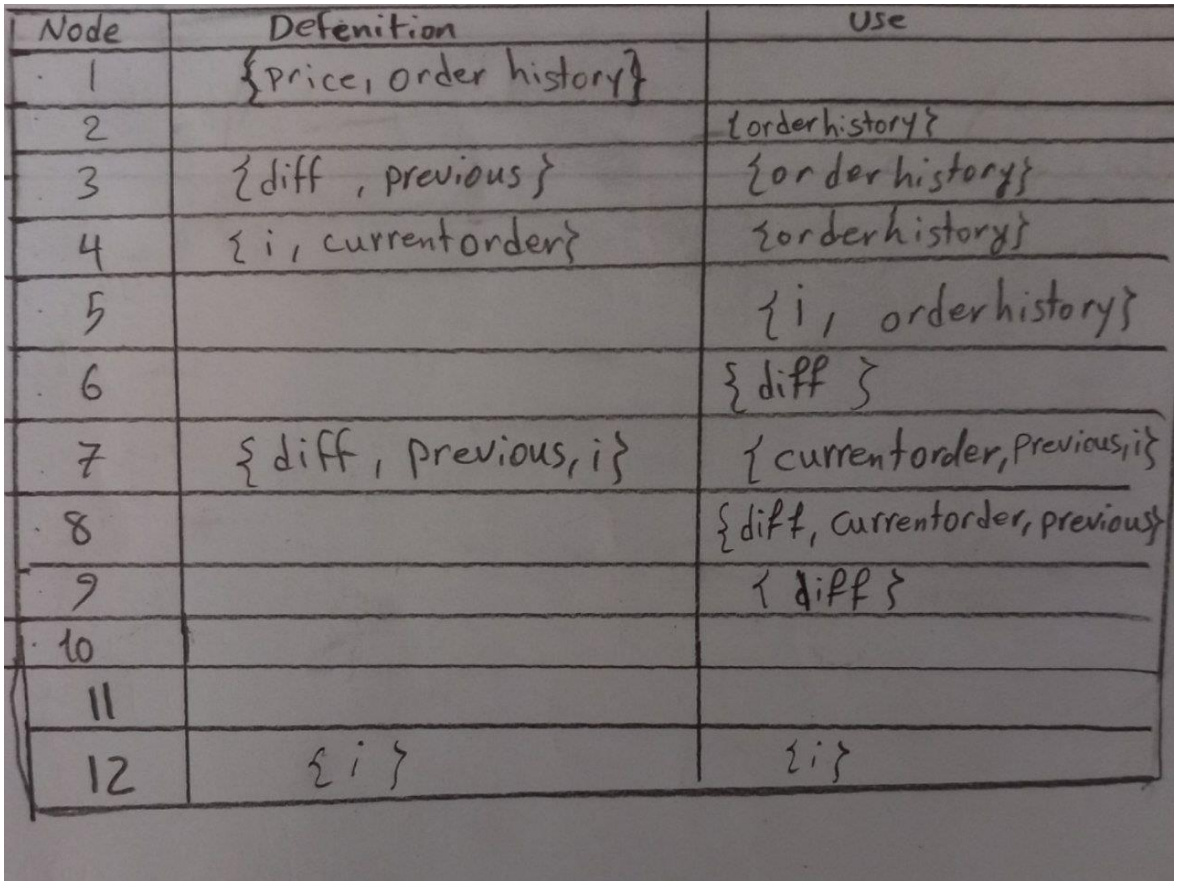
همانطور که گفته شد مشابه مثال قبل برای دریافت 100% branch coverage نیاز هست که هر دو سوی branch بررسی بشود اما در این حالت اگر سناریو در نظر بگیریم که obj از instance Order باشد. در نتیجه statement result مقدار true میگیرد و در نهایت خروجی میدهد که در این صورت به statement coverage صد درصد رسیده ایم اما باز هم branch coverage صد درصد نداشته ایم.

2. سوال دوم:

```
int getQuantityPatternByPrice(int price) {
    if (orderHistory.size() == 0) {
        return 0;
    }

    var diff = 0;
    var previous = orderHistory.get(0);

    for (Order currentOrder : orderHistory) {

        if (diff == 0) {
            diff = currentOrder.quantity - previous.quantity;
            previous = currentOrder;
        } else if (diff != currentOrder.quantity - previous.quantity) {
            return 0;
        }
    }

    return diff;
}
```



لیست همه ی DU Path ها:



Node	Defenition	Use
1	{price, order history}	
2		{orderhistory}
3	{diff , previous}	{order history}
4	{i, currentorder}	{orderhistory}
5		{i, orderhistory}
6		{diff}
7	{diff, previous, i}	{currentorder, previous, i}
8		{diff, currentorder, previous}
9		{diff}
10		
11		
12	{i}	{i}

Variable	DU Pairs	DU Path
Price		
orderHistory	(1,2) (1,3) (1,4) (1,5)	[1,2] [1,2,3] [1,2,3,4] [1,2,3,4,5]
Diff	(3,6) (3,8) (3,9) (7,8) (7,9)	[3,4,5,6] [3,4,5,6,8] [3,4,5,9] [7,5,6,8] [7,5,9]
Previous	(3,7) (3,8) (7,8)	[3,4,5,6,7] [3,4,5,6,8] [7,5,6,8]
I	(4,5) (4,7) (4,12)	[4,5] [4,5,6,7] [4,5,6,8,12]
currentOrder	(4,7) (4,8)	[4,5,6,7] [4,5,6,8]

لیست همه ی Prime Path ها:

- اگر * به این معنا باشد که:Cannot be extended b/c would contain illegal cycle
- اگر ! به این معنا باشد که:Cannot be extended b/c ends at terminal node
- در نهایت PP ها با هایلایت مشخص شده است که مجموعه ی تمام هایلایت شده ها می‌باشد

خواهیم داشت

Len 1:

1) [1]
2) [2]
3) [3]
4) [4]
5) [5]
6) [6]
7) [7]
8) [8]
9) [9]!
10) [10]!
11) [11]!
12) [12]

Len 2:

13) [1,2]
14) [2,10]!
15) [2,3]
16) [3,4]
17) [4,5]
18) [5,6]
19) [5,9]!
20) [6,7]
21) [6,8]
22) [7,5]
23) [8,11]!
24) [8,12]
25) [12,5]

Len 3:

26) [1,2,10]!
27) [1,2,3]
28) [2,3,4]
29) [3,4,5]
30) [4,5,9]!
31) [4,5,6]
32) [5,6,7]
33) [5,6,8]
34) [6,7,5]
35) [6,8,12]
36) [6,8,11]!
37) [8,12,5]
38) [7,5,6]

Len 4:

39) [1,2,3,4]
40) [2,3,4,5]
41) [3,4,5,9]!

42) [3,4,5,6]
43) [4,5,6,7]
44) [4,5,6,8]
45) [5,6,7,5]*
46) [5,6,8,12]
47) [5,6,8,11]!
48) [6,7,5,9]!
49) [6,7,5,6]*
50) [6,8,12,5]

Len 5:

51) [1,2,3,4,5]
52) [2,3,4,5,6]
53) [2,3,4,5,9]!
54) [3,4,5,6,7]
55) [3,4,5,6,8]
56) [4,5,6,8,11]!
57) [4,5,6,8,12]
58) [5,6,8,12,5]*
59) [6,8,12,5,6]*
60) [8,12,5,6,8]*
61) [12,5,6,8,12]*

Len 6:

62) [1,2,3,4,5,9]!
63) [1,2,3,4,5,6]
64) [2,3,4,5,6,7]
65) [2,3,4,5,6,8]
66) [3,4,5,6,8,11]!
67) [3,4,5,6,8,12]

Len 7:

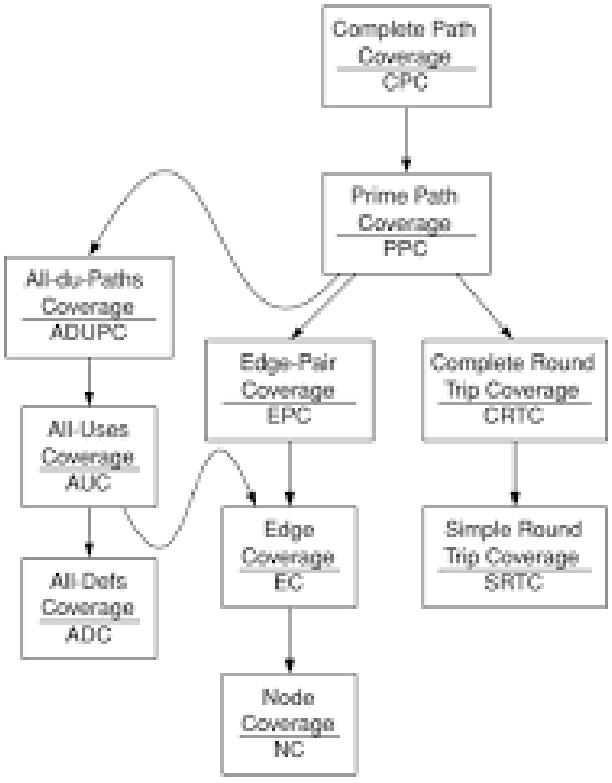
68) [1,2,3,4,5,6,7]
69) [1,2,3,4,5,6,8]
70) [2,3,4,5,6,8,11]!
71) [2,3,4,5,6,8,12]

Len 8:

72) [1,2,3,4,5,6,8,11]!
73) [1,2,3,4,5,6,8,12]

به این ترتیب در این حالت از مجموع بررسی تا طول 8 مسیر ها مجموعا 9 Prime Path بدست آمد که میتوان برای آنها تست Generate کرد.

3. سوال سوم:



بر اساس نتایج بدست آمده از کتاب Ammann & Offutt میتوان نتیجه گرفت که چون All-du-Paths Coverage یک Subsumption از Relation های Prime Path Coverage را شامل میشود پس میتوان تستی نوشت که در PPC باشد ولی در ADUPC نباشد.
برای مثال به ازای PP در سوال بالا تمام حلقه ها با شروع از نود های 5 6 8 12 هر یک یکبار حداقل طی شده که در مثال DU Path شاهد پیماش 8->12->5->6 نیستیم.

4. سوال چهارم:

- استفاده از روش PP در مسایل پیچیده میتواند منجر به ایجاد تعداد تست کیس زیادی شود که سر بار تست زیاد ایجاد میکند. همچنین که گاها بررسی همه مسیر ها ارایه شده در این روش شاید در برنامه غیر ممکن باشد به خصوص برای برنامه های بزرگتر
- گاها پیش بینی مسیر های PP میتواند زمان بر و هزینه بر تلفی شود. در این روش نیاز هست که به ازای هر path تست کیس نگه داری کنیم.
- در این روش نگه داری از کد های تست به صورت به روز کار سختی تلقی میشود. در صورت تغییر هر prime path باید تمامی متاثر از ان بروز بشود.
- استفاده از این روش لزوما تمامی باگ ها را پیدا نمی کند و گاها استفاده از روش های کم هزینه تر برای مثال EPC میتواند برخی اشکالات را در زمان کمتر پیدا کند.
- در صورتی که از نظر بودجه و سایر منابع کمبود داشته باشیم شاید استفاده از روش های جایگزین مثل DU Path coverage جایگزین بهتری باشد.