Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования

"Кубанский государственный технологический университет"

(ФГБОУ ВО "КубГТУ")

Институт КСиИБ

Кафедра ИСП

**ОТЧЁТ**

по лабораторной работе №9

в рамках освоения дисциплины

**«Рефакторинг и работа с унаследованным кодом»**

**Тема: «Выделение бизнес-логики и применение различных вариантов визуального представления»**

Обучающегося группы 21-КБ-ПР1

Направление подготовки 09.03.04 Программная инженерия

Студент Кузнецов Вадим Алексеевич

Преподаватель Степанова Елизавета Владимировна

Краснодар  
2024

**Цель работы:**

Изучить техники рефакторинга и способы совместного использования общего кода в различных приложениях.

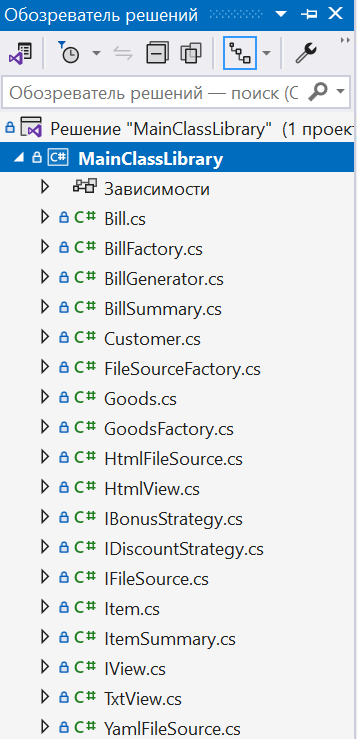
**Задание:**

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

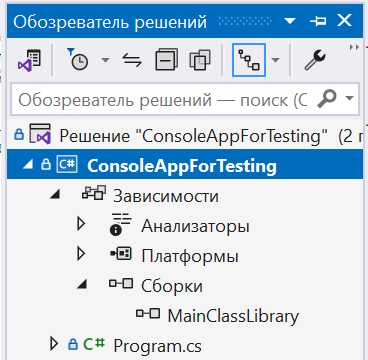
2) Проверить согласованность результатов и корректность работы нового варианта программы с результатами старой версии.

**1) Создать новый проект C# Class Library.**

**2) Перенести в него основные классы старого консольного приложения (все кроме Program).**



**3) Добавить ссылку на новую библиотеку в консольном приложении.**



**4) Добавить пространство имен новой библиотеки в консольное приложение.**

using РРУК\_01;

namespace ConsoleAppForTesting

{

internal class Program

{

static void Main(string[] args)

{

string filename = "BillInfo.html";

if (args.Length == 1)

filename = args[0];

IFileSource fileSource = FileSourceFactory.CreateFileSource(filename);

using (FileStream fs = new FileStream(filename, FileMode.Open))

using (StreamReader sr = new StreamReader(fs))

{

BillFactory factory = new BillFactory(fileSource);

BillGenerator bill = factory.CreateBill(sr, "NewYearsSettings.json",new HtmlView());

string billOutput = bill.GenerateBill();

Console.WriteLine(billOutput);

}

}

}

}

**5) Обновить ссылки на тестируемые классы в тестовом проекте.**

using РРУК\_01;

namespace TestingTheLoyaltyProgram

{

public class Tests

{

private string nameHtmlSource = "BillInfo.html";

private string nameYamlSource = "BillInfo.yaml";

private IFileSource fileSourceHtml;

private IFileSource fileSourceYaml;

private BillFactory billFactoryHtml;

private BillFactory billFactoryYaml;

[SetUp]

public void Setup()

{

IView view = new TxtView();

fileSourceHtml = FileSourceFactory.CreateFileSource(nameHtmlSource);

billFactoryHtml = new BillFactory(fileSourceHtml);

fileSourceYaml = FileSourceFactory.CreateFileSource(nameYamlSource);

billFactoryYaml = new BillFactory(fileSourceYaml);

}

[Test]

public void NewYearDiscountAndBonusForRegularGoods()

{

// Предположим, что даты предновогодних недель учитываются внутри стратегий

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 1\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Milk REG\r\n" +

"ItemsTotalCount: 1\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 5000 1"; // Покупка на 5000 рублей

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactoryYaml.CreateBill(sr, "NewYearsSettings.json");

var result = billGenerator.GenerateBill();

// Ожидаем, что скидка не применяется, но бонусы начисляются 7% от 5000

Assert.IsTrue(result.Contains("Сумма счета составляет 5000"));

Assert.IsTrue(result.Contains("Вы заработали 350 бонусных балов"));

}

}

[Test]

public void UsualDiscountAndBonusForRegularGoods()

{

// Предположим, что даты предновогодних недель учитываются внутри стратегий

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 1\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Razor\_blades REG\r\n" +

"ItemsTotalCount: 1\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 5000 1"; // Покупка на 5000 рублей

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactoryYaml.CreateBill(sr, "RegularSettings.json");

var result = billGenerator.GenerateBill();

// Ожидаем, что скидка не применяется, но бонусы начисляются 5% от 5000

Assert.IsTrue(result.Contains("Сумма счета составляет 5000"));

Assert.IsTrue(result.Contains("Вы заработали 250 бонусных балов"));

}

}

[Test]

public void NewYearDiscountForSaleGoods()

{

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 1\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: Christmas\_Tree SAL\r\n" +

"ItemsTotalCount: 1\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 2000 2"; // Покупка на 4000 рублей

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactoryYaml.CreateBill(sr, "NewYearsSettings.json");

var result = billGenerator.GenerateBill();

// Ожидаем скидку 3% на товары со скидкой при покупке более чем на 2000 руб.

Assert.IsTrue(result.Contains("Сумма счета составляет 3880,00")); // 4000 - 3% скидка

Assert.IsTrue(result.Contains("Вы заработали 40 бонусных балов")); // 1% от 4000

}

}

[Test]

public void NewYearDiscountForSpecialGoods()

{

string testData = "CustomerName: Test\r\n" +

"CustomerBonus: 10\r\n" +

"GoodsTotalCount: 1\r\n" +

"# ID: NAME TYPE(REG/SAL/SPO)\r\n" +

"1: New\_Year\_Candy SPO\r\n" +

"ItemsTotalCount: 1\r\n" +

"# ID: GID PRICE QTY\r\n" +

"1: 1 300 11"; // Покупка на 3300 рублей

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactoryYaml.CreateBill(sr, "NewYearsSettings.json");

var result = billGenerator.GenerateBill();

// Ожидаем скидку 5% на акционные товары при покупке более чем на 3000 руб.

Assert.IsTrue(result.Contains("Сумма счета составляет 3125,00")); // 3300 - 5% скидка

Assert.IsTrue(result.Contains("Вы заработали 0 бонусных балов")); // Нет бонусов для акционных товаров

}

}

[Test]

public void Html\_NewYearDiscountForThreeGoods()

{

string testData = "<html>\r\n<body>\r\n " +

"<customerName>Test</customerName>\r\n " +

"<customerBonus>10</customerBonus>\r\n " +

"<goodsCount>3</goodsCount>\r\n " +

"<good>\r\n " +

"<goodType>REG</goodType>\r\n " +

"<goodName>Cola</goodName>\r\n " +

"<goodType>SAL</goodType>\r\n " +

"<goodName>Pepsi</goodName>\r\n " +

"<goodType>SPO</goodType>\r\n " +

"<goodName>Fanta</goodName>\r\n " +

"</good>\r\n " +

"<itemsCount>3</itemsCount>\r\n " +

"<item>\r\n " +

"<itemId>1</itemId>\r\n " +

"<itemPrice>90</itemPrice>\r\n " +

"<itemQty>60</itemQty>\r\n" +

"<itemId>2</itemId>\r\n " +

"<itemPrice>50</itemPrice>\r\n " +

"<itemQty>44</itemQty>\r\n\r\n " +

"<itemId>3</itemId>\r\n " +

"<itemPrice>76</itemPrice>\r\n " +

"<itemQty>40</itemQty>\r\n " +

"</item>\r\n" +

"</body>\r\n" +

"</html>\r\n"; // Покупка на 3300 рублей

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactoryHtml.CreateBill(sr, "NewYearsSettings.json");

var result = billGenerator.GenerateBill();

// Ожидаем скидку 5% на акционные товары при покупке более чем на 3000 руб.

Assert.IsTrue(result.Contains("Сумма счета составляет 10250,00")); // 3300 - 5% скидка

Assert.IsTrue(result.Contains("Вы заработали 400 бонусных балов")); // Нет бонусов для акционных товаров

}

}

[Test]

public void Html\_RegularDiscountForThreeGoods()

{

string testData = "<html>\r\n<body>\r\n " +

"<customerName>Test</customerName>\r\n " +

"<customerBonus>10</customerBonus>\r\n " +

"<goodsCount>3</goodsCount>\r\n " +

"<good>\r\n " +

"<goodType>REG</goodType>\r\n " +

"<goodName>Cola</goodName>\r\n " +

"<goodType>SAL</goodType>\r\n " +

"<goodName>Pepsi</goodName>\r\n " +

"<goodType>SPO</goodType>\r\n " +

"<goodName>Fanta</goodName>\r\n " +

"</good>\r\n " +

"<itemsCount>3</itemsCount>\r\n " +

"<item>\r\n " +

"<itemId>1</itemId>\r\n " +

"<itemPrice>90</itemPrice>\r\n " +

"<itemQty>60</itemQty>\r\n" +

"<itemId>2</itemId>\r\n " +

"<itemPrice>50</itemPrice>\r\n " +

"<itemQty>44</itemQty>\r\n\r\n " +

"<itemId>3</itemId>\r\n " +

"<itemPrice>76</itemPrice>\r\n " +

"<itemQty>40</itemQty>\r\n " +

"</item>\r\n" +

"</body>\r\n" +

"</html>\r\n"; // Покупка на 3300 рублей

using (StringReader sr = new StringReader(testData))

{

BillGenerator billGenerator = billFactoryHtml.CreateBill(sr, "RegularSettings.json");

var result = billGenerator.GenerateBill();

// Ожидаем скидку 5% на акционные товары при покупке более чем на 3000 руб.

Assert.IsTrue(result.Contains("Сумма счета составляет 10430,800")); // 3300 - 5% скидка

Assert.IsTrue(result.Contains("Вы заработали 292 бонусных балов")); // Нет бонусов для акционных товаров

}

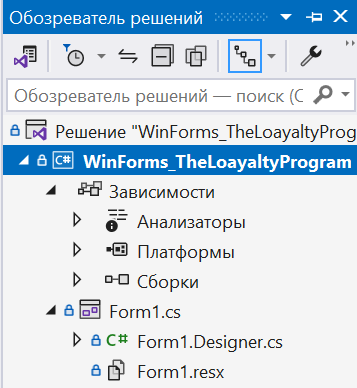
}

}

}

**6) Создать новый проект оконного приложения C# Windows Forms Application.**

**7) Добавить ссылку на новую библиотеку в оконном приложении.**



**8) Добавить пространство имен новой библиотеки в оконное приложение.**

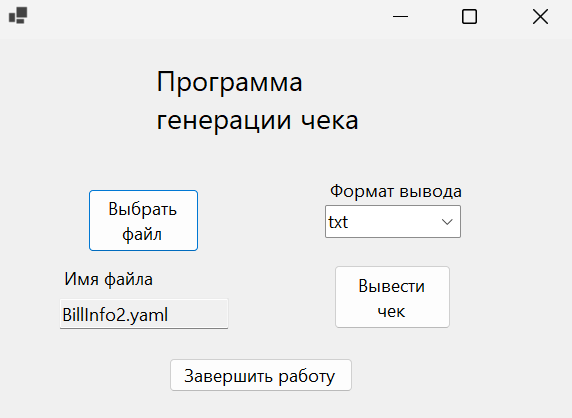
using РРУК\_01;

namespace WinForms\_TheLoayaltyProgram

{

public partial class Form1 : Form

**9) Добавить необходимые визуальные компоненты (диалоги для выбора загружаемого файла и результатов работы, поле для вывода результатов генерации чека, кнопки управления работой (запуск генерации, выход и т.д.), переключатели для выбора формата вывода (HTML/TXT) и т.д.)**

****

**10) Связать визуальные компоненты с кодом библиотеки.**

using РРУК\_01;

namespace WinForms\_TheLoayaltyProgram

{

public partial class Form1 : Form

{

string billOutput;

string filename;

IView view;

public Form1()

{

InitializeComponent();

comboBox.Items.Add("txt");

comboBox.Items.Add("html");

comboBox.SelectedIndex = 0;

}

public void Proccess()

{

if (comboBox.Text == "html")

{

view = new HtmlView();

}

else if (comboBox.Text == "txt")

{

view = new TxtView();

}

else

{

MessageBox.Show("Выберите формат файла", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

if(filename != null)

{

IFileSource fileSource = FileSourceFactory.CreateFileSource(filename);

using (FileStream fs = new FileStream(filename, FileMode.Open))

using (StreamReader sr = new StreamReader(fs))

{

if (view != null)

{

BillFactory factory = new BillFactory(fileSource);

BillGenerator bill = factory.CreateBill(sr, "NewYearsSettings.json", view);

billOutput = bill.GenerateBill();

if (view.GetType() == typeof(HtmlView))

{

HtmlForm htmlForm = new HtmlForm(billOutput);

htmlForm.Show();

}

else

{

TxtForm txtForm = new TxtForm(billOutput);

txtForm.Show();

}

}

}

}

else

{

MessageBox.Show("Выберите файл","Файл не выбран", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void ShowBill\_Click(object sender, EventArgs e)

{

Proccess();

}

private void button\_Exit\_Click(object sender, EventArgs e)

{

Close();

}

private void button\_ToSelect\_Click(object sender, EventArgs e)

{

OpenFileDialog openFileDialog = new OpenFileDialog

{

InitialDirectory = @"D:\GIT\RRUK\Labs\Ninth lab\WinForms\_TheLoayaltyProgram\WinForms\_TheLoayaltyProgram\bin\Debug\net8.0-windows",

Filter = "Yaml файлы (\*.yaml)|\*.yaml|HTML файлы (\*.html;\*.htm)|\*.html;\*.htm",

FilterIndex = 1,

RestoreDirectory = true

};

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

// Получаем путь к выбранному файлу

string filePath = openFileDialog.FileName;

filename = Path.GetFileName(filePath);

TB\_FileName.Text = filename;

// Здесь можно выполнить операции с выбранным файлом

MessageBox.Show($"Выбраный файл: {filename}", "Файл выбран", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

**Контрольные вопросы**

**1) Что такое сборка?**

Сборка в контексте .NET является компилированным кодом, который исполняется средой CLR (Common Language Runtime). Она содержит один или несколько файлов, которые могут включать управляемый код (.dll или .exe), ресурсы, типы данных и метаданные, описывающие содержимое сборки. Сборки являются основным строительным блоком приложений .NET и служат единицей развертывания, версионности и безопасности.

**2) Как создаются динамические библиотеки?**

Динамические библиотеки (DLL - Dynamic Link Libraries) в .NET создаются путем компиляции кода с использованием компилятора языка, такого как csc.exe для C# или vbc.exe для VB.NET, с указанием ключа компиляции для создания библиотеки (например, /target:library для csc.exe). В проектах Visual Studio можно создать проект типа "Class Library", который автоматически настроен на создание DLL. После компиляции проекта генерируется файл DLL, который содержит скомпилированный код и может быть использован в других приложениях.

**3) Как связать и использовать библиотеку в своем приложении?**

Для использования динамической библиотеки в своем приложении необходимо сначала добавить ссылку на нее. В Visual Studio это делается путем добавления ссылки на сборку в проект через контекстное меню "Ссылки" (References) в обозревателе решений. После добавления ссылки на DLL можно использовать ее типы, методы и свойства в коде, добавив соответствующий оператор using (C#) или Imports (VB.NET) с именем пространства имен библиотеки. Компилятор учитывает информацию о сборке при сборке приложения, а CLR загружает и связывает сборку во время выполнения.

**Заключение**

В результате выполнения лабораторной работы были изучены техники рефакторинга и способы совместного использования общего кода в различных приложениях.