Буниф Мажда

18-КБ-ПР2

**Отчет ЭПО Лаб 6**

**4 Задание**

1) Изменить процесс назначения бонусов и скидок в соответствии с предложенными в пункте 2.2 изменениями, применив описанные в теории методы рефакторинга.

2) Проверить согласованность результатов и корректность работы нового варианта программы с результатами старой версии.

3) Оформить отчёт.

**Листинг :**

**IBonusStrategy**

public interface IBonusStrategy

{

int GetBonus(int quantity,double price);

bool PossibilityOfBonus(int quantity, double price);

}

**IDiscountStrategy**

public interface IDiscountStrategy

{

double GetDiscount(int quantity, double price);

}

**Class AmountForQuantity**

public class AmountForQuantity : IBonusStrategy

{

private int \_Bonusquantity;

private double \_percent;

public AmountForQuantity(int Bonusquantity, double percent)

{

\_Bonusquantity = Bonusquantity;

\_percent = percent;

}

public int GetBonus(int quantity, double price)

{

return (int)(price \* quantity\* \_percent);

}

public bool PossibilityOfBonus(int quantity, double price)

{

if (\_Bonusquantity != -1 & quantity > \_Bonusquantity) return true;

return false;

}

}

**Class AmountForSum**

public class AmountForSum : IBonusStrategy

{

private double \_sum, \_percent, \_Bonusquantity;

public AmountForSum(double sum, double percent, int Bonusquantity)

{

\_sum = sum;

\_percent = percent;

\_Bonusquantity = Bonusquantity;

}

public int GetBonus(int quantity, double price)

{

if (price\*quantity > \_sum) return (int)(price \* quantity \* \_percent);

return 0;

/// OR

///

//if (price> \_sum)

//return (int)(price \* \_percent);

}

public bool PossibilityOfBonus(int quantity, double price)

{

if (\_Bonusquantity != -1 & quantity > \_Bonusquantity) return true;

return false;

}

**Class FixedAmount**

public class FixedAmount: IBonusStrategy

{

double \_percent;

int \_Bonusquantity;

public FixedAmount(double percent, int Bonusquantity)

{

\_percent = percent;

\_Bonusquantity = Bonusquantity;

}

public int GetBonus(int quantity, double price)

{

return (int)(price \*\_percent);

}

public bool PossibilityOfBonus(int quantity, double price)

{

if (\_Bonusquantity != -1 & quantity > \_Bonusquantity) return true;

return false;

}

}

**Class FixedPercent**

public class FixedPercent: IDiscountStrategy

{

private double \_percent;

public FixedPercent(double percent)

{

\_percent = percent;

}

public double GetDiscount(int quantity, double price)

{

double discount = GetSum()\*\_percent;

return discount;

double GetSum()

{

return quantity \* price;

}

}

}

**Class PercentForQuantity**

public class PercentForQuantity : IDiscountStrategy

{

private int \_quantity;

private double \_percent;

public PercentForQuantity(int quantity, double percent)

{

\_quantity = quantity;

\_percent = percent;

}

public double GetDiscount(int quantity, double price)

{

double discount = 0;

if (quantity >= \_quantity)

discount = GetSum() \* \_percent;

return discount;

double GetSum()

{

return quantity \* price;

}

}

**Class PercentForSum**

public class PercentForSum : IDiscountStrategy

{

private double \_sum, \_percent;

public PercentForSum(double sum, double percent)

{

\_sum = sum;

\_percent = percent;

}

public double GetDiscount(int quantity, double price)

{

double discount = 0;

if (price >= \_sum)

discount = GetSum() \* \_percent;

return discount;

double GetSum()

{

return quantity \* price;

}

}

}

**Tests**

[TestFixture]

public class TestsNewMethods

{

TXTPresenter txtPresenter;

HTMLPresenter htmlPresenter;

[SetUp]

public void Setup()

{

txtPresenter = new TXTPresenter();

htmlPresenter = new HTMLPresenter();

}

[Test]

public void TestRegularItemWithoutDiscount()

{

Goods pepsi = BillFactory.Create("REG", "pepsi");

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

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

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

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 TestSpecialItemWithDiscount()

{

Goods Cola = BillFactory.Create("SPO", "Cola");

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

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

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

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 TestRegularItemAfterNewYearOffer()

{

Goods fanta = BillFactory.Create("REG", "fanta", DateTime.Parse("20.12.2020"));

Item i1 = new Item(fanta, 1, 65.99);

Customer x = new Customer("Sara", 0);

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

b1.addGoods(i1);

string actual = b1.GenerateBill();

string expected = "Счет для Sara\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tfanta\t\t65,99\t1\t65,99\t0\t65,99\t0\nСумма счета составляет 65,99\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestRegularItemWithinNewYearOffer()

{

Goods fanta = BillFactory.Create("REG", "fanta", DateTime.Parse("17.12.2020"));

Item i1 = new Item(fanta, 1, 65.99);

Customer x = new Customer("Sara", 0);

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

b1.addGoods(i1);

string actual = b1.GenerateBill();

string expected = "Счет для Sara\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tfanta\t\t65,99\t1\t65,99\t0\t65,99\t3\nСумма счета составляет 65,99\nВы заработали 3 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSpecialItemAfterNewYearOffer()

{

Goods Cola = BillFactory.Create("SPO", "Cola", DateTime.Parse("20.12.2020"));

Item i1 = new Item(Cola, 20, 0.39);

Customer x = new Customer("Ana", 0);

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

b1.addGoods(i1);

string actual = b1.GenerateBill();

string expected = "Счет для Ana\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCola\t\t0,39\t20\t7,8\t0\t7,8\t0\nСумма счета составляет 7,8\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSpecialItemWithinNewYearOffer()

{

Goods Cola = BillFactory.Create("SPO", "Cola", DateTime.Parse("17.12.2020"));

Item i1 = new Item(Cola, 20, 0.39);

Customer x = new Customer("Ana", 0);

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

b1.addGoods(i1);

string actual = b1.GenerateBill();

string expected = "Счет для Ana\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCola\t\t0,39\t20\t7,8\t0,039\t7,761\t0\nСумма счета составляет 7,761\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSaleItemAfterNewYearOffer()

{

Goods Coffee = BillFactory.Create("SAL", "Coffee", DateTime.Parse("22.12.2020"));

Item i1 = new Item(Coffee, 4, 14.50);

Customer x = new Customer("Nouha", 0);

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

b1.addGoods(i1);

string actual = b1.GenerateBill();

string expected = "Счет для Nouha\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCoffee\t\t14,5\t4\t58\t0\t58\t0\nСумма счета составляет 58\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

[Test]

public void TestSaleItemWithinNewYearOffer()

{

Goods Coffee = BillFactory.Create("SAL", "Coffee", DateTime.Parse("17.12.2020"));

Item i1 = new Item(Coffee, 4, 14.50);

Customer x = new Customer("Nouha", 0);

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

b1.addGoods(i1);

string actual = b1.GenerateBill();

string expected = "Счет для Nouha\n\tНазвание\tЦена\tКол-воСтоимость\tСкидка\tСумма\tБонус\n\tCoffee\t\t14,5\t4\t58\t0,58\t57,42\t0\nСумма счета составляет 57,42\nВы заработали 0 бонусных баллов";

Assert.AreEqual(expected, actual);

}

}

}

**By running the program:**

