The Coding Challenge - Checkout

Thank you for performing this exercise to demonstrate your skills. This exercise is not about speed, completion or even solving a particularly difficult problem - it is more important to us to see how you approach a problem and structure a solution.

Invest not more than 4 hours for this exercise. Write into a readme file what still needs to be done or what you want to improve in the future. We are especially interested in a foresighted approach, which will be easy to customize and maintain in the future.

The exercise is a way to start a dialogue with you on how you solve problems.

Exercise

Implement the code for a checkout system with a pricing schema in a supermarket. This exercise *is not* about framework know-how, but in order to make reviewing easier for us, please use:

- Java
- Maven oder Gradle

for the implementation.

This exercise describes pricing schemes and requirements for enhanceability for a system that calculates the total cost of a shopping cart based on a set of pricing rules. The items in a supermarket are identified by using Stock Keeping Units or SKUs. For this, we will use individual letters of the alphabet (e.g. A,B,C) and the items are priced individually. In addition to that some items are multi priced: buy *n* of them and they will cost you *y* Euro.

For example: Item 'A' costs 40 Cent individually, but this week there is a special offer: buy three 'A's and they will cost you 1 Euro.

SKU	Unit Price	Special Price
A	40	3 for 100
B	50	2 for 80
C	25	
D	20	

The checkout should accept items in any order. E.g. so if we scan **B**, **A** and another **B**, it will recognize the two **B**'s and price them at 80 Cent, in order to result in a total price of 1,20 Euro.

Because the pricing can change frequently, we need to be able to pass in a set of pricing rules each time we start handling a checkout transaction.

The interface to the checkout should look like this:

```
var checkout = new CheckOut(pricing_rules);
checkout.scan(item);
checkout.scan(item);
price = checkout.total();
```

Here is a Unit test in Java. The helper method calculatePrice let's you specify a sequence of items (using a string), calling the checkout's scan method for each item before returning the total price:

```
public class TestPrice {

    public int calculatePrice(String goods) {
        CheckOut checkout = new CheckOut(rule);
        for(int i=0; i<goods.length(); i++) {
            checkout.scan(String.valueOf(goods.charAt(i)));
        }
        return checkout.total();
    }

    @Test
    public void totals() {
        assertEquals(0, calculatePrice(""));
        assertEquals(40, calculatePrice("A"));
        assertEquals(90, calculatePrice("AB"));
        assertEquals(135, calculatePrice("CDBA"));

        assertEquals(80, calculatePrice("AA"));
        assertEquals(100, calculatePrice("AAA"));
        assertEquals(140, calculatePrice("AAAA"));
        assertEquals(180, calculatePrice("AAAAA"));
        assertEquals(200, calculatePrice("AAAAAA"));

        assertEquals(150, calculatePrice("AAAB"));
        assertEquals(180, calculatePrice("AAABB"));
        assertEquals(200, calculatePrice("AAABBD"));
        assertEquals(200, calculatePrice("DABABA"));
    }

    @Test
    public void incremental() {
        CheckOut checkout = new CheckOut(rule);
```

```
        assertEquals(0, checkout.total);
        checkout.scan("A"); assertEquals(40, checkout.total);
        checkout.scan("B"); assertEquals(90, checkout.total);
        checkout.scan("A"); assertEquals(130, checkout.total);
        checkout.scan("A"); assertEquals(150, checkout.total);
        checkout.scan("B"); assertEquals(180, checkout.total);

    }
}
```

The exercise doesn't mention the format of the pricing rules:

- How can these be specified in such a way that the checkout doesn't know about particular items and their pricing strategies?
- How can we make the design flexible enough so that we can add new styles of pricing rules in the future?

Next steps

Send us a .zip file with your source code. After we have reviewed it, you will be informed if or when we invite you to the next interview. In this next interview, we are going to talk about your solution in the format of a code review.