Министерство науки и высшего образования Российской Федерации

ФГБОУ ВО «Кубанский Государственный Технологический Университет»

Кафедра информационных систем и программирования

Отчет по лабораторной работе №5

По дисциплине «Рефакторинг и работа с унаследованным кодом»

«Сложные техники рефакторинга. Выделение модели предметной области»

Выполнил студент

Группы 21-КБ-ПР2

Жулькин Д.П.

Преподаватель:

Степанова Елизавета Владимировна

Краснодар

2024

Цель работы: Изучить техники рефакторинга позволяющие выполнять отделение модели предметной области от непосредственно процесса генерации чека.

Задание:

1) Разделение модели предметной области и процесса генерации представления.

2) Проверить корректность работы нового метода и работоспособность формирования счета

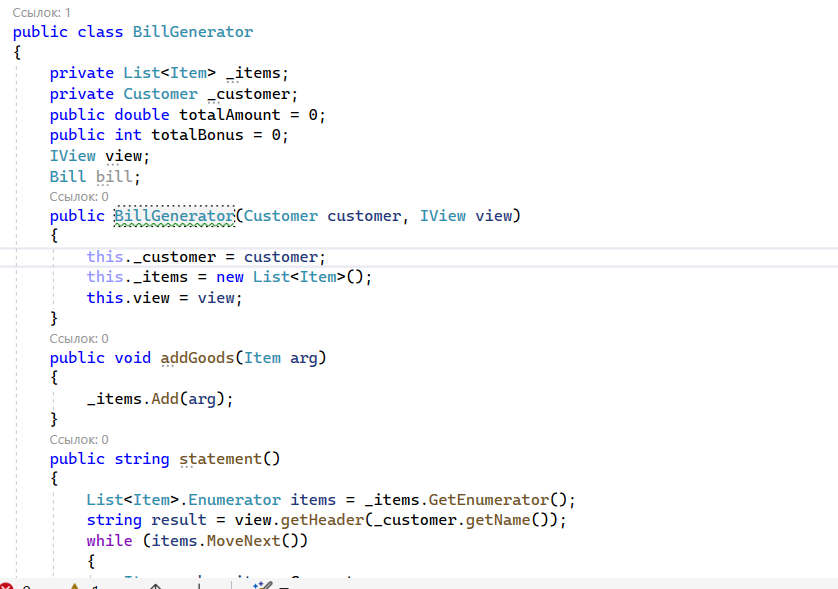


Рисунок 1 – Создание BillGenerator

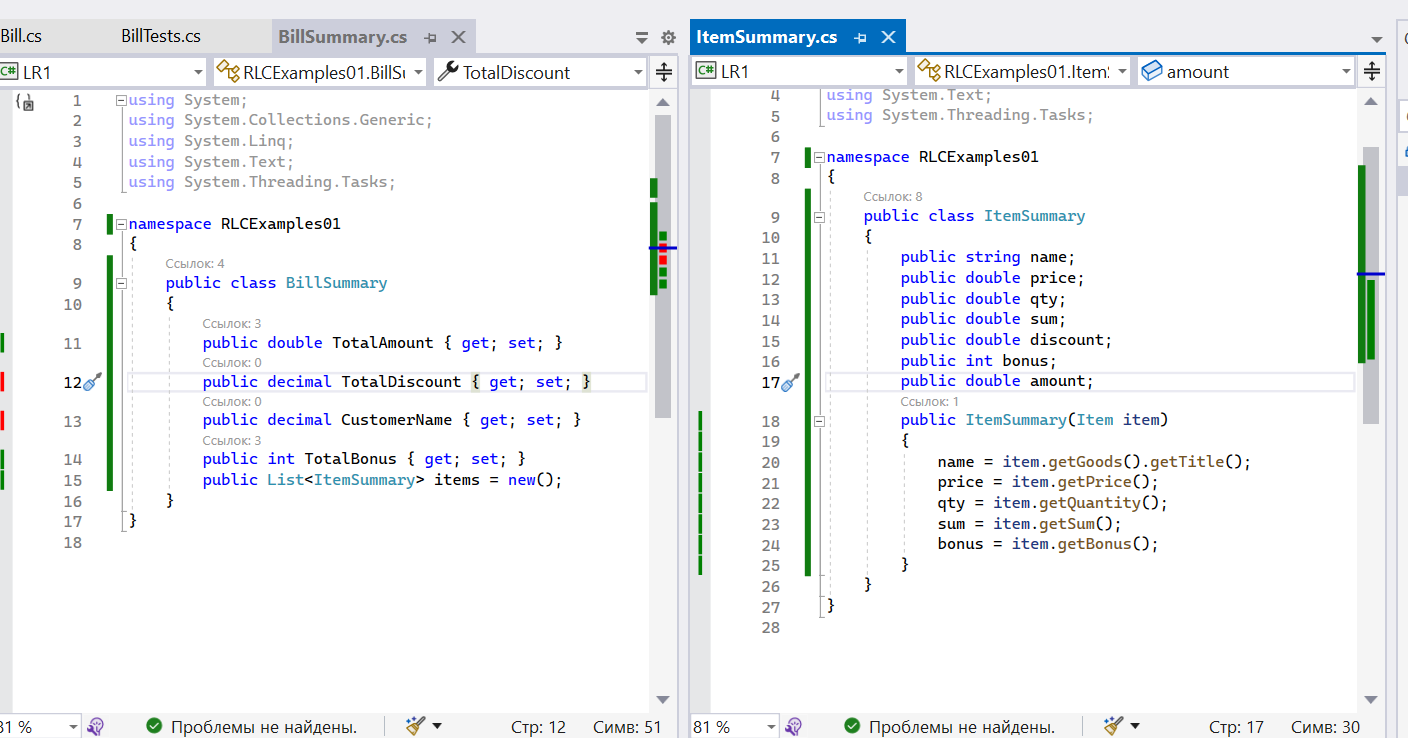


Рисунок 2 – Классы BillSummary и ItemSummary

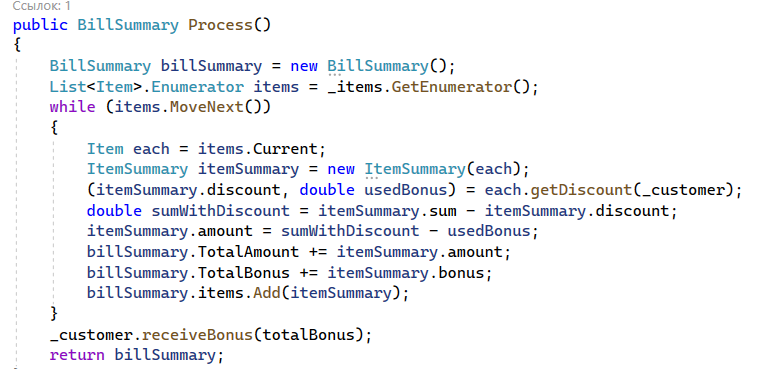


Рисунок 3 – Новый метод Process в классе Bill

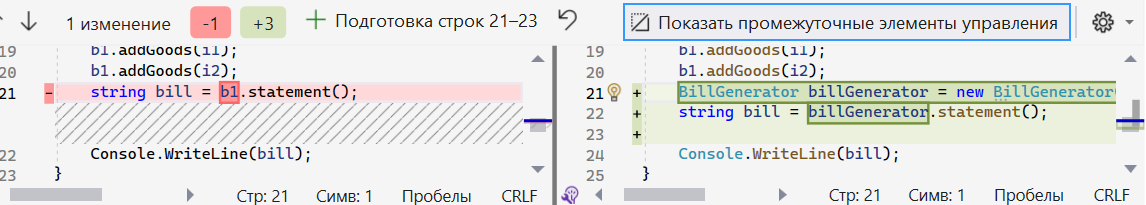


Рисунок 4 – Преобразование вызова метода statement для нового класса BillGenerator

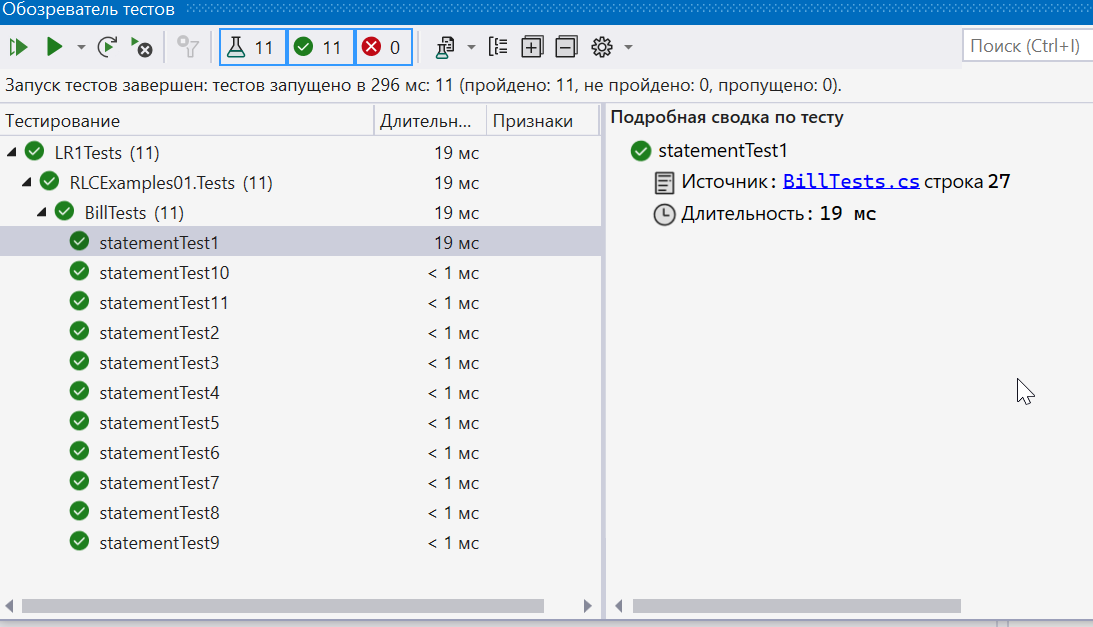


Рисунок 5 – Тесты пройдены успешно

Вывод: все задания выполнены успешно, изучены сложные техники рефакторинга позволяющие выполнять разделение представления от вычисления.

Листинг:

namespace RLCExamples01

{

public class Bill

{

private List<Item> \_items;

private Customer \_customer;

public double totalAmount = 0;

public int totalBonus = 0;

IView view;

public Bill(Customer customer, IView view)

{

this.\_customer = customer;

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

this.view = view;

}

public void addGoods(Item arg)

{

\_items.Add(arg);

}

public BillSummary Process()

{

BillSummary billSummary = new BillSummary();

List<Item>.Enumerator items = \_items.GetEnumerator();

while (items.MoveNext())

{

Item each = items.Current;

ItemSummary itemSummary = new ItemSummary(each);

(itemSummary.discount, double usedBonus) = each.getDiscount(\_customer);

double sumWithDiscount = itemSummary.sum - itemSummary.discount;

itemSummary.amount = sumWithDiscount - usedBonus;

billSummary.TotalAmount += itemSummary.amount;

billSummary.TotalBonus += itemSummary.bonus;

billSummary.items.Add(itemSummary);

}

\_customer.receiveBonus(totalBonus);

return billSummary;

}

}

public class BillGenerator

{

private List<Item> \_items;

private Customer \_customer;

public double totalAmount = 0;

public int totalBonus = 0;

IView view;

Bill bill;

public BillGenerator(Customer customer, IView view, Bill bill)

{

this.\_customer = customer;

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

this.view = view;

this.bill = bill;

}

public void addGoods(Item arg)

{

\_items.Add(arg);

}

public string statement()

{

BillSummary billSummary = bill.Process();

List<Item>.Enumerator items = \_items.GetEnumerator();

string result = view.getHeader(\_customer.getName());

foreach (ItemSummary billItem in billSummary.items)

{

result += view.getItemString(billItem);

}

\_customer.receiveBonus(totalBonus);

totalAmount = billSummary.TotalAmount; totalBonus = billSummary.TotalBonus;

result += view.getFooter(billSummary.TotalAmount, billSummary.TotalBonus);

return result;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RLCExamples01

{

public class BillSummary

{

public double TotalAmount { get; set; }

public decimal TotalDiscount { get; set; }

public decimal CustomerName { get; set; }

public int TotalBonus { get; set; }

public List<ItemSummary> items = new();

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace RLCExamples01

{

public class ItemSummary

{

public string name;

public double price;

public double qty;

public double sum;

public double discount;

public int bonus;

public double amount;

public ItemSummary(Item item)

{

name = item.getGoods().getTitle();

price = item.getPrice();

qty = item.getQuantity();

sum = item.getSum();

bonus = item.getBonus();

}

}

}