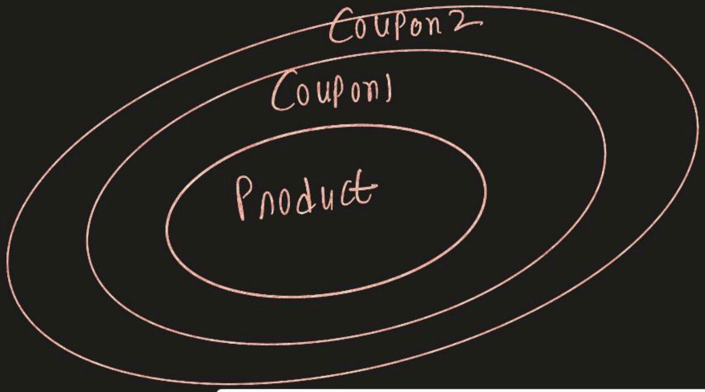
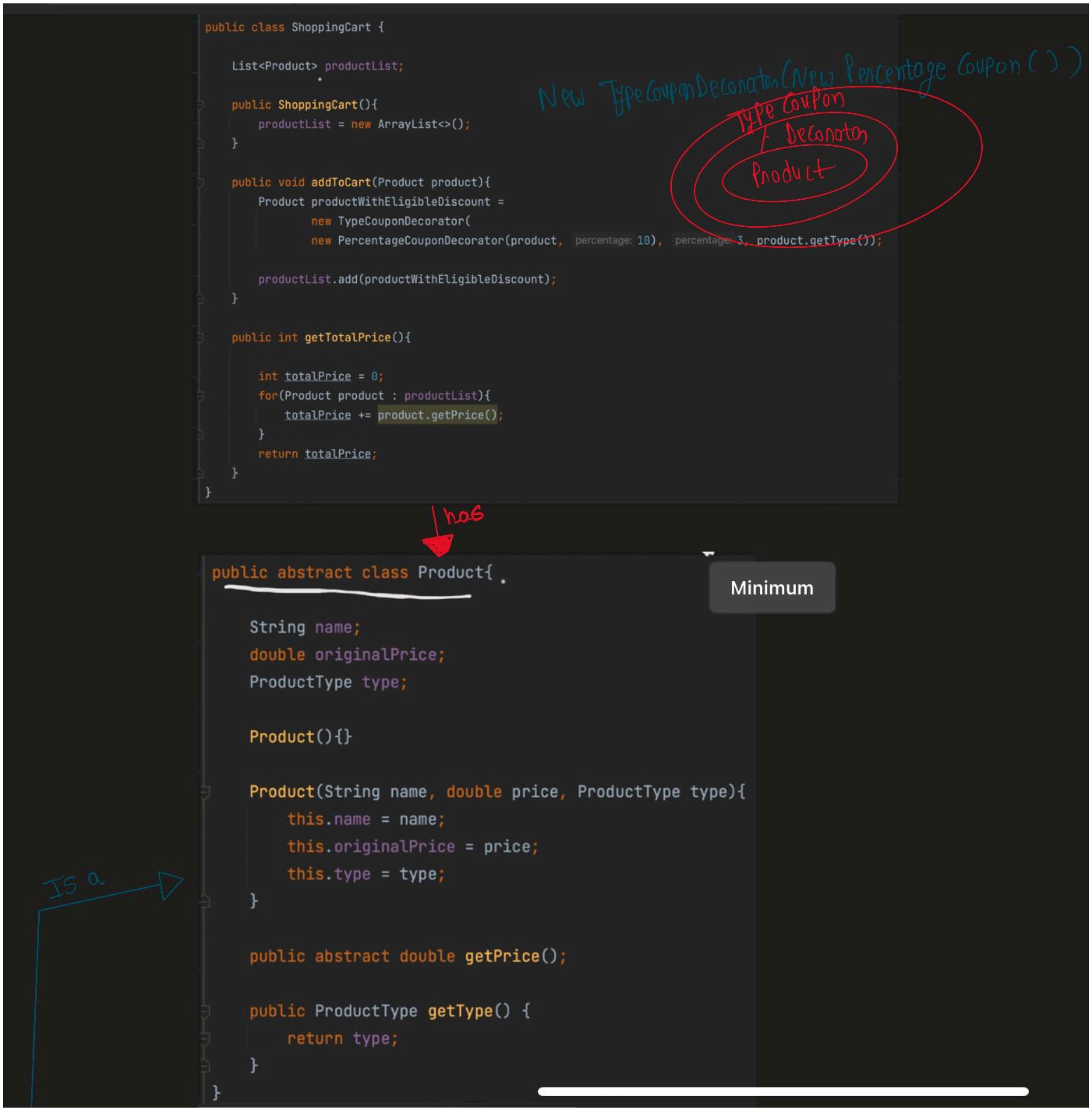
32 LLD of apply Coupons on shopping Cart Products

"Given Shopping cart with products and coupons and calculate the net price after applying coupons on products.
Coupons can be of different types with certain conditions.
1. N% off that is 10% off for all the individual
2.P% off on next item
3.D% off on Nth item of Type T.
Sequentially wants to apply all the coupons on the cart and get the Total amount."

like decorator Pattern

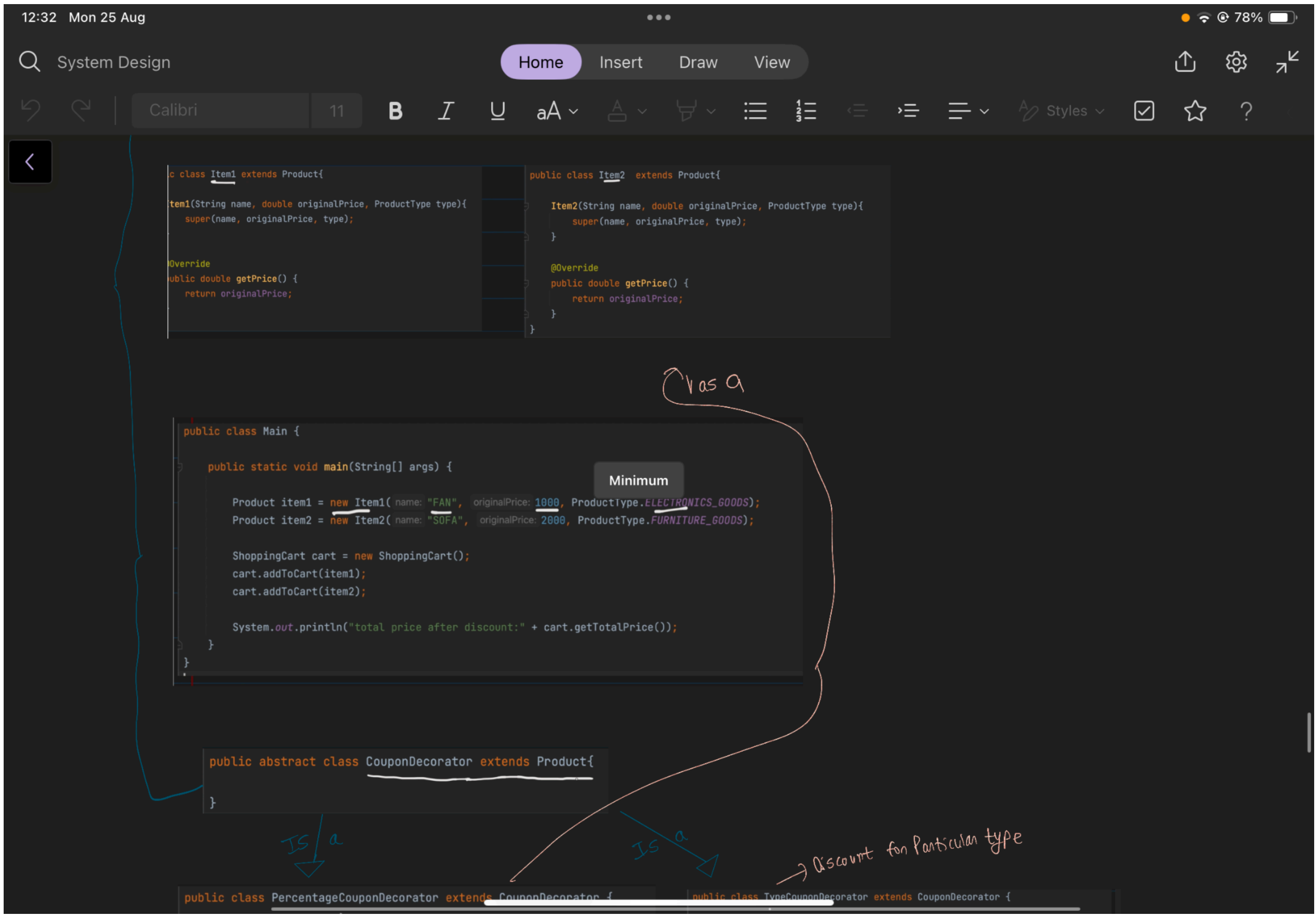
Minimum





```java
public class ShoppingCart {

    List<Product> productList;

    public ShoppingCart(){
        productList = new ArrayList<>();
    }

    public void addToCart(Product product){
        Product productWithEligibleDiscount =
                new TypeCouponDecorator(
                new PercentageCouponDecorator(product,  percentage: 10),  percentage: 3, product.getType());

        productList.add(productWithEligibleDiscount);
    }

    public int getTotalPrice(){

        int totalPrice = 0;
        for(Product product : productList){
            totalPrice += product.getPrice();
        }
        return totalPrice;
    }
}
```

```java
public abstract class Product{

    String name;
    double originalPrice;
    ProductType type;

    Product(){}

    Product(String name, double price, ProductType type){
        this.name = name;
        this.originalPrice = price;
        this.type = type;
    }

    public abstract double getPrice();

    public ProductType getType() {
        return type;
    }
}
```



```java
c class Item1 extends Product{

tem1(String name, double originalPrice, ProductType type){
    super(name, originalPrice, type);


@Override
ublic double getPrice() {
    return originalPrice;
```

```java
public class Item2  extends Product{

    Item2(String name, double originalPrice, ProductType type){
        super(name, originalPrice, type);
    }

    @Override
    public double getPrice() {
        return originalPrice;
    }
}
```

Has a

```java
public class Main {

    public static void main(String[] args) {

        Product item1 = new Item1( name: "FAN",  originalPrice: 1000, ProductType.ELECTRONICS_GOODS);
        Product item2 = new Item2( name: "SOFA",  originalPrice: 2000, ProductType.FURNITURE_GOODS);

        ShoppingCart cart = new ShoppingCart();
        cart.addToCart(item1);
        cart.addToCart(item2);

        System.out.println("total price after discount:" + cart.getTotalPrice());
    }
}
```

```java
public abstract class CouponDecorator extends Product{

}
```

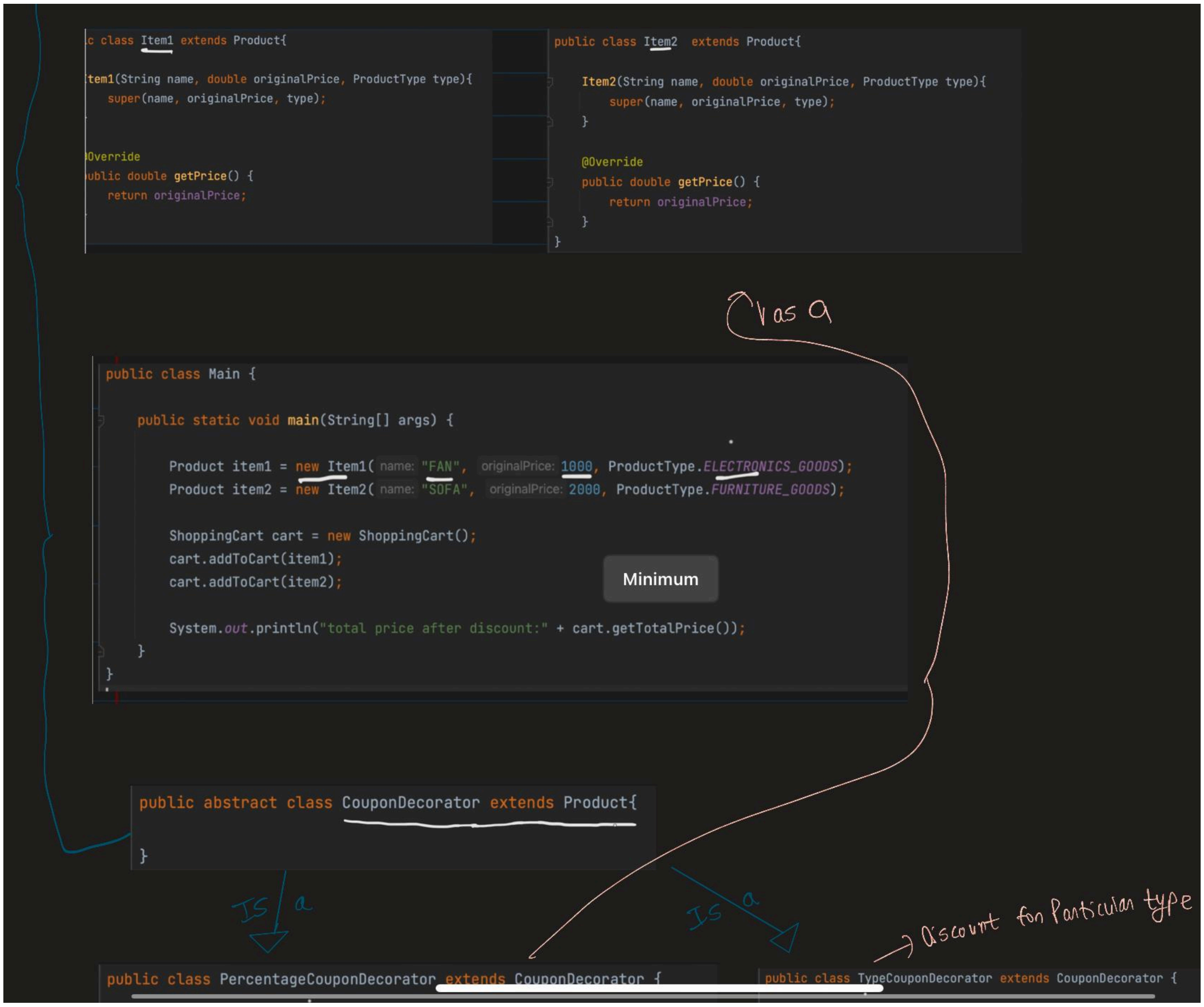
Is a

Is a

→ discount for Particular type

```java
public class PercentageCouponDecorator extends CouponDecorator {
```

```java
public class TypeCouponDecorator extends CouponDecorator {
```



```
c class Item1 extends Product{

Item1(String name, double originalPrice, ProductType type){
    super(name, originalPrice, type);


Override
ublic double getPrice() {
    return originalPrice;
```

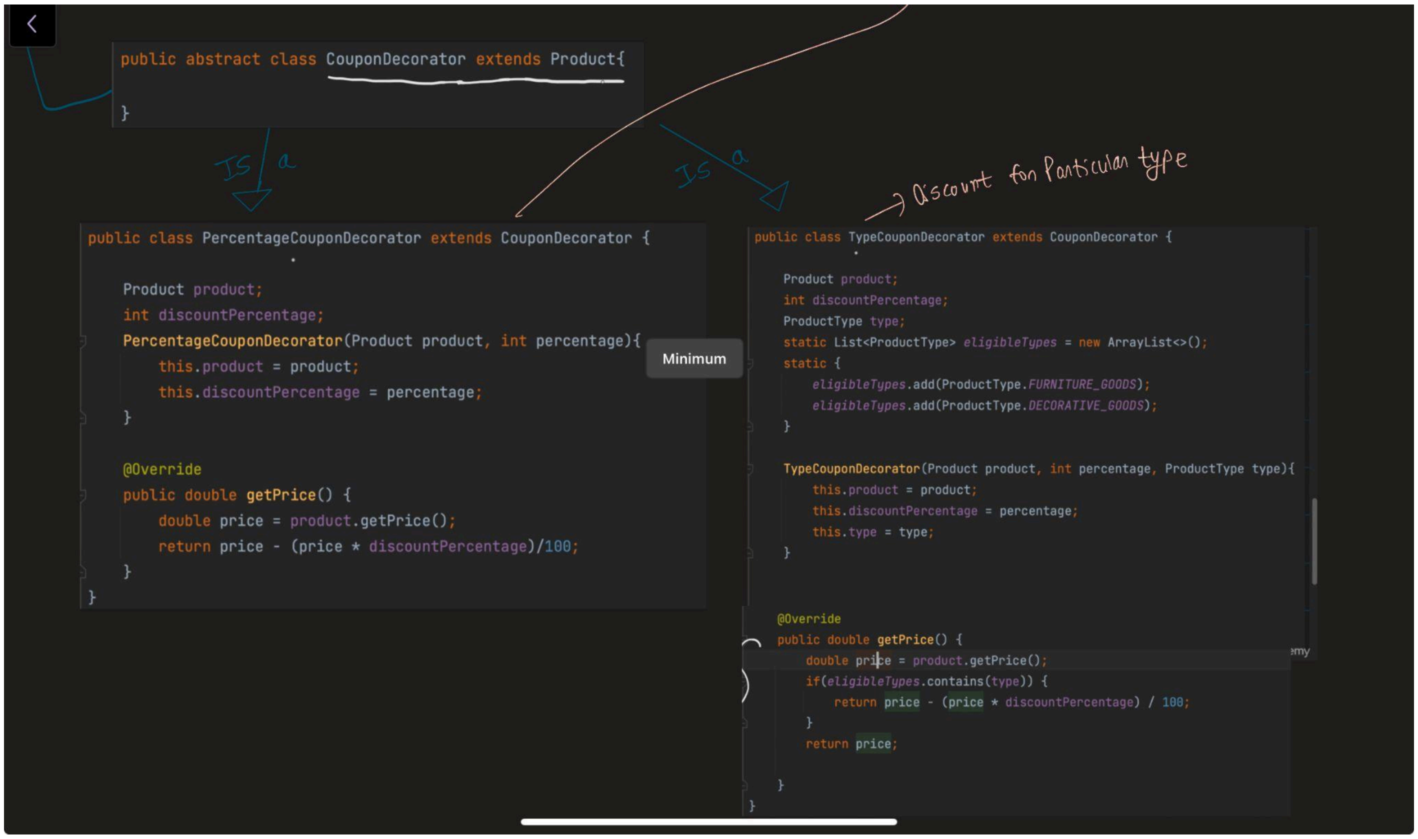
```
public class Item2  extends Product{

    Item2(String name, double originalPrice, ProductType type){
        super(name, originalPrice, type);
    }

    @Override
    public double getPrice() {
        return originalPrice;
    }
}
```

```
public class Main {

    public static void main(String[] args) {

        Product item1 = new Item1( name: "FAN",  originalPrice: 1000, ProductType.ELECTRONICS_GOODS);
        Product item2 = new Item2( name: "SOFA",  originalPrice: 2000, ProductType.FURNITURE_GOODS);

        ShoppingCart cart = new ShoppingCart();
        cart.addToCart(item1);
        cart.addToCart(item2);

        System.out.println("total price after discount:" + cart.getTotalPrice());
    }
}
```

```
public abstract class CouponDecorator extends Product{

}
```

```
public class PercentageCouponDecorator extends CouponDecorator {
```

```
public class TypeCouponDecorator extends CouponDecorator {
```



```java
public abstract class CouponDecorator extends Product{

}
```

```java
public class PercentageCouponDecorator extends CouponDecorator {

    Product product;
    int discountPercentage;
    PercentageCouponDecorator(Product product, int percentage){
        this.product = product;
        this.discountPercentage = percentage;
    }

    @Override
    public double getPrice() {
        double price = product.getPrice();
        return price - (price * discountPercentage)/100;
    }
}
```

```java
public class TypeCouponDecorator extends CouponDecorator {

    Product product;
    int discountPercentage;
    ProductType type;
    static List<ProductType> eligibleTypes = new ArrayList<>();
    static {
        eligibleTypes.add(ProductType.FURNITURE_GOODS);
        eligibleTypes.add(ProductType.DECORATIVE_GOODS);
    }

    TypeCouponDecorator(Product product, int percentage, ProductType type){
        this.product = product;
        this.discountPercentage = percentage;
        this.type = type;
    }


    @Override
    public double getPrice() {
        double price = product.getPrice();
        if(eligibleTypes.contains(type)) {
            return price - (price * discountPercentage) / 100;
        }
        return price;

    }
}
```