# Assignment. Shop

The advance in E-commerce has made online-shopping a pervasive and convenient daily experience. Customers could browse different online stores, each of which provides a list of products for customers to buy.

In this question, you'll build three classes, `Product`, `Store`, and `Customer`, to implement the online shopping process. See below for detailed requirements. Note that to implement the required functionality, you are free to add variables and methods that are not mentioned in this document; however, please **DO NOT** change the definitions of the existing variables and methods that we have specified in this document.

## 2.1 Product

**Attributes**

```java
private static int cnt; // initialized to 0, and will increase by 1 when the
constructor is called.
private int id;  // unique for each product and the value is set to cnt.
private String name;
private float price;
private ArrayList<Integer> ratings; // ratings from different customers; default
is empty.
```

**Constructor**

```java
public Product(String name, float price);
```

The constructor of the `Product` class.

- The `id` of the first product is 1. The given `price` is always valid.

**Methods**

```java
public boolean setRating(int rating);
public float getAvgRating();
public String toString();
```

`public boolean setRating(int rating);`

Set `rating` to the product. The `rating` would be added to this product's rating list `ratings`.

- A `rating` should be within the range [1,5]; in other words, there are only 5 possible values for `rating`. If `rating` is not in this range, do not add it into `rating` and return `false`.

```
public float getAvgRating();
```

Return the average rating of this product, which is computed as the average of all the ratings it has received so far.

```
public String toString();
```

Return a string description of this product, in the format of `Product ID id, name, RMB price, Rating average-rating`, e.g., `Product ID 12345, Laptop, RMB 10000.00, Rating 4.5`.

- Round `price` to 2 decimal places and `rating` to 1 decimal place.

## 2.2 Store

**Attributes**

```
private static int cnt; // initialized to 0, and will increase by 1 when the
constructor is called.
private int id;  // unique for each store and the value is set to cnt.
private String name;
private ArrayList<Product> productList;
private float income;
```

**Constructors**

```
public Store(String name);
public Store(String name, ArrayList<Product> productList, float income);
```

There are two constructors of the `Store` class. One for constructing a new store, with `income = 0` and nothing in the `productList`. The other constructs an existing store with given `income` and `productList`.

- The `id` of the first store is 1. The given `income` and `productList` are always valid.

**Methods**

```
public boolean hasProduct(Product product);
public boolean addProduct(Product product);
public boolean removeProduct(Product product);
public ArrayList<Product> getProductList();
public void transact(Product product, int method);
```

```
public boolean hasProduct(Product product);
```

A method to determine whether this store has the given `product` . Return `true` if the `product` is in the `productList` of the store; otherwise, return `false`.

```
public boolean addProduct(Product product);
```

Add product to the productList; return a boolean indicating whether the operation succeeds.

- Suppose each product, which is uniquely identified by its id, could only appear once in the productList of a particular store. That is, the same product will not appear in the productList of multiple stores (this kind of invalid case will not happen); and in the same productList, a product appears not more than once.
- If a product already exists in productList, return false and productList remains the same; otherwise, add product to productList and return true.

```
public boolean removeProduct(Product product);
```

Remove product from productList; return a boolean indicating whether the operation succeeds.

- If product exists in productList, remove it from productList and return true; otherwise, return false and productList remains the same.

```
public ArrayList<Product> getProductList();
```

Return productList.

```
public void transact(Product product, int method);
```

This is an interface method for stores to handle customers' purchases or refunds. Suppose that all the arguments are valid here.

- method = 0 means purchasing the product from this store. The product should be removed from the productList and the income of this store should increase by an amount equal to the price of the product.
- (Bonus) method = 1 means refunding the product to the store. The productList and income of the store should also be updated accordingly (suppose that the store adds this product back to its productList and could re-sell this product).

## 2.3 Customer

**Attributes**

```
private static int cnt; // initialized to 0, and will increase by 1 when the
constructor is called.
private int id;  // unique for each customer and the value is set to cnt.
private String name;
private ArrayList<Product> shoppingCart; // The list of purchased products;
default is empty.
private float wallet;
```

**Constructor**

```
public Customer(String name, float wallet);
```

The constructor of the `Customer` class.

- The `id` of the first customer is 1. The given `wallet` is always valid.

**Methods**

```
public boolean rateProduct(Product product, int rating);
public void updateWallet(float amount);
public boolean purchaseProduct(Store store, Product product);
public void viewShoppingCart(SortBy sortMethod);
public boolean refundProduct(Product product);
```

`public boolean rateProduct(Product product, int rating);`

A customer can rate a `product` using this method.

- For invalid `rating`, return `false`; otherwise return `true`.

`public void updateWallet(float amount);`

Update the `wallet` of this customer. The `amount` could be positive (gaining money) or negative (consuming money). Assume that arguments are always valid.

`public boolean purchaseProduct(Store store, Product product);`

Purchase `product` from `store`.

- Return `true` if the `store` has this `product` and the customer has enough money in the `wallet` to purchase this `product`; return `false` otherwise. Note that the `shoppingCart` of this customer as well as his/her `wallet` should be updated accordingly.

`public void viewShoppingCart(SortBy sortMethod);`

Display the purchased products in the `shoppingCart` of this customer. The order of displaying is specified by `sortMethod`. Below we provide an Enum `SortBy`, which says that sorting could be performed by the `PurchaseTime`, `Rating`, or the `Price` of products.

2.4 SortBy (Enum)

Attributes

```
public enum SortBy {
    PurchaseTime, Rating, Price
}
```

Suppose a customer Alice has purchased a few products from different stores.

```
Customer alice = new Customer("Alice", 20000);
// code for creating stores and products are ommitted
alice.purchaseProduct(store1, product_laptop);
alice.purchaseProduct(store1, product_table);
alice.purchaseProduct(store2, product_mouse);
alice.purchaseProduct(store3, product_phone);
```

Then, calling

```
alice.viewShoppingCart(SortBy.PurchaseTime);
```

will display (see `Product.toString()` for the format)

```
Product ID 2, Laptop, RMB 10000.00, Rating 4.5
Product ID 4, Table, RMB 300.00, Rating 4.3
Product ID 3, Mouse, RMB 100.00, Rating 3.0
Product ID 1, Phone, RMB 7000.00, Rating 4.5
```

Calling

```
alice.viewShoppingCart(SortBy.Rating);
```

will display

```
Product ID 3, Mouse, RMB 100.00, Rating 3.0
Product ID 4, Table, RMB 300.00, Rating 4.3
Product ID 2, Laptop, RMB 10000.00, Rating 4.5
Product ID 1, Phone, RMB 7000.00, Rating 4.5
```

- If the average rating is different without rounding but the same after rounding to one decimal place, then sort by the actual average rating (without rounding).
- If products have exactly the same average rating, they should be sorted by the purchase time.

Calling

```
alice.viewShoppingCart(SortBy.Price);
```

will display

```
Product ID 3, Mouse, RMB 100.00, Rating 3.0
Product ID 4, Table, RMB 300.00, Rating 4.3
Product ID 1, Phone, RMB 7000.00, Rating 4.5
Product ID 2, Laptop, RMB 10000.00, Rating 4.5
```

- If products have the same price, they should be sorted by the purchase time.

`public boolean refundProduct(Product product);`

**(Bonus)**: Return the `product` to the store where it was sold and get the money back. Return `true` if this customer has indeed purchased this product before and `false` otherwise. Note that the `shoppingCart` and `wallet` of this customer should be updated accordingly. In addition, the corresponding store should enable the refund process to update the `productList` and `income`.