Notes about the RBC assignment

Yibing Li

1. The solution doesn't contain a rich set of functionalities. It is based on the information given. It is possible to implement a lot more functionalities such as applying various discount strategies for different products. However, following the usual agile practice of only implementing things when you actually need it, given they were not mentioned in the requirements, I didn't include those 'extra functionalities' here.

2. If I were to consider more complex strategies for calculating prices, for example, take into consideration of various discounts to different products to get the total prices, then I can design a more sophisticated solution by using the **Visitor Pattern**:

a. Each product will be modelled as a separate class: Apply, Orange...;

b. In each class, implement a method: **accept(BasketVisitor)** which will call the corresponding **visit(Product)** method in the **BasketVisitor** class(see description below) to calculate the price**.**

c.Define a class **BasetVisitor,** in which a set of visit methods (e.g. **visit(Apple); visit(Orange)...** are implemented. Each method applies relevant logic (e.g. discount strategy) to calculate the price and return the result.