12008073_AndreasMariusBaisan

5.4_Refactoring Steps

```java
private Double getPrice(Order order) {
    Double totalPrice = 0.0d;
    for (Item item : order.getItems()) {
        totalPrice += item.getPrice();
    }
    return totalPrice;
}
```

```java
private Order addShippingCost(Double totalPrice, Order order) {
    if (totalPrice <= 100) {
        Item item = new Item();
        item.setId(99l);
        item.setName("Porto und Versand");
        if (totalPrice > 90) {
            item.setPrice(totalPrice * 0.05);
        } else if (totalPrice > 50) {
            item.setPrice(7.5d);
        } else {
            item.setPrice(10d);
        }
        order.getItems().add(item);
    }
    return order;
}
```

```java
private void printOrder(Order order, Double totalPrice) {
    System.out.println("Rechnung:");
    for (Item item : order.getItems()) {
        System.out.println(item.getName() + ": " + item.getPrice());
    }
    System.out.println("Total: " + totalPrice);
}
```

Aus den jeweiligen einzelnen Abschnitten eigene Klassen erstellt.
Außerdem bei getPrice einen return Wert hinzugefügt, um doppelten Code nach der Berechnung der Versandkosten zu vermeiden.
addShippingCost returned die Order die übergeben wird mit den eingefügten Versandkosten die durch die Übergabe des totalPrice ausgewählt wurden.
printOrder ist anschließend das ausgeben der Order.

```java
public void erstelleRechnung(Order order) {
    Double totalPrice = getPrice(order);
    order = addShippingCost(totalPrice, order);
    totalPrice = getPrice(order);
    printOrder(order, totalPrice);
}
```

Unsere neue erstelleRechnung methode führt die gesamte Bestellung aus, nur über den neuen Weg, das Ergebnis ist wie in der alten Methode.
Dadurch sind wir LongMethod und DuplicatedCode entgangen.

```java
public class Main {
    public static void main(String[] args) {

        SmellyClassRefactored scr = new SmellyClassRefactored();
        SmellyClass sc = new SmellyClass();

        Customer customer = new Customer();
        customer.setId(1L);
        customer.setVorname("Stefan");
        customer.setNachname("Haarbringer");

        Item item1 = new Item();
        Item item2 = new Item();
        Item item3 = new Item();

        item1.setId(1L);
        item1.setPrice(15.0d);
        item1.setName("Wein");

        item2.setId(2L);
        item2.setPrice(20.0d);
        item2.setName("Wodka");

        item3.setId(1L);
        item3.setPrice(25.0d);
        item3.setName("Champagner");
        ArrayList<Item> list = new ArrayList<>();
        list.add(item1);
        list.add(item2);
        list.add(item3);

        ArrayList<Item> list2 = new ArrayList<>();
        list2.add(item1);
        list2.add(item2);
        list2.add(item3);

        Order order1 = new Order();
        order1.setId(1L);
        order1.setCustomer(customer);
        order1.setItems(list);

        Order order2 = new Order();
        order2.setId(1L);
        order2.setCustomer(customer);
        order2.setItems(list2);

        System.out.println("Bestellung 1 (SmellyClass)");
        sc.erstelleRechnung(order1);
        System.out.println("-------------------------------------");
        System.out.println("Bestellung 2 (SmellyClassRefactored)");
        scr.erstelleRechnung(order2);
```

```
Bestellung 1 (SmellyClass)
Rechnung:
Wein: 15.0
Wodka: 20.0
Champagner: 25.0
Porto und Versand: 7.5
Total: 67.5
-------------------------------------
Bestellung 2 (SmellyClassRefactored)
Rechnung:
Wein: 15.0
Wodka: 20.0
Champagner: 25.0
Porto und Versand: 7.5
Total: 67.5
```

Zur Überprüfung das ganze in der Main Methode getestet.
Gleiches Ergebnis und aufgeräumter Code.