Adwaith Ramesh

James Gao

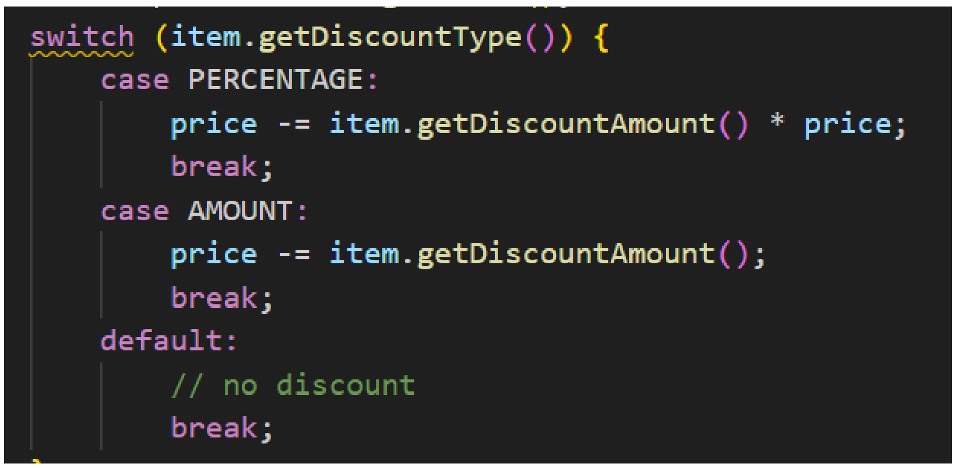
Leila Kazemzadeh

Matthew Sebastian

Michael Zuo

Sahadev Bharath

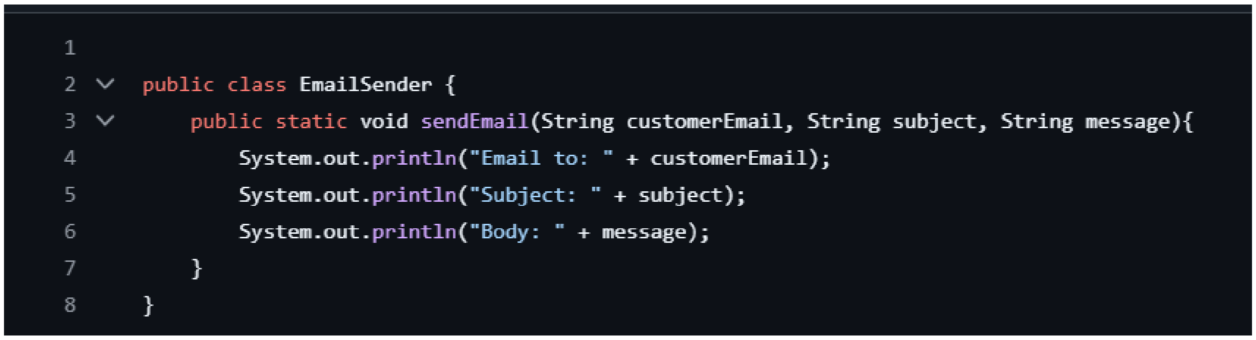
1.



```java
switch (item.getDiscountType()) {
    case PERCENTAGE:
        price -= item.getDiscountAmount() * price;
        break;
    case AMOUNT:
        price -= item.getDiscountAmount();
        break;
    default:
        // no discount
        break;
```

This snippet of code uses a switch statement, which should not be utilized. Fixed by using different methods and calling them to clear logic. The methods use if statements instead of switch statements.

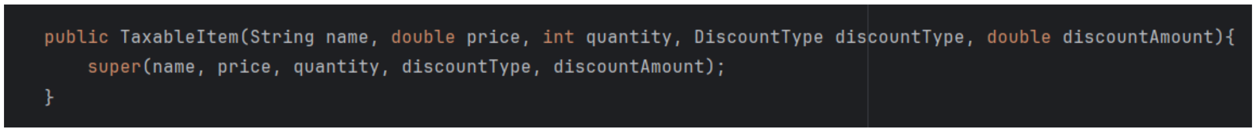
2.



```java

public class EmailSender {
    public static void sendEmail(String customerEmail, String subject, String message){
        System.out.println("Email to: " + customerEmail);
        System.out.println("Subject: " + subject);
        System.out.println("Body: " + message);
    }
}
```

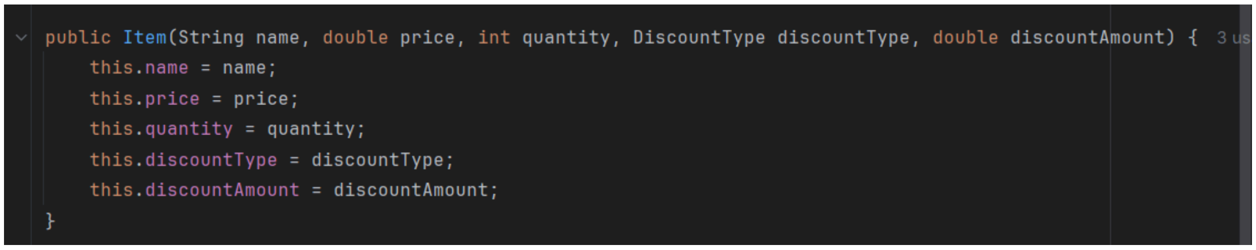
The EmailSender class uses the default implicit constructor. Create a private constructor to hide the implicit public one.

3.

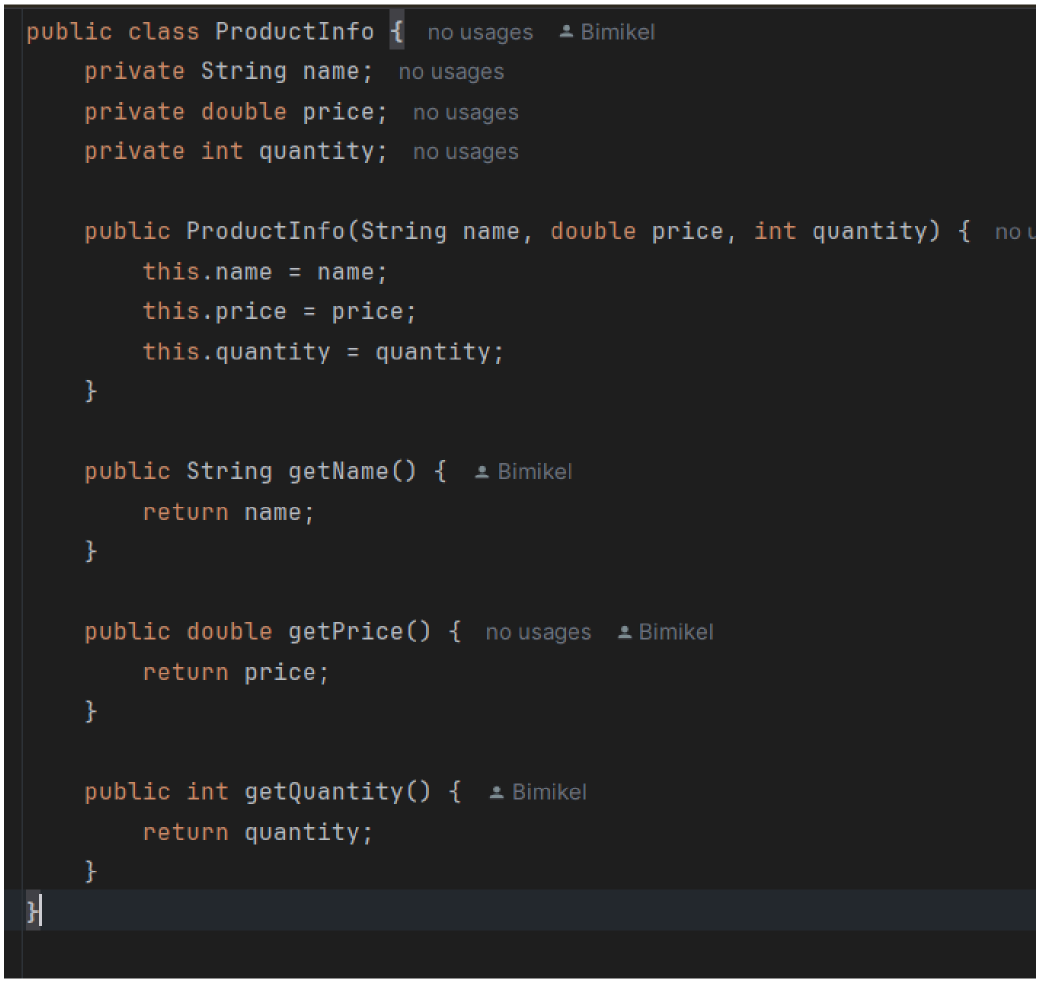

```
public TaxableItem(String name, double price, int quantity, DiscountType discountType, double discountAmount){
    super(name, price, quantity, discountType, discountAmount);
}
```

TaxableItem.java defines TaxableItem to have more than 4 parameters. To fix this, I created a new class ProductInfo that takes in name, price, and quantity to act as one of the parameters for TaxableItem and make the parameter list shorter.

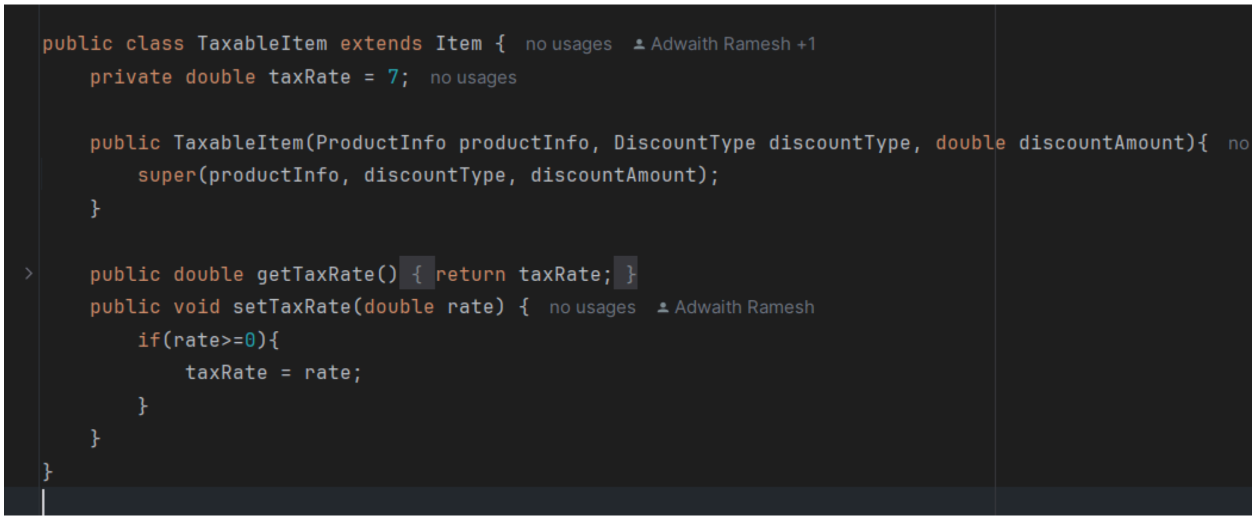
4.


```
public Item(String name, double price, int quantity, DiscountType discountType, double discountAmount) {  3 us
    this.name = name;
    this.price = price;
    this.quantity = quantity;
    this.discountType = discountType;
    this.discountAmount = discountAmount;
}
```

Item has more than 4 parameters. The same solution can be implemented.

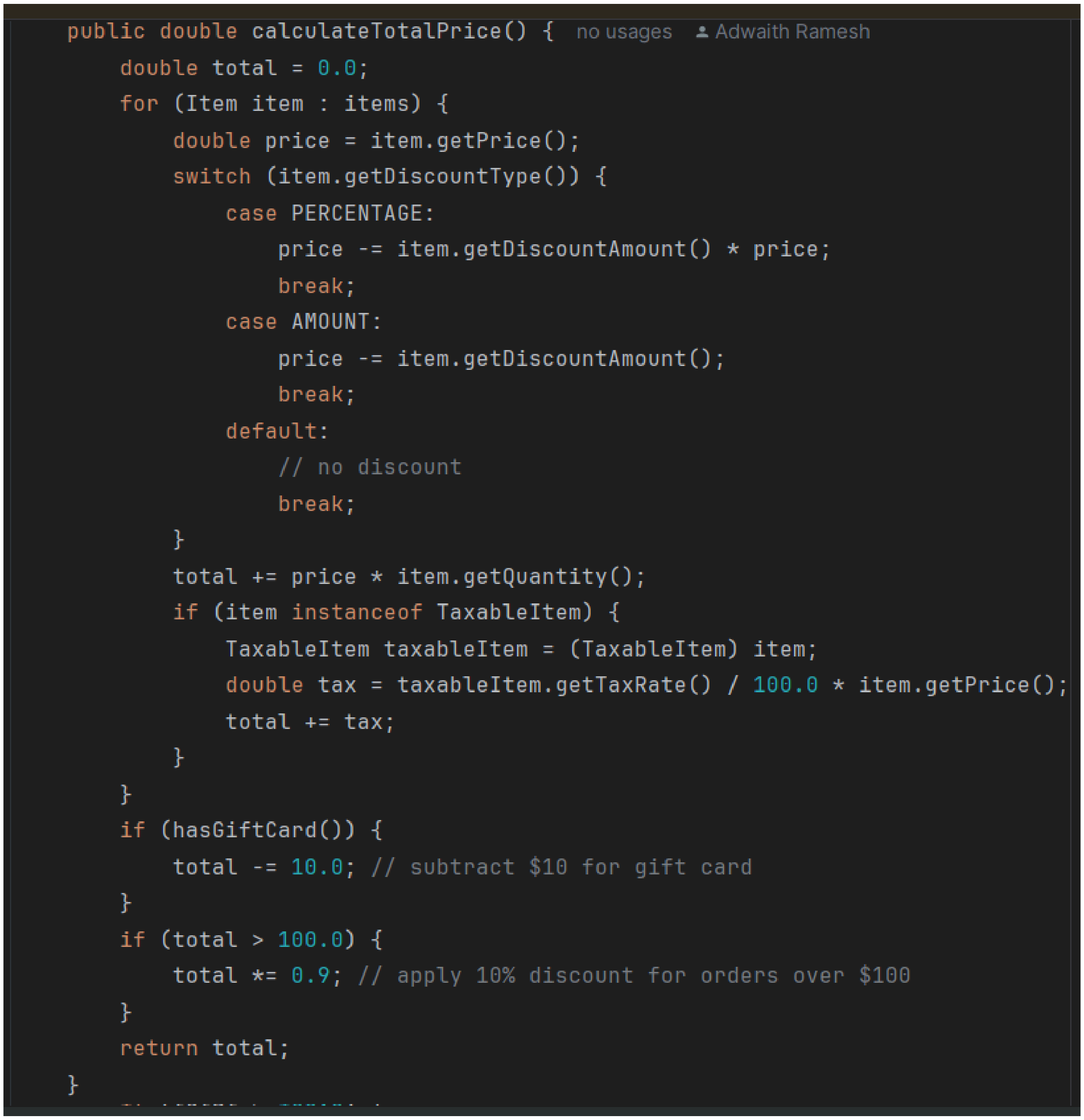


```
public class ProductInfo {  no usages  Bimikel
    private String name;  no usages
    private double price;  no usages
    private int quantity;  no usages

    public ProductInfo(String name, double price, int quantity) {  no u
        this.name = name;
        this.price = price;
        this.quantity = quantity;
    }

    public String getName() {  Bimikel
        return name;
    }

    public double getPrice() {  no usages  Bimikel
        return price;
    }

    public int getQuantity() {  Bimikel
        return quantity;
    }
}
```



```
public class TaxableItem extends Item {  no usages  Adwaith Ramesh +1
    private double taxRate = 7;  no usages

    public TaxableItem(ProductInfo productInfo, DiscountType discountType, double discountAmount){  no
        super(productInfo, discountType, discountAmount);
    }

    public double getTaxRate() { return taxRate; }
    public void setTaxRate(double rate) {  no usages  Adwaith Ramesh
        if(rate>=0){
            taxRate = rate;
        }
    }
}
```

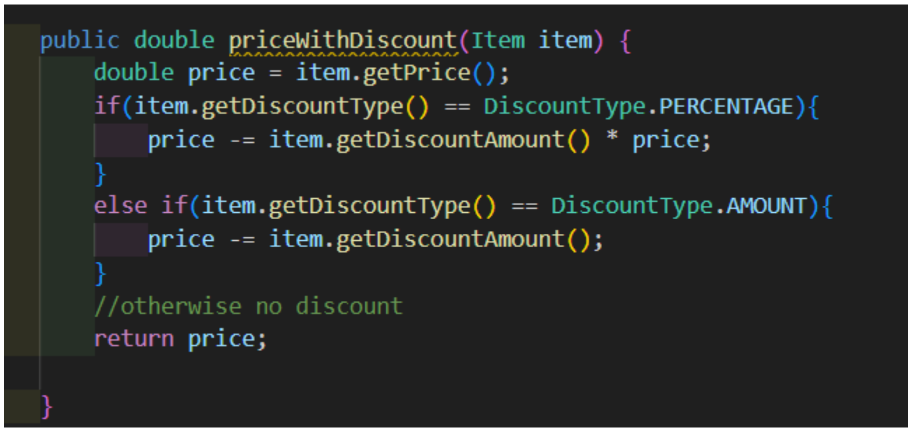
5.



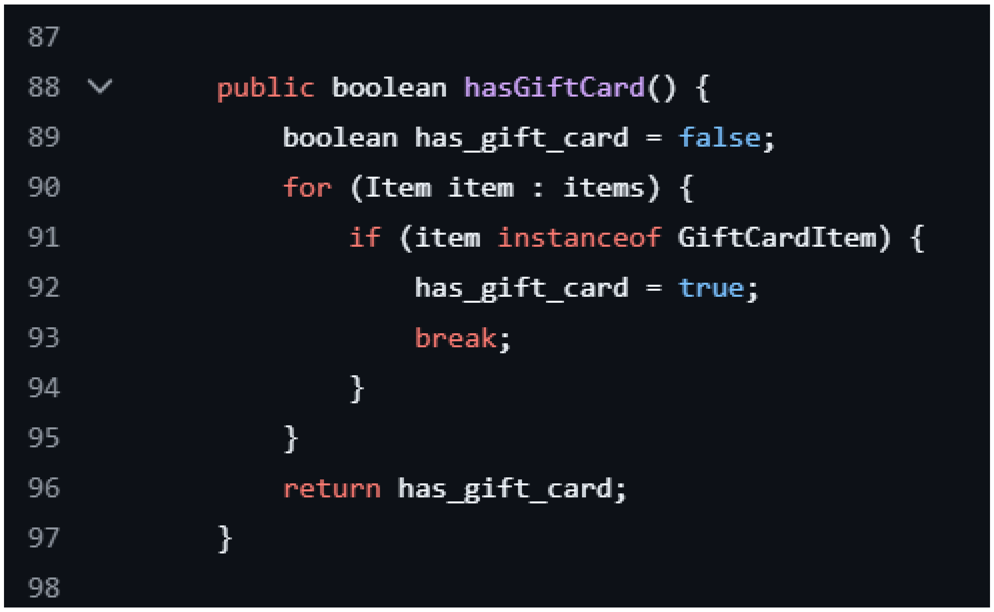
```
public double calculateTotalPrice() {  no usages  Adwaith Ramesh
    double total = 0.0;
    for (Item item : items) {
        double price = item.getPrice();
        switch (item.getDiscountType()) {
            case PERCENTAGE:
                price -= item.getDiscountAmount() * price;
                break;
            case AMOUNT:
                price -= item.getDiscountAmount();
                break;
            default:
                // no discount
                break;
        }
        total += price * item.getQuantity();
        if (item instanceof TaxableItem) {
            TaxableItem taxableItem = (TaxableItem) item;
            double tax = taxableItem.getTaxRate() / 100.0 * item.getPrice();
            total += tax;
        }
    }
    if (hasGiftCard()) {
        total -= 10.0; // subtract $10 for gift card
    }
    if (total > 100.0) {
        total *= 0.9; // apply 10% discount for orders over $100
    }
    return total;
}
```

Long method, the special cases of discount can be calculated separately. In order to fix this I pulled out a new method “priceWithDiscount()” which does the work of the switch statement using an if statement, decreases the length of the method, and makes the code more readable.

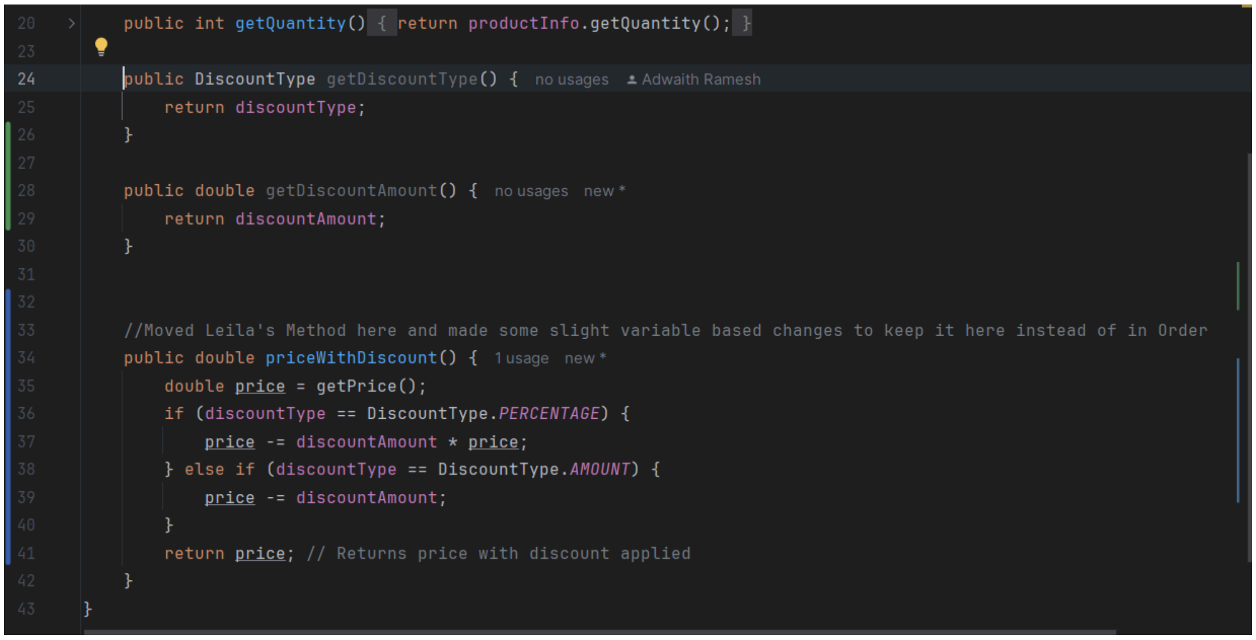
New method shown below:


```java
public double priceWithDiscount(Item item) {
    double price = item.getPrice();
    if(item.getDiscountType() == DiscountType.PERCENTAGE){
        price -= item.getDiscountAmount() * price;
    }
    else if(item.getDiscountType() == DiscountType.AMOUNT){
        price -= item.getDiscountAmount();
    }
    //otherwise no discount
    return price;

}
```

6.

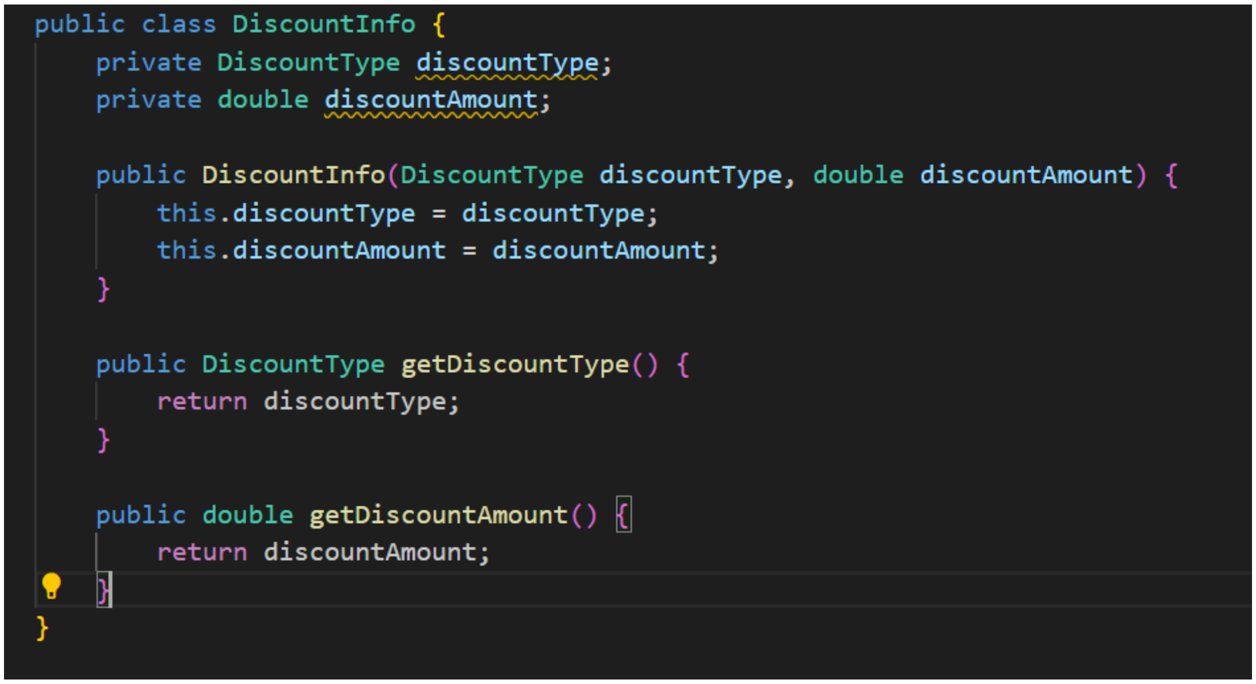

```java

    public boolean hasGiftCard() {
        boolean has_gift_card = false;
        for (Item item : items) {
            if (item instanceof GiftCardItem) {
                has_gift_card = true;
                break;
            }
        }
        return has_gift_card;
    }

```

The has_gift_card boolean does not follow the preferred name format. Refactor the boolean name to hasGiftCard.



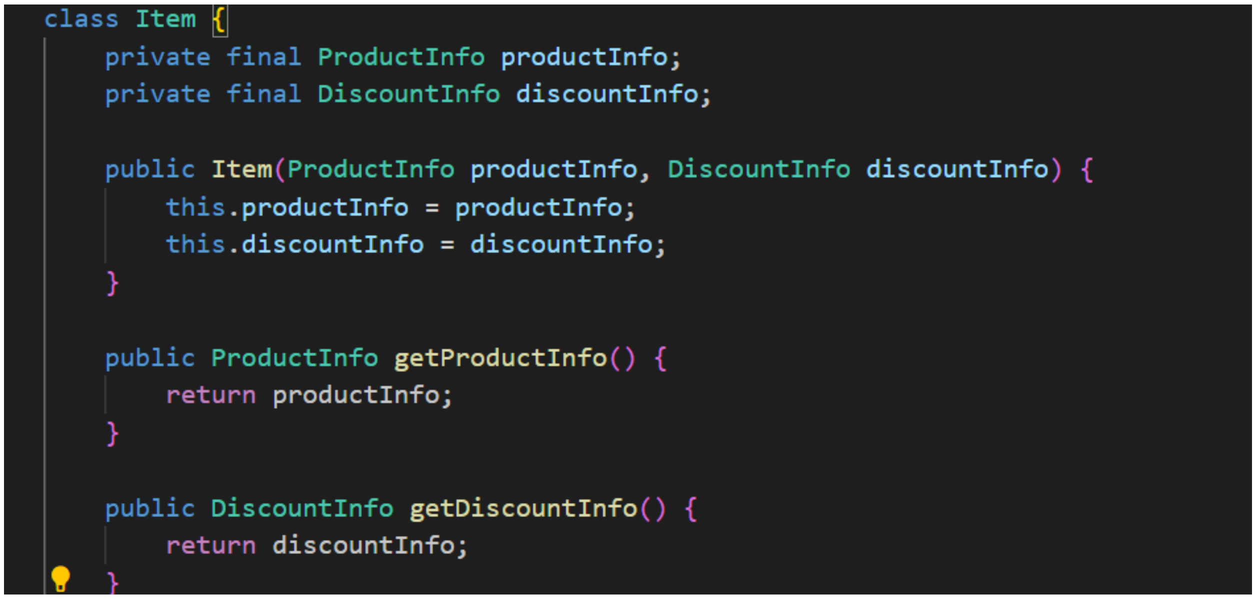
```
    public int getQuantity() { return productInfo.getQuantity(); }

    public DiscountType getDiscountType() {  no usages  Adwaith Ramesh
        return discountType;
    }

    public double getDiscountAmount() {  no usages  new *
        return discountAmount;
    }


    //Moved Leila's Method here and made some slight variable based changes to keep it here instead of in Order
    public double priceWithDiscount() {  1 usage  new *
        double price = getPrice();
        if (discountType == DiscountType.PERCENTAGE) {
            price -= discountAmount * price;
        } else if (discountType == DiscountType.AMOUNT) {
            price -= discountAmount;
        }
        return price; // Returns price with discount applied
    }
}
```

Moved the priceWithDiscount method from Order to Item to reduce a bit of coupling and improve feature envy. Also removed the DiscountType and discountAmount since they arent used anymore.



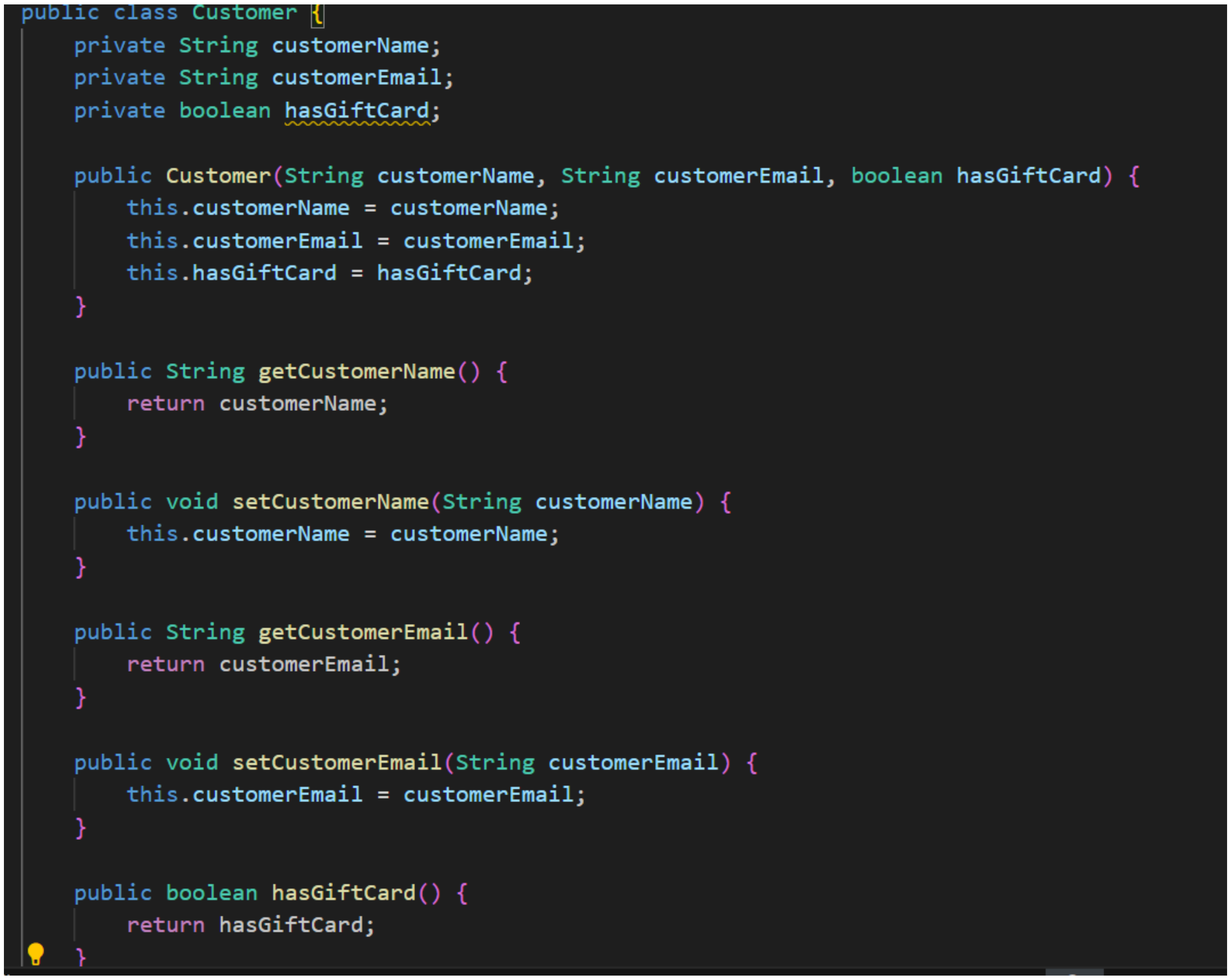
```
public class DiscountInfo {
    private DiscountType discountType;
    private double discountAmount;

    public DiscountInfo(DiscountType discountType, double discountAmount) {
        this.discountType = discountType;
        this.discountAmount = discountAmount;
    }

    public DiscountType getDiscountType() {
        return discountType;
    }

    public double getDiscountAmount() {
        return discountAmount;
    }
}
```

Added a DiscountInfo class to keep discount information separate



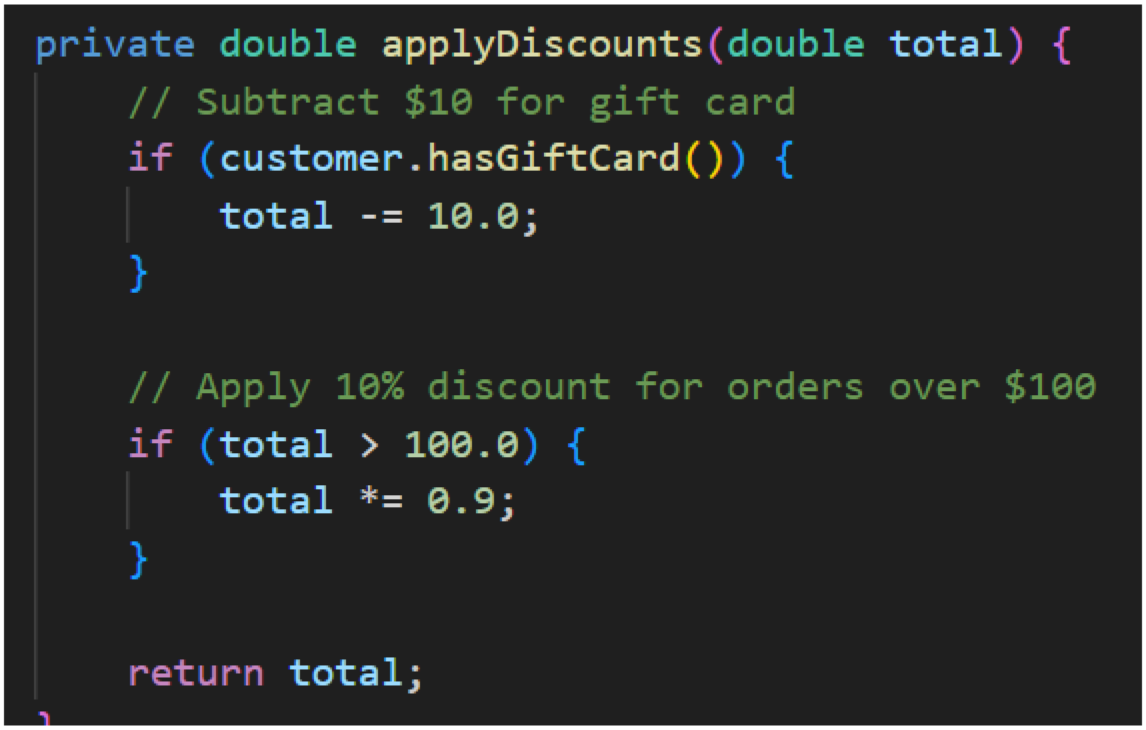
```java
class Item {
    private final ProductInfo productInfo;
    private final DiscountInfo discountInfo;

    public Item(ProductInfo productInfo, DiscountInfo discountInfo) {
        this.productInfo = productInfo;
        this.discountInfo = discountInfo;
    }

    public ProductInfo getProductInfo() {
        return productInfo;
    }

    public DiscountInfo getDiscountInfo() {
        return discountInfo;
    }
```

Removed GiftCardItem class and added it to Customer. Modified Item with the discountInfo



```java
public class Customer {
    private String customerName;
    private String customerEmail;
    private boolean hasGiftCard;

    public Customer(String customerName, String customerEmail, boolean hasGiftCard) {
        this.customerName = customerName;
        this.customerEmail = customerEmail;
        this.hasGiftCard = hasGiftCard;
    }

    public String getCustomerName() {
        return customerName;
    }

    public void setCustomerName(String customerName) {
        this.customerName = customerName;
    }

    public String getCustomerEmail() {
        return customerEmail;
    }

    public void setCustomerEmail(String customerEmail) {
        this.customerEmail = customerEmail;
    }

    public boolean hasGiftCard() {
        return hasGiftCard;
    }
```

Added a customer class to separate customer from order (added hasGiftCard instead of the giftcard class)



```java
private double applyDiscounts(double total) {
    // Subtract $10 for gift card
    if (customer.hasGiftCard()) {
        total -= 10.0;
    }

    // Apply 10% discount for orders over $100
    if (total > 100.0) {
        total *= 0.9;
    }

    return total;
```

Separate discounts into its own method