Software Systems Analysis & Design
Fall 2021
Assignment 2
Team 32

Vladimir Makharev, Artem Batalov, Kirill Kuznetsov
Murat Shaikhutdinov, Pavel Baharuev
B20-01, B20-01, B20-04, B20-05, B20-01
Innopolis University

November 4, 2021

Contents

1 Report — Supermarket Billing Software **2**
1.1 Requirements 2
1.2 Solution Description 2
1.3 UML Class Diagram 3

1 Report — Supermarket Billing Software

1.1 Requirements

Description

Most of the supermarkets use software for preparing the final bill before the goods are packed. This software itself can very easily be designed by someone starting in Java (Note that most of the actual software used for billing in supermarkets use Java itself). This billing software sums the amount for individual items and then adds them up to get the final sum that the customer has to pay. The number of individual items, the price of each item should be editable for the user. Also, there should be an option to remove the item from the list altogether.

Design patterns

- Adapter
- Bridge
- Composite
- **Decorator – chosen**
- Flyweight

Technical requirements

Programming language: *Java*, as usual :)

1.2 Solution Description

Source code: GitHub Repository (follows the Google Java Style Convention)

Why do we use *Decorator* design pattern?

We use the decorator design pattern for changing the product properties easier. It is beneficial for us because we have a lot of cases when the product properties can alter. Usually, it applies to specific products. Hence, the decorator is the most suitable pattern for this purpose.

Code semantics

Our code imitates the work of a supermarket cash register billing. We have a customer, cashier (cash register), and shopping cart. Customers can put products from the product list into their shopping cart or take them out of it. When the customer has filled the shopping cart, they create an order and push it to the cashier. The cashier can edit the parameters of the order, for example, change product names, prices, and amount. For doing it, the cashier uses the key. Finally, the cash register can calculate the sum of the products inside the cart (offer a bill).

Code class relationships explanation

We have classes for the customers, product cart, cash register, products, and decorators. For each customer, we have one shopping cart. In this cart, we can add and remove products. We use the cart by operating it within the customer (only the customer sends the request to the shopping cart). When the order is ready, we push it to the cash register. The cashier can edit the parameters of the product list using a key. We check the key in the product class. Customers can interact with products only by cloning them to a cart (prototype design pattern from assignment 1). However, the cashier can edit the parameters of the products using a unique key to change protected fields of the product class. Another feature: if we need to apply the evening discount for the group of products or decrease the price for rotten products, the decorator pattern will solve the problem.

1.3 UML Class Diagram

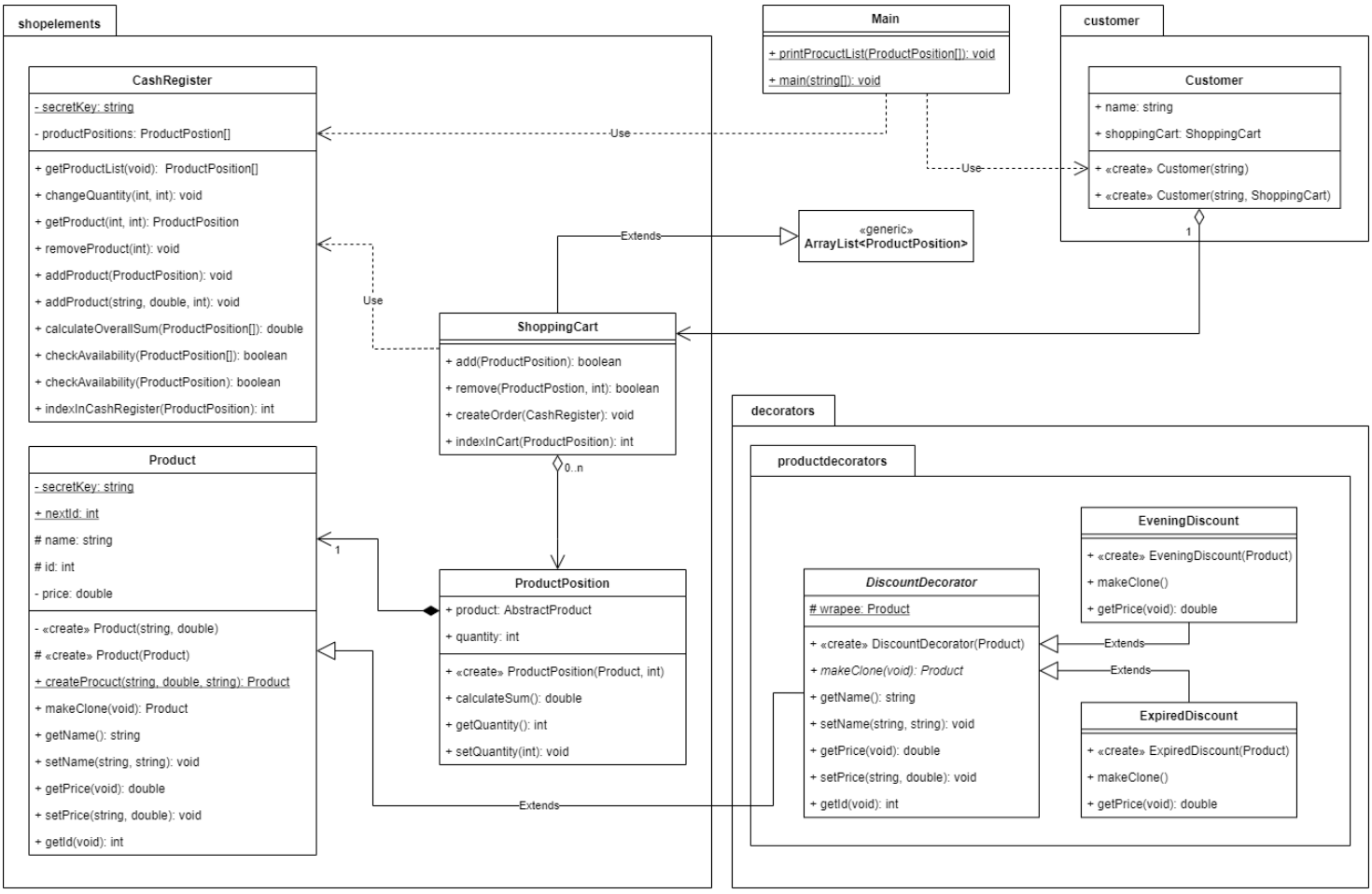

Figure 1: The UML class diagram of Supermarket Billing Software.