Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №5

на тему:

**«Использование коллекций»**

БГУИР 6-05-0612-02 113

|  |
| --- |
| Выполнил студент группы 453503  ХАЛАМОВ Николай Андреевич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2025

# 1 Индивидуальное задание

**Задание 1. Вариант 3.** Предметная область: ЖЭС. В ЖЭС хранятся тарифы на коммунальные услуги. ЖЭС имеет информацию обо всех жильцах. При потреблении жильцами коммунальных услуг информация регистрируется в системе.

Система должна позволять выполнять следующие задачи:

– ввод тарифов;

– ввод информации о жильцах и потребленных услугах;

– после ввода фамилии, выводить сумму всех потребленных услуг;

– выводить стоимость всех оказанных услуг.

# 2 Выполнение работы

Перед выполнением работы следует разработать диаграмму классов для наглядного выполнения поставленной задачи (см. рисунок 1).

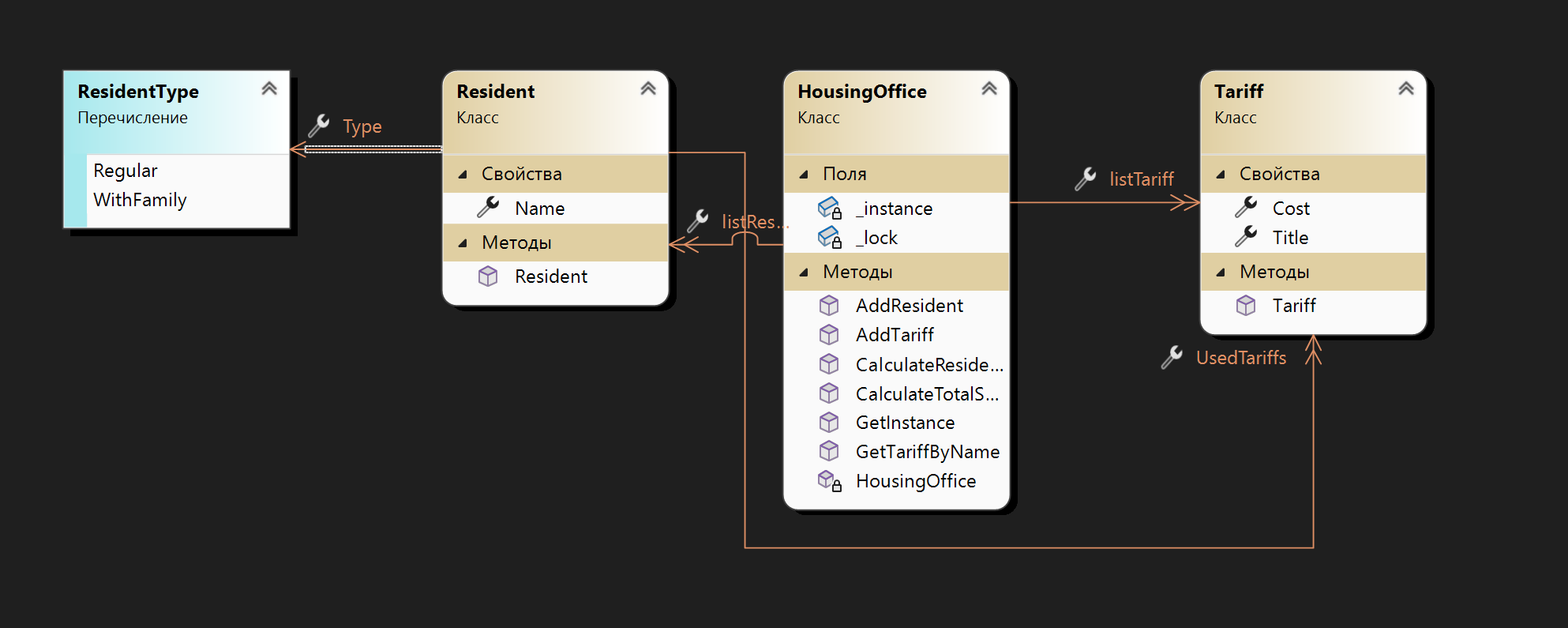


Рисунок 1 – Диаграмма классов

Для выполнения задания были созданы классы HousingOffice, Tariff, Resident, Demonstration(cм. рисунок 2).

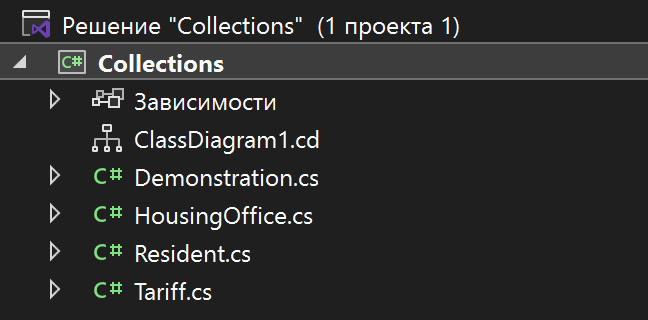


Рисунок 2 – Обозреватель решений

Мы реализовали класс Resident с перечислением ResidentType и полями, описывающими используемые тарифы, имя. Добавили геттеры и сеттеры.

namespace Collections

{

public enum ResidentType

{

Regular,

WithFamily,

}

internal class Resident

{

public string Name { get; }

public ResidentType Type { get; }

public List<Tariff> UsedTariffs { get; } = new List<Tariff>();

public Resident(string name, ResidentType type)

{

Name = name;

Type = type;

}

}

}

Взглянем на реализацию класса Tariff, у которого есть поля Title и Cost, которые отвечают за хранение имени тарифа и его стоимости соответственно.

namespace Collections

{

internal class Tariff

{

public double Cost { get; }

public string Title { get; }

public Tariff(double cost, string title)

{

Cost = cost;

Title = title;

}

}

}

Мы реализовали класс HousingOffice где добавили все методы для работы программы, требуемые по условию. Использовали паттерн проектирования Singleton. Разделили структуру программы на отдельные методы с понятным функционалом.

namespace Collections

{

internal class HousingOffice

{

private static HousingOffice \_instance;

private static readonly object \_lock = new object();

public List<Tariff> listTariff { get; set; }

public List<Resident> listResident { get; set; }

private HousingOffice()

{

listTariff = new List<Tariff>();

listResident = new List<Resident>();

}

public static HousingOffice GetInstance()

{

if (\_instance == null)

{

lock (\_lock)

{

if (\_instance == null)

{

\_instance = new HousingOffice();

}

}

}

return \_instance;

}

public void AddResident(Resident resident)

{

listResident.Add(resident);

}

public void AddTariff(Tariff tariff)

{

listTariff.Add(tariff);

}

public double CalculateResidentSum(string residentName)

{

var resident = listResident.FirstOrDefault(r =>

r.Name.Equals(residentName, StringComparison.OrdinalIgnoreCase));

return resident?.UsedTariffs.Sum(t => t.Cost) ?? 0;

}

public double CalculateTotalSum()

{

return listResident.Sum(r => r.UsedTariffs.Sum(t => t.Cost));

}

public Tariff GetTariffByName(string name)

{

return listTariff.FirstOrDefault(t =>

t.Title.Equals(name, StringComparison.OrdinalIgnoreCase));

}

}

Покажем принцип работы этой программы в классе Demonstration. Используя шаблон проектирования Singleton создаём объект нашего класса и реализуем принцип работы программы вместе с меню для взаимодействия с пользователем и проверками на ввод.

namespace Collections

{

internal class Demonstration

{

static void Main(string[] args)

{

var housingOffice = HousingOffice.GetInstance();

while (true)

{

Console.Clear();

Console.WriteLine("Меню:");

Console.WriteLine("0 - Информация о задании");

Console.WriteLine("1 - Добавить услугу");

Console.WriteLine("2 - Добавить жильца");

Console.WriteLine("3 - Сумма услуг жильца");

Console.WriteLine("4 - Общая сумма услуг");

Console.WriteLine("5 - Список тарифов");

Console.WriteLine("6 - Список жильцов");

Console.WriteLine("7 - Секрет");

Console.WriteLine("8 - Выход");

var choice = GetMenuChoice();

switch (choice)

{

case 0: ShowTaskInfo(); break;

case 1: AddTariff(housingOffice); break;

case 2: AddResident(housingOffice); break;

case 3: ShowResidentSum(housingOffice); break;

case 4: ShowTotalSum(housingOffice); break;

case 5: ShowTariffs(housingOffice); break;

case 6: ShowResidents(housingOffice); break;

case 7: ShowSecret(); break;

case 8: return;

}

Console.WriteLine("\nНажмите любую клавишу для продолжения...");

Console.ReadKey();

}

}

static int GetMenuChoice()

{

while (true)

{

Console.Write("\nВыберите пункт (0-8): ");

if (int.TryParse(Console.ReadLine(), out int choice) && choice >= 0 && choice <= 8)

return choice;

Console.WriteLine("Ошибка: введите число от 0 до 8");

}

}

static void ShowTaskInfo()

{

Console.WriteLine("Предметная область: ЖЭС.\n" +

"В ЖЭС хранятся тарифы на коммунальные услуги.\n" +

"Система позволяет:\n" +

"- Вводить тарифы\n" +

"- Добавлять жильцов\n" +

"- Рассчитывать суммы платежей\n\n" +

"Лабораторная работа №5. Выполнил студент группы 453503 Халамов Н.");

}

static void AddTariff(HousingOffice housingOffice)

{

string title;

while (true)

{

Console.Write("Введите название услуги: ");

title = Console.ReadLine().Trim();

if (!string.IsNullOrWhiteSpace(title))

break;

Console.WriteLine("Ошибка: название не может быть пустым");

}

double cost;

while (true)

{

Console.Write("Введите стоимость: ");

if (double.TryParse(Console.ReadLine(), out cost) && cost >= 0)

break;

Console.WriteLine("Ошибка: введите корректную стоимость (число >= 0)");

}

if (housingOffice.listTariff.Any(t => t.Title.Equals(title, StringComparison.OrdinalIgnoreCase)))

{

Console.WriteLine($"Ошибка: тариф '{title}' уже существует");

return;

}

housingOffice.AddTariff(new Tariff(cost, title));

Console.WriteLine($"Тариф '{title}' успешно добавлен");

}

static void AddResident(HousingOffice housingOffice)

{

string name;

while (true)

{

Console.Write("Введите фамилию жильца: ");

name = Console.ReadLine()?.Trim();

if (!string.IsNullOrWhiteSpace(name))

break;

Console.WriteLine("Ошибка: фамилия не может быть пустой");

}

if (housingOffice.listResident.Any(r => r.Name.Equals(name, StringComparison.OrdinalIgnoreCase)))

{

Console.WriteLine($"Ошибка: жилец '{name}' уже существует");

return;

}

Console.Write("Тип жильца (1-Regular, иначе-WithFamily): ");

var type = Console.ReadLine() == "1" ? ResidentType.Regular : ResidentType.WithFamily;

var resident = new Resident(name, type);

AddTariffsToResident(housingOffice, resident);

housingOffice.AddResident(resident);

Console.WriteLine($"Жилец '{name}' успешно добавлен");

}

static void AddTariffsToResident(HousingOffice housingOffice, Resident resident)

{

if (housingOffice.listTariff.Count == 0)

{

Console.WriteLine("Нет доступных тарифов для добавления");

return;

}

while (true)

{

Console.WriteLine("\nДоступные тарифы:");

housingOffice.listTariff.ForEach(t => Console.WriteLine($"- {t.Title}"));

Console.Write("Добавить тариф? (1-Да, иначе-Нет): ");

if (Console.ReadLine() != "1") break;

Console.Write("Введите название тарифа: ");

var tariffName = Console.ReadLine()?.Trim();

var tariff = housingOffice.GetTariffByName(tariffName);

if (tariff == null)

{

Console.WriteLine("Тариф не найден");

continue;

}

resident.UsedTariffs.Add(tariff);

Console.WriteLine($"Тариф '{tariff.Title}' добавлен");

}

}

static void ShowResidentSum(HousingOffice housingOffice)

{

Console.Write("Введите фамилию жильца: ");

var name = Console.ReadLine()?.Trim();

if (string.IsNullOrWhiteSpace(name))

{

Console.WriteLine("Ошибка: фамилия не может быть пустой");

return;

}

var sum = housingOffice.CalculateResidentSum(name);

Console.WriteLine($"Сумма услуг для {name}: {sum} руб.");

}

static void ShowTotalSum(HousingOffice housingOffice)

{

Console.WriteLine($"Общая сумма всех услуг: {housingOffice.CalculateTotalSum()} руб.");

}

static void ShowTariffs(HousingOffice housingOffice)

{

if (housingOffice.listTariff.Count == 0)

{

Console.WriteLine("Список тарифов пуст");

return;

}

Console.WriteLine("Список тарифов:");

housingOffice.listTariff.ForEach(t =>

Console.WriteLine($"- {t.Title}: {t.Cost} руб."));

}

static void ShowResidents(HousingOffice housingOffice)

{

if (housingOffice.listResident.Count == 0)

{

Console.WriteLine("Список жильцов пуст");

return;

}

Console.WriteLine("Список жильцов:");

foreach (var r in housingOffice.listResident)

{

Console.WriteLine($"\n{r.Name} ({r.Type})");

if (r.UsedTariffs.Count > 0)

{

Console.WriteLine("Тарифы:");

r.UsedTariffs.ForEach(t => Console.WriteLine($" - {t.Title}: {t.Cost} руб."));

}

else

{

Console.WriteLine("Нет назначенных тарифов");

}

}

}

static void ShowSecret()

{

Console.WriteLine("1 апреля близко...");

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 3).

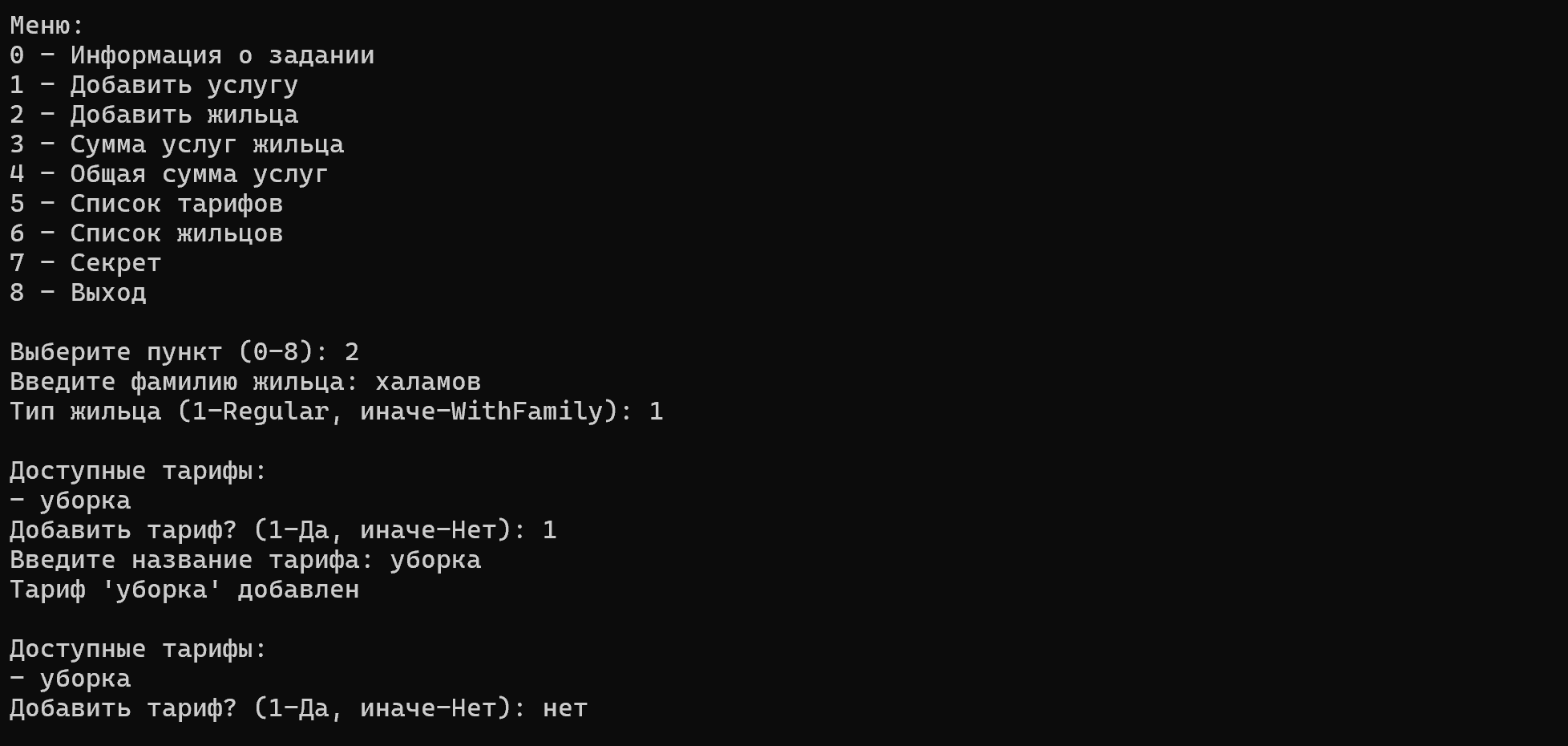


Рисунок 3 – Результат работы программы

# Вывод

В ходе лабораторной работы были изучены принципы построения диаграмм классов, использования коллекции List<T> и получены навыки проектирования приложения, состоящих из нескольких взаимосвязанных классов.