Буниф Мажда

18-КБ-ПР2

**Отчет ЭПО Лаб 2**

**4 Задание**

1) Преобразовать метод statement класса Bill, и перенести методы получения скидки и начисления бонусов в класс Goods в соответствии с предложенными в пункте 2.6 изменениями, применив описанные в теории методы рефакторинга.

2) Проверить корректность работы нового метода и работоспособность старого метода формирования счета.

**Листинг :**

**Bill.cs // class without GetBonus()&& GetDiscount() mehods**

public class Bill

{

private List<Item> \_items;

private Customer \_customer;

public Bill(Customer customer)

{

this.\_customer = customer;

this.\_items = new List<Item>();

}

public void addGoods(Item arg)

{

\_items.Add(arg);

}

public string Statement()

{

double totalAmount = 0;

int totalBonus = 0;

String Print = "Счет для " + \_customer.getName() + "\n";

Print += GetHeader();

//foreach (Item item in \_items) // Formula in paragraph 2.10-9

//{

// double itemPrice = Total\_Price(item);

// int bonus = item.GetBonus();

// double discount = item.GetDiscount();

// double PriceWithDiscount = itemPrice - discount;

// double usedBonus = UseBonus(item, PriceWithDiscount);

// double thisAmount = PriceWithDiscount - usedBonus;

// totalAmount = totalAmount + thisAmount;

// totalBonus = totalBonus + bonus;

// Print += GetItemString(discount, bonus, thisAmount, item); // show the item string

//}

foreach (Item item in \_items)

{

int bonus = item.GetBonus(); // after the change

double discount = item.GetDiscount(); // after the change

double totalPrice = item.getQuantity() \* item.getPrice();

double PriceWithDiscount = totalPrice - discount;

double usedBonus = UseBonus(item, PriceWithDiscount);

double thisAmount = PriceWithDiscount - usedBonus;

totalAmount =+ thisAmount;

totalBonus =+ totalBonus + bonus;

Print += GetItemString(discount, bonus, thisAmount, item);

}

Print += GetFooter(totalAmount, totalBonus); //display the footer

\_customer.receiveBonus(totalBonus);

return Print;

}

//The Header

public string GetHeader()

{

return "\t" + "Название" + "\t" + "Цена" +

"\t" + "Кол-во" + "Стоимость" + "\t" + "Скидка" +

"\t" + "Сумма" + "\t" + "Бонус" + "\n";

}

//The Footer

private string GetFooter(double totalAmount, int totalBonus)

{

return "Сумма счета составляет " + totalAmount.ToString() + "\nВы заработали " + totalBonus.ToString() + " бонусных баллов";

}

//String of Items

private string GetItemString(double discount, int bonus, double thisAmount, Item item)

{

return "\t" + item.getGoods().getTitle() + "\t" + // or item

"\t" + item.getPrice() + "\t" + item.getQuantity() +

"\t" + (item.getQuantity() \* item.getPrice()).ToString() +

"\t" + discount.ToString() + "\t" + thisAmount.ToString() +

"\t" + bonus.ToString() + "\n";

}

//Formula Of Total Cost

private double Total\_Price(Item item)

{

return item.getQuantity() \* item.getPrice();

}

//using Bonuses

private double UseBonus(Item item, double PriceWithDiscount)

{

double bonus = 0.0;

if ((item.getGoods().getPriceCode() == Goods.REGULAR) && item.getQuantity() > 5)

bonus = \_customer.useBonus((int)PriceWithDiscount);

else if ((item.getGoods().getPriceCode() == Goods.SPECIAL\_OFFER) && item.getQuantity() > 1)

bonus = \_customer.useBonus((int)PriceWithDiscount);

return bonus;

}

}}

**Item.cs //**

public class Item

{

private Goods \_Goods;

private int \_quantity;

private double \_price;

public Item(Goods Goods, int quantity, double price)

{

\_Goods = Goods;

\_quantity = quantity;

\_price = price;

}

public int getQuantity()

{

return \_quantity;

}

public double getPrice()

{

return \_price;

}

public Goods getGoods()

{

return \_Goods;

}

public int GetBonus()

{

return \_Goods.GetBonus(\_quantity, \_price);

}

//public double GetDiscount(Item item) => item.GetDiscount(item);

//public double GetDiscount(Item item) => item.GetDiscount();

public double GetDiscount()

{

return \_Goods.GetDiscount(\_quantity, \_price);

}

}

}

**Goods.cs // copy &&set GetDiscount() and GetBonus() as public**

public class Goods

{

public const int REGULAR = 0;

public const int SALE = 1;

public const int SPECIAL\_OFFER = 2;

private String \_title;

private int \_priceCode;

public Goods(String title, int priceCode)

{

\_title = title;

\_priceCode = priceCode;

}

public int getPriceCode()

{

return \_priceCode;

}

public void setPriceCode(int arg)

{

\_priceCode = arg;

}

public String getTitle()

{

return \_title;

}

//public GetDiscount

public double GetDiscount(int quantity, double price)

{

double discount = 0.0;

switch (getPriceCode())

{

case Goods.REGULAR:

if (quantity > 2)

return discount = ((quantity\*price)\* 0.03);

break;

case Goods.SPECIAL\_OFFER:

if (quantity> 10)

return discount = ((quantity\*price) \* 0.005);

break;

case Goods.SALE:

if (quantity> 3)

return discount = ((quantity\*price) \* 0.01);

break;

}

return discount;

}

//public GetBonus

public int GetBonus(int quantity, double price)

{

int bonus = 0;

switch (getPriceCode())

{

case Goods.REGULAR:

bonus = (int)(quantity \* price \* 0.05);

break;

case Goods.SALE:

bonus = (int)(quantity \* price\* 0.01);

break;

}

return bonus;

}

}

}

**Tests.cs**

[TestFixture]

public class TestsNewMethods

{

[Test]

public void TestRegularItemWithDiscount()

{

Goods fanta = new Goods("fanta", Goods.REGULAR);

Item i1 = new Item(fanta, 3, 65);

Customer x = new Customer("Sara", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Sara\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tfanta\t\t65\t3\t195\t5,85\t189,15\t9\nСумма счета составляет 189,15\nВы заработали 9 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestRegularItemWithoutDiscount()

{

Goods pepsi = new Goods("pepsi", Goods.REGULAR);

Item i1 = new Item(pepsi, 1, 65);

Customer x = new Customer("Ali", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Ali\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tpepsi\t\t65\t1\t65\t0\t65\t3\nСумма счета составляет 65\nВы заработали 3 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestRegularItemWithoutBonuses()

{

Goods cola = new Goods("Cola", Goods.REGULAR);

Item i1 = new Item(cola, 5, 65);

Customer x = new Customer("Tao", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Tao\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCola\t\t65\t5\t325\t9,75\t315,25\t16\nСумма счета составляет 315,25\nВы заработали 16 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestRegularItemWithBonuses()

{

Goods cola = new Goods("Cola", Goods.REGULAR);

Item i1 = new Item(cola, 6, 65);

Customer x = new Customer("Aya", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Aya\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCola\t\t65\t6\t390\t11,7\t368,3\t19\nСумма счета составляет 368,3\nВы заработали 19 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSpecialItemWithDiscount()

{

Goods Cola = new Goods("Cola", Goods.SPECIAL\_OFFER);

Item i1 = new Item(Cola, 12, 65);

Customer x = new Customer("Ana", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Ana\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCola\t\t65\t12\t780\t3,9\t766,1\t0\nСумма счета составляет 766,1\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSpecialItemWithoutDiscount()

{

Goods Mirenda = new Goods("Mirenda", Goods.SPECIAL\_OFFER);

Item i1 = new Item(Mirenda, 10, 65);

Customer x = new Customer("Marwa", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Marwa\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tMirenda\t\t65\t10\t650\t0\t640\t0\nСумма счета составляет 640\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSaleItemWithoutDiscount()

{

Goods Coffee = new Goods("Coffee", Goods.SALE);

Item i1 = new Item(Coffee, 3, 65);

Customer x = new Customer("Nouha", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для Nouha\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCoffee\t\t65\t3\t195\t0\t195\t1\nСумма счета составляет 195\nВы заработали 1 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSaleItemWithDiscount()

{

Goods OrangeJus = new Goods("OrangeJus", Goods.SALE);

Item i1 = new Item(OrangeJus, 4, 65);

Customer x = new Customer("David", 10);

Bill b1 = new Bill(x);

b1.addGoods(i1);

string actual = b1.Statement();

string expected = "Счет для David\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tOrangeJus\t\t65\t4\t260\t2,6\t257,4\t2\nСумма счета составляет 257,4\nВы заработали 2 бонусных баллов";

Assert.AreEqual(expected, actual);

}

}

}

**By running the same previous tests the output is:**

