Assignment #3

2019006171 김은민

운영체제 Windows 10

이클립스 버전 4.18.0.v20201202-1800 Build id: I20201202-1800

자바 버전 java 15.0.2 2021-01-19

<<interface>> Payable

+ pay (int amount) : void

Cash

- Currency: String
- Amount: int
- + pay (int): void
- + Cash(String, int)
- + getCurrency (): String
- + getAmount(): int
- + toString(): String

Credit

- Bank : String
- Limit: int
- amountUsed : int
- + pay (int): void
- + Credit (String, int, int)
- + getBank() : String
- + getLimit(): int
- + getAmountUsed (): int
- + toString(): String

Product

- Name : String
- Price : int
- Quantity: int
- waitList : ArrayList < Customer >
- + Product (String, int, int)
- + getQuantity (): int
- + getPrice(): int
- + getName (): String
- + addQuantity(int) : void
- + setQuantity(int) : void
- + wait(Customer) : void
- + getWaitList () : ArrayList < Customer >

Food

- expirationDateTime : LocalDateTime
- + isExpired(LocalDateTime) : Boolean
- + Food(String, int, int, int, int, int, int)
- + getExpirationDateTime() : LocalDateTime
- + setExpirationDateTime(LocalDateTime) : void

Manufactured

- Brand : String
- + Manufactured (String, int, String, int)
- + getBrand() : String

InventoryManager

- Observers : ArrayList < Observer >
- + addObserver (Observer) : void
- + deleteObserver (Observer) : void
- + notifyObservers() : void

Mart

- salesList : ArrayList < Product >
- <u>transactionNum</u>: int
- <u>uniqueInstance</u>: Mart
- Mart ()
- + getInstance() : Mart
- + getSalesList() : ArrayList < Product >
- + setSalesList (int) : void
- + printSalesList(): void
- + addInventory(int, int) : void
- + printExpired (): void
- + updateExpired(int) : void
- + makeReceipt(Customer, int) : void

<<Interface>>
Observer

+ update(InventoryManager) : void

Customer

- Name: String
- Wallet : ArrayList < Payable >
- shoppingCart : ArrayList < Product >
- + Customer(String, int)
- + getWallet(): ArrayList < Payable >
- + getName : String
- + getShoppingCart(): ArrayList < Product >
- + update(InventoryManager) : void
- + purchase (Product, int) : void
- + printShoppingCart() : void
- + getTotalPrice(): int
- + printWallet(): void
- + pay (int): void

InvalidAccessException

- + InvalidAccessException()
- + InvalidAccessException (String)

Exception

...

•••

NotEnoughBalanceException

- + NotEnoughBalanceException ()
- + NotEnoughBalanceException (String)

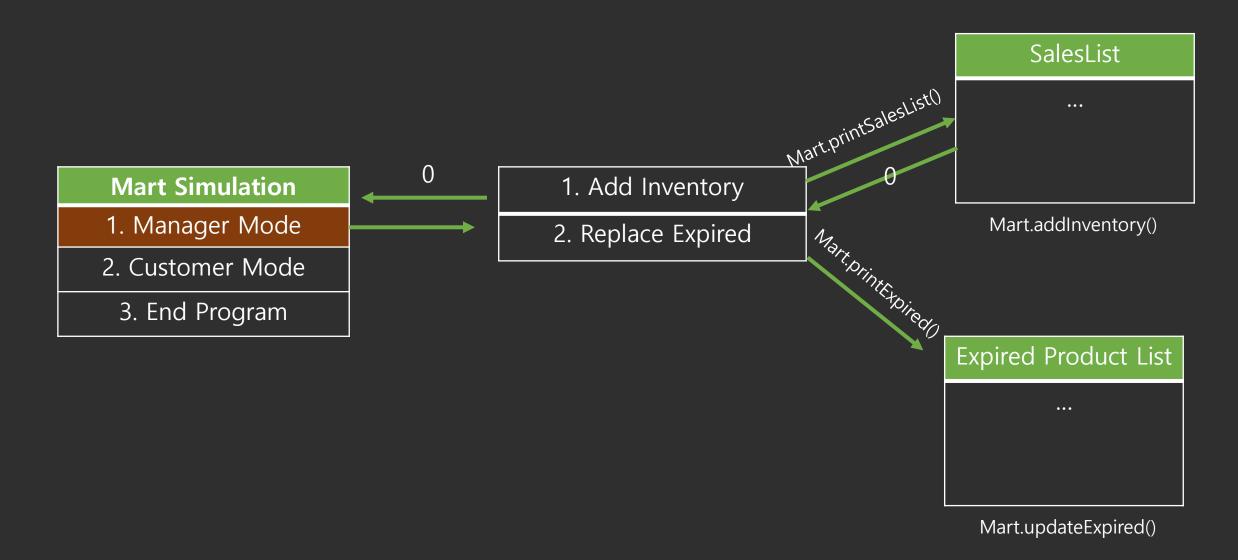
ExpiredProductException

- + ExpiredProductException ()
- + ExpiredProductException (String)

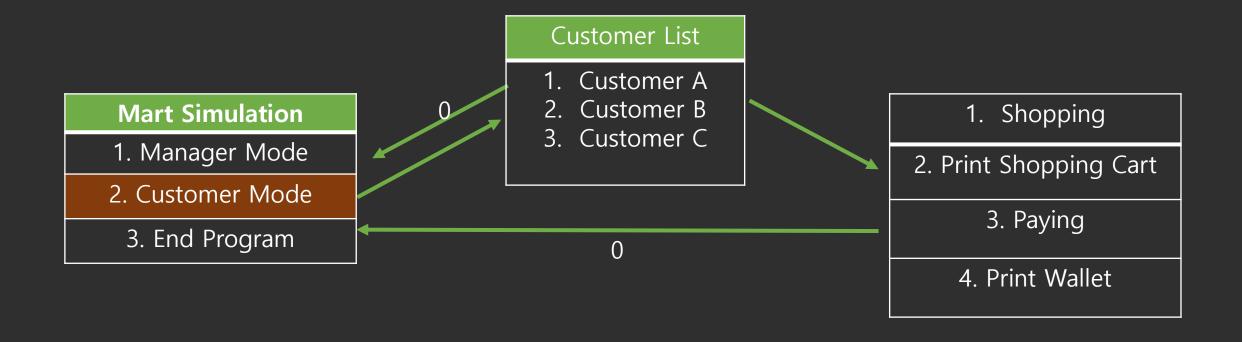
Execution

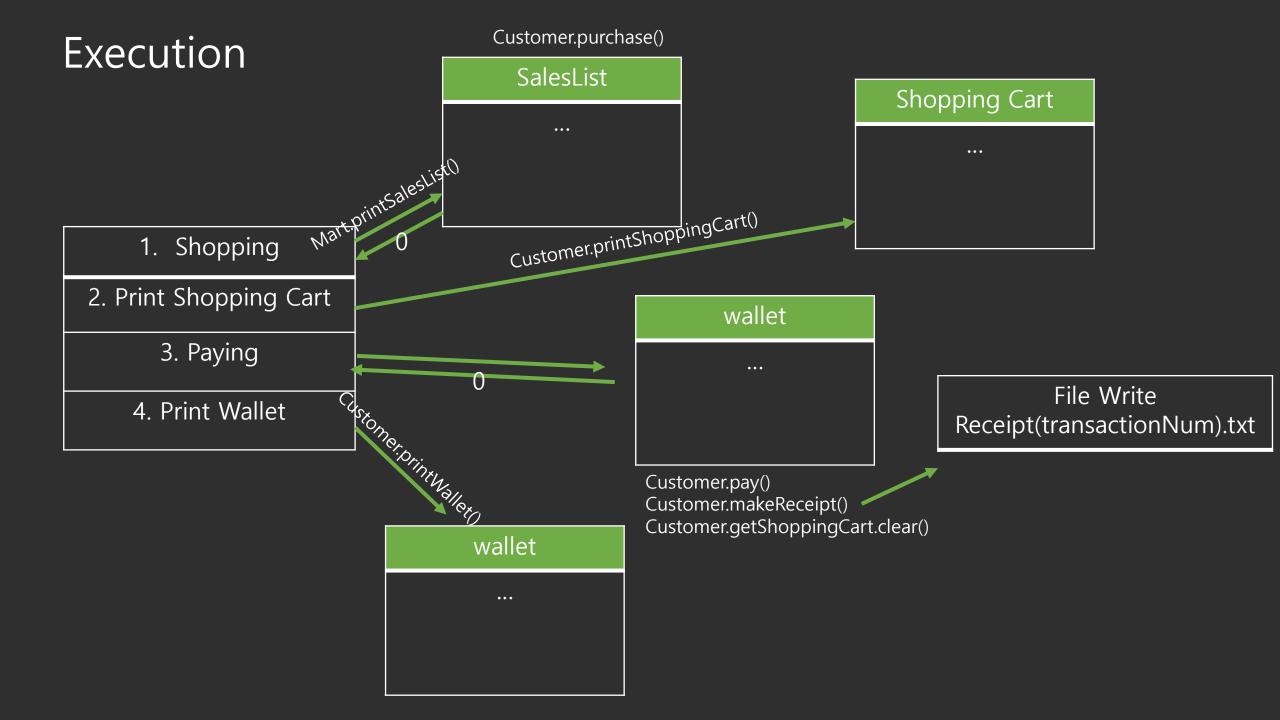


Execution



Execution





Design Pattern

Singleton Pattern

```
Mart.java
public class Mart extends InventoryManager {
   private ArrayList<Product> salesList;
   public static int transactionNum = 1000;

   private static Mart uniqueInstance;
   private Mart() {};
   public synchronized static Mart getInstance() {
        if (uniqueInstance == null)
            uniqueInstance = new Mart();
        return uniqueInstance;
   }
```

- Private Constructor of class Mart
- Unique instance of class Mart
- getInstance() method returns the unique instance of Mart

```
Shopping.java
Mart mart = Mart.getInstance();
```

Unique instance of Mart is referenced in shopping.java

Design Pattern

Observer Pattern

1. Shopping

2. Print Shopping Cart

3. Paying

4. Print Wallet



Add customer to the product wait list

product.wait(this);

- 1. Customer buys all of the remaining stock
- 2. Set product quantity to 0
- 3. Add customer to the product wait list

product.wait(this);

Design Pattern

Observer Pattern

Manager Mode

- 1. Add Inventory
- 2. Replace Expired

If there were 2 customers on the wait list, and first customer bought all of the remaining stock,
The second customer remains on the wait list



For customers on the waiting list of restocked item, let the item be purchased by those customers

And remove customers from the waiting list

Customer tries to buy more than the amount of refilled stock - Remains on the wait list