Буниф Мажда

18-КБ-ПР2

**Отчет ЭПО Лаб 5**

**4 Задание**

1) Преобразовать процесс создания объектов в методе Main класса Program 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 :**

**AbstractContentFile**

public abstract class AbstractContentFile

{

public abstract void SetSource(TextReader textReader);

public abstract Customer GetCustomer();

public abstract int GetGoodsCount();

public abstract Goods GetNextGood();

public abstract int GetItemsCount();

public abstract Item GetNextItem(Goods[] good);

}

**YAMLFile**

public class YAMLFile : AbstractContentFile

{

protected string \_source;

protected TextReader sr;

public YAMLFile(TextReader source) => SetSource(source);

public override void SetSource(TextReader source) => sr = source;

private string GetNextLine()

{

string line;

do

{

line = sr.ReadLine();

} while (line.StartsWith("#"));

if (line == null) throw new Exception();

return line;

}

public override Customer GetCustomer()

{

string line = GetNextLine();

string[] result = line.Split(':');

string name = result[1].Trim();

line = sr.ReadLine();

result = line.Split(':');

int bonus = Convert.ToInt32(result[1].Trim());

return new Customer(name, bonus);

}

public override int GetGoodsCount()

{

string line = GetNextLine();

string[] result = line.Split(':');

return Convert.ToInt32(result[1].Trim());

}

public override Goods GetNextGood()

{

string line = GetNextLine();

string[] result = line.Split(':');

result = result[1].Trim().Split();

string type = result[1].Trim();

return BillFactory.Create(type, result[0]);

}

public override int GetItemsCount()

{

string line = GetNextLine();

string[] result = line.Split(':');

return Convert.ToInt32(result[1].Trim());

}

public override Item GetNextItem(Goods[] good)

{

string line = GetNextLine();

string[] result = line.Split(':');

result = result[1].Trim().Split();

int Goods\_Id = Convert.ToInt32(result[0].Trim());

double price = Convert.ToDouble(result[1].Trim());

int quantity = Convert.ToInt32(result[2].Trim());

return new Item(good[Goods\_Id-1], quantity, price);

}

**BillFactory**

public class BillFactory

{

public static Goods Create(string type, string item\_name)

{

switch(type)

{

case "REG":

return new RegularItem(item\_name);

case "SAL":

return new SalesItem(item\_name);

case "SPO":

return new SpecialItem(item\_name);

default:

throw new Exception();

}

**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;

}

}

}

**Tests**

[TestFixture]

public class TestsNewMethods

{

TXTPresenter txtPresenter;

HTMLPresenter htmlPresenter;

[SetUp]

public void Setup()

{

txtPresenter = new TXTPresenter();

htmlPresenter = new HTMLPresenter();

}

[Test]

public void TestRegularItemWithDiscount()

{

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

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

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

//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\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 = 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 TestRegularItemWithoutBonuses()

{

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

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

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

// IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

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 = BillFactory.Create("REG", "cola");

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

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

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

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 = 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 TestSpecialItemWithoutDiscount()

{

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

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

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

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

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 = BillFactory.Create("SAL", "Coffee");

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

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

//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\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 = BillFactory.Create("SAL", "OrangeJus");

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

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

//IPresenter p = new TXTPresenter();

BillGenerator b1 = new BillGenerator(x, txtPresenter);

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 program :**

