2 Grocery coupons

2.1 Introduction

Most grocery stores offer discount coupons. Coupons apply on specific items, and shoppers can apply them to their purchase.

Each type of grocery coupon applies to an item (represented as a simple string) and offers a way to get a description of that coupon.

In order to entice customers and appear competitive, grocery stores offer three types of coupons:

1. “Amount off”: an example of this type of coupon is “Get $2 off a box of napkins”. This type of coupon can be applied per quantity of item. For example, if a customer buys 5 boxes of napkins, they would get $10 off using one such coupon. The description of this coupon would be *“$2 off item box of napkins”*.
2. “Percent off”: this is a typical “sale” coupon. An example of this type of coupon is “Get 20% off a box of napkins”. Much like the above, this coupon can be applied per quantity of item. For example, if a customer buys 5 boxes of napkins priced at $5 each, they would get 20% off on each box costing them a total of $20. The description of this coupon would be *“20% off item box of napkins”*
3. “Buy 2 get 1 free”: this type of coupon entices customers to buy more of an item. The minimum quantity to be bought, and the number of free items can be customized for the coupon. An example of this type of coupon is “Buy 3 boxes of napkins, get 2 free”. Much like the above, this coupon can be applied multiple times. For example, if a customer buys 9 boxes of napkins and applies this coupon, they would pay the price for only 6 boxes. The description of this coupon would be *“Buy 3 get 2 free for item box of napkins”*.

Accordingly each coupon offers operations to get its description and obtain the discounted price given a quantity of items and an original price per unit item. It must be ensured that a customer cannot earn money by choosing to buy an item and applying coupons towards the purchase.

2.2 Stacking of coupons

Stores often allow coupons to be stacked (i.e. combined). The stacking rules for these coupons are:

* A coupon can be stackable or not. Two coupons can be stacked only if both of them are stackable.
* Two coupons can be stacked only if they apply to the same item.
* Coupons of different types cannot be stacked.
* **Two “amount-off” coupons are stacked to provide a new “amount-off” coupon whose amount discount is the sum of both discounts.**
* **Two “percent-off” coupons are stacked to provide a new “percent-off” coupon whose percentage discount is the sum of the two discounts.**
* **Two “buy x get y free” coupons can be stacked to provide a “buy x get y free” coupon. This coupon is one of the two coupons to be stacked. If the first coupon offers an equal or better discount (ratio of x and y) than the second coupon then it is chosen, else the other is chosen.**

Accordingly each coupon should offer operations that support the stacking feature.

2.3 What to do

Design the data for the above in a way that captures their similarities and accurately represents the relevant data. Create interfaces/classes as you see fit and write appropriate constructors that allows one to create a question as specified above.

2.3.1 Tests

Write tests for all classes (create empty implementations, then write tests and finally fill the implementations). Read the test notes on the course web page to help you design tests effectively.

2.3.2 Documentation

We expect your code to be well-commented. The expectations are:

* Each interface and class contains a comment above it explaining specifically what it represents. This should be in plain language, understandable by anybody wishing to use it. Comment above a class should be specific: it should not merely state that it is an implementation of a particular interface.
* Each public method in the interface should have information about what this method accomplishes (purpose), the nature and explanation of any arguments, return values and exceptions thrown by it and whether it changes the calling object in any way (contract).
* If a class implements a method declared in an interface that it implements, **and** the comments in the interface describe this implementation completely and accurately, there is no need to replicate that documentation in the class.
* All comments should be in Javadoc-style.

2.4 Grading criteria

* How appropriately you capture the data and operations in your design of classes, interfaces and methods
* Correctness of the code
* Quality and coverage of your tests
* Quality of documentation. Since you will be designing everything, make sure your documentation is enough for others to understand (your peers, the graders, etc.)
* Code style

As the design is completely up to you, this assignment will not have an automated test grader. You have only your tests to verify the correctness of your implementation. We will verify that you have written them.