Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский Государственный Технологический Университет»

Кафедра информационных систем и программирования

Отчет по лабораторной работе №3

По дисциплине «Рефакторинг и работа с унаследованным кодом»

«Базовые техники рефакторинга. Межклассовые рефакторинги»

Выполнил студент

Группы 21-КБ-ПР2

Жулькин Д.П.

Преподаватель:

Степанова Елизавета Владимировна

Краснодар

2024

Цель работы: Изучить базовые техники рефакторинга позволяющие различным образом манипулировать методами между классами программы.

Задание:

1) Избавиться от switch в методах GetBonus и GetDiscount

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

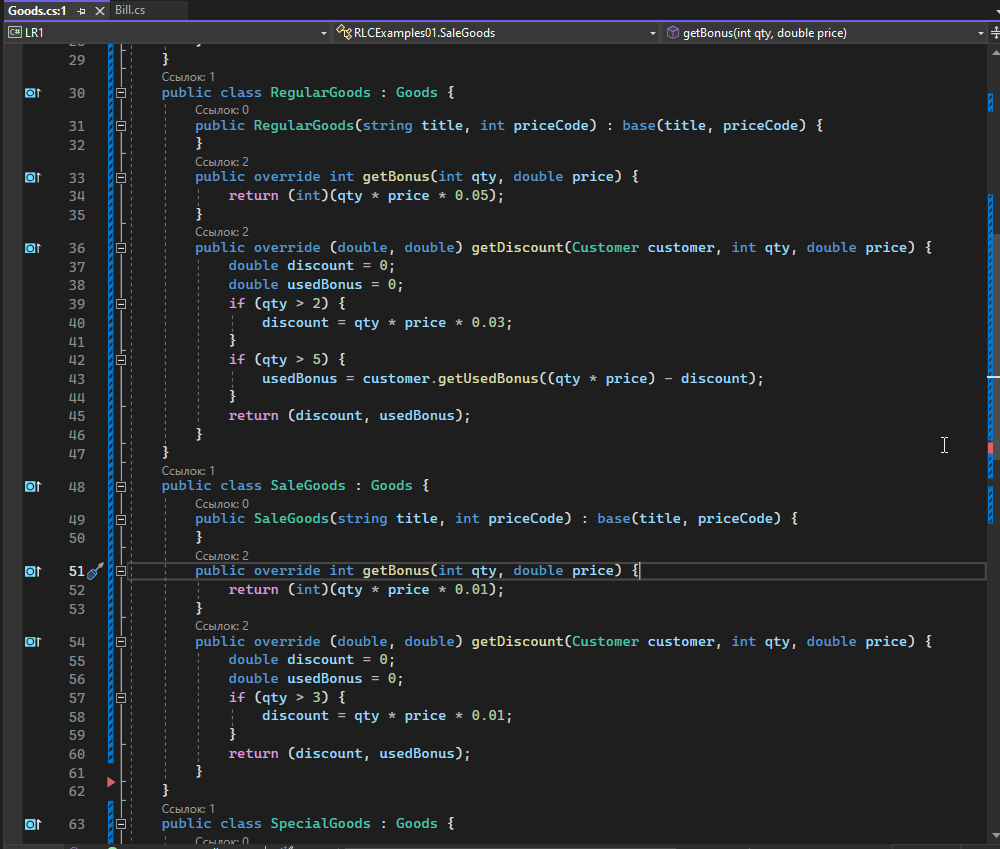


Рисунок 1 – Создание подклассов и выделение из ветвей switch.

В базовом классе private поля заменены на protected, а методы getBonus и getDiscount получили метку virtual и возвращают default.

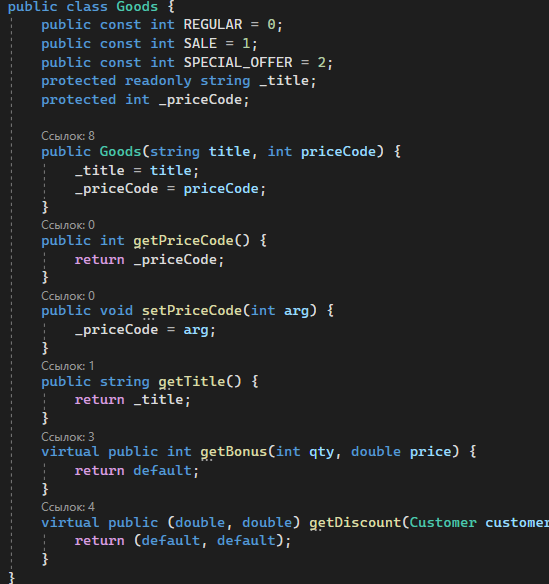


Рисунок 2 – Базовый класс.

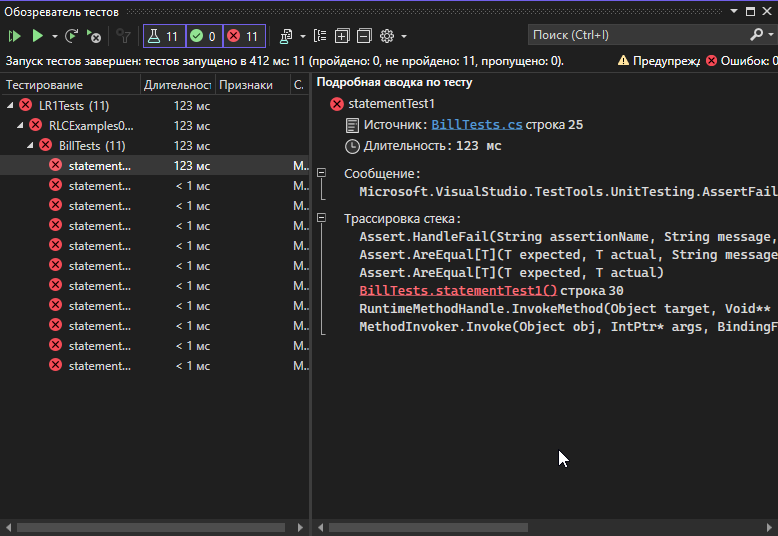


Рисунок 3 – Состояние тестов после создания подклассов.

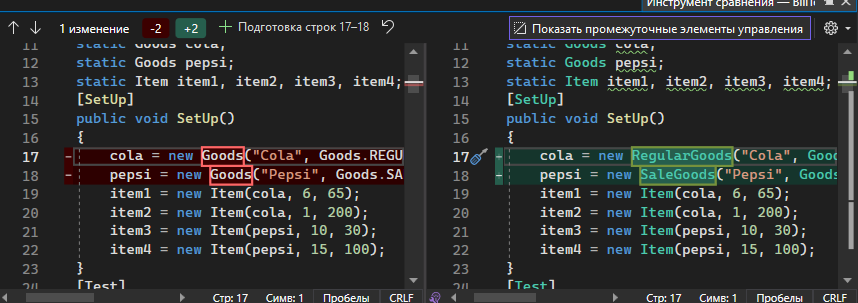


Рисунок 4 – Обновление тестов

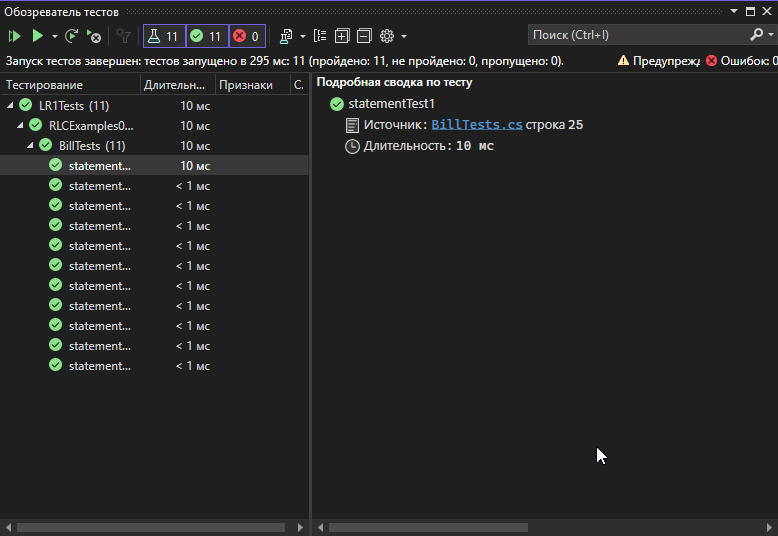


Рисунок 5 – Тесты работают

Вывод: все задания выполнены успешно, изучены базовые техники рефакторинга позволяющие различным образом манипулировать методами между классами программы.

Листинг:

namespace RLCExamples01 {

// Класс, который представляет данные о товаре

public class Goods {

protected readonly string \_title;

public Goods(string title) {

\_title = title;

}

public string getTitle() {

return \_title;

}

virtual public int getBonus(int qty, double price) {

return default;

}

virtual public (double, double) getDiscount(Customer customer, int qty, double price) {

return (default, default);

}

}

public class RegularGoods : Goods {

public RegularGoods(string title) : base(title) {

}

public override int getBonus(int qty, double price) {

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

}

public override (double, double) getDiscount(Customer customer, int qty, double price) {

double discount = 0;

double usedBonus = 0;

if (qty > 2) {

discount = qty \* price \* 0.03;

}

if (qty > 5) {

usedBonus = customer.getUsedBonus((qty \* price) - discount);

}

return (discount, usedBonus);

}

}

public class SaleGoods : Goods {

public SaleGoods(string title) : base(title) {

}

public override int getBonus(int qty, double price) {

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

}

public override (double, double) getDiscount(Customer customer, int qty, double price) {

double discount = 0;

double usedBonus = 0;

if (qty > 3) {

discount = qty \* price \* 0.01;

}

return (discount, usedBonus);

}

}

public class SpecialGoods : Goods {

public SpecialGoods(string title) : base(title) {

}

public override (double, double) getDiscount(Customer customer, int qty, double price) {

double discount = 0;

double usedBonus = 0;

if (qty > 10) {

discount = qty \* price \* 0.005;

}

if (qty > 1) {

usedBonus = customer.getUsedBonus((qty \* price) - discount);

}

return (discount, usedBonus);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RLCExamples01

{

// Класс, представляющий клиента магазина.

public class Customer

{

private int bonus;

private string name;

public Customer(string name, int bonus)

{

this.name = name;

this.bonus = bonus;

}

public string getName()

{

return name;

}

public int getBonus()

{

return bonus;

}

public void receiveBonus(int bonus)

{

this.bonus = bonus;

}

public int useBonus(int needBonus)

{

int bonusTaken;

if (needBonus > bonus)

{

bonusTaken = bonus;

bonus = 0;

}

else

{

bonusTaken = needBonus;

bonus = bonus - needBonus;

}

return bonusTaken;

}

public int getUsedBonus(double needBonus) {

return useBonus((int)needBonus);

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace RLCExamples01

{

// Класс, представляющий данные о чеке

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 getGoods().getBonus(\_quantity, \_price);

}

public (double, double) getDiscount(Customer customer) {

return getGoods().getDiscount(customer, \_quantity, \_price);

}

public double getSum() {

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

}

}

}