Буниф Мажда

18-КБ-ПР2

**Отчет ЭПО Лаб 4**

**4 Задание**

1) Преобразовать методы и классы программы в соответствии с предложенными в пункте 2.7 изменениями, применив описанные в теории методы рефакторинга.

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

**Листинг :**

**Interface IPresenter**

public interface IPresenter

{

string GetHeader();

string GetFooter(double totalAmount, int totalBonus);

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

}

**BillGenerator**

public class BillGenerator

{

private List<Item> \_items;

private Customer \_customer;

private IPresenter p;

public BillGenerator(Customer customer, IPresenter presenter)

{

this.\_customer = customer;

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

this.p = presenter;

}

public void addGoods(Item arg)

{

\_items.Add(arg);

}

public string GenerateBill()

{

double totalAmount = 0;

int totalBonus = 0;

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

Print += p.GetHeader();

foreach (Item item in \_items)

{

int bonus = item.GetBonus();

double discount = item.GetDiscount();

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 += p.GetItemString(discount, bonus, thisAmount, item);

}

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

\_customer.receiveBonus(totalBonus);

return Print;

}

public void Presenter(IPresenter presenter)

{

p = presenter;

}

//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().GetType() == typeof(RegularItem)) && item.getQuantity() > 5)

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

else if ((item.getGoods().GetType() == typeof(SpecialItem)) && item.getQuantity() > 1)

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

return bonus;

}

}

}

**HTML Presenter Class**

public class HTMLPresenter : IPresenter

{

//The Header

public string GetHeader()

{

throw new NotImplementedException();

}

//The Footer

public string GetFooter(double totalAmount, int totalBonus)

{

throw new NotImplementedException();

}

//String of Items

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

{

throw new NotImplementedException();

}

}

}

**Class Implements IPresenter Interface**

public class TXTPresenter : IPresenter

{

//The Header

public string GetHeader()

{

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

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

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

}

//The Footer

public string GetFooter(double totalAmount, int totalBonus)

{

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

}

//String of Items

public 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";

}

}

}

**Tests.cs**

[TestFixture]

public class TestsNewMethods

{

[Test]

public void TestRegularItemWithDiscount()

{

Goods fanta = new RegularItem("fanta");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1= new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 RegularItem("pepsi");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 RegularItem("cola");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 RegularItem("Cola");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 SpecialItem("Cola");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 SpecialItem("Mirenda");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 SalesItem("Coffee");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 SalesItem("OrangeJus");

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

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

IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, p);

b1.addGoods(i1);

string actual = b1.GenerateBill();

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 while using BillGenerator the output is:**

