Буниф Мажда

18-КБ-ПР2

**Отчет ЭПО Лаб 3**

**4 Задание**

1. Преобразовать методы и классы программы в соответствии с предложенными в пункте 2.8 изменениями, применив описанные в теории методы рефакторинга.
2. 2) Проверить корректность работы нового метода и работоспособность формирования счета.

**Листинг :**

**RegularItem.cs**

public class RegularItem : Goods

{

public RegularItem(String title) : base(title) { }

public override int GetBonus(int qty, double price)

{

int bonus = 0;

return bonus = (int)(qty \* price \* 0.05);

}

public override double GetDiscount(int qty, double price)

{

double discount = 0.0;

if (qty > 2)

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

else

{

return 0;

}

}

}

}

**SalesItem.cs**

public class SalesItem : Goods

{

public SalesItem(String title) : base(title)

{

}

public override int GetBonus(int qty, double price)

{

int bonus = 0;

return bonus = (int)(qty \* price \* 0.01);

}

public override double GetDiscount(int qty, double price)

{

double discount = 0.0;

if (qty > 3)

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

else

return 0;

}

}

}

**SpecialItem.cs**

public class SpecialItem: Goods

{

public SpecialItem(String title) : base(title) { }

public override int GetBonus(int qty, double price)

{

return 0;

}

public override double GetDiscount(int qty, double price)

{

double discount = 0.0;

if (qty > 10)

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

else

return 0;

}

}

}

**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);

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 RegularItem("pepsi");

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 RegularItem("cola");

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 RegularItem("Cola");

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 SpecialItem("Cola");

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 SpecialItem("Mirenda");

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 SalesItem("Coffee");

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 SalesItem("OrangeJus");

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:**

