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 67

|  |
| --- |
| Выполнил студент группы 353503  КОХАН Артём Игоревич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2024

# 1 Индивидуальное задание

**Задание 1. Вариант 7.** Предметная область: Фирма грузоперевозок.

Фирма имеет список тарифов по перевозке грузов. Клиент регистрируется в системе, после чего может заказать перевозку определенного объема груза.

Система должна позволять выполнять следующие задачи:

– ввод тарифов;

– регистрация клиента и заказ на перевозку грузов;

– вывод суммы заказа для определенного клиента;

– подсчет суммарной стоимости всех заказов.

# 2 Выполнение работы

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

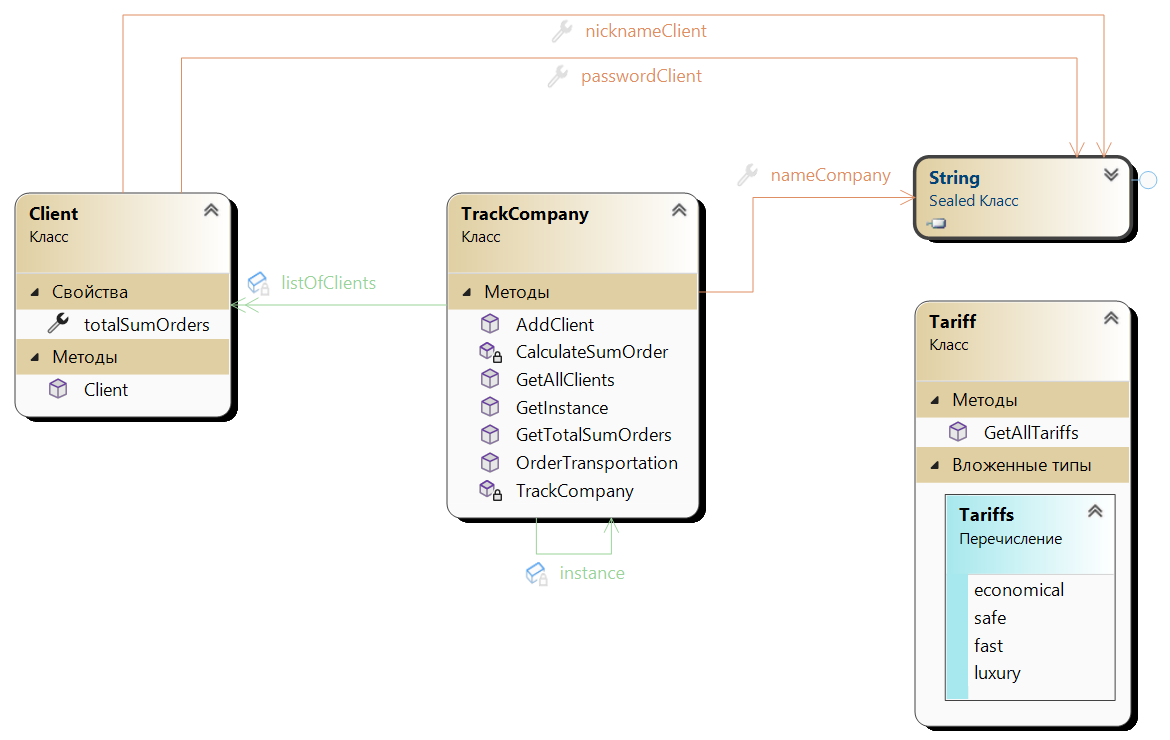


Рисунок 1 – Диаграмма классов

Для выполнения задания в проект была добавлена папка Classes, в которой находятся классы Client, Tariff, TrackCompany (cм. рисунок 2).

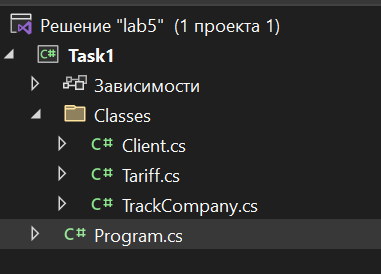


Рисунок 2 – Обозреватель решений

Рассмотрим реализацию класса Client, где имеются поля nicknameClient, passwordClient, totalSumOrders. Эти поля отвечают за хранение ника, пароля и общей суммы заказов клиента соответственно. Общую сумму заказов по умолчанию инициализируем нулём.

namespace Task1.Classes

{

internal class Client

{

public string nicknameClient { get; private set; }

public string passwordClient { get; private set; }

public double totalSumOrders { get; set; } = 0;

public Client(string nicknameClient, string passwordClient)

{

this.nicknameClient = nicknameClient;

this.passwordClient = passwordClient;

}

}

}

Взглянем на реализацию класса Tariff, в котором был использован тип данных – перечисление. Этот тип данных был использован для перечисления всех тарифов нашей фирмы. Помимо этого, в классе имеется статический метод, который выводит на экран список тарифов.

namespace Task1.Classes

{

internal class Tariff

{

public enum Tariffs : byte

{

economical = 10,

safe = 20,

fast = 25,

luxury = 50

}

public static void GetAllTariffs()

{

string[] enumTariffs = Enum.GetNames(typeof(Tariffs));

for (byte i = 0; i < enumTariffs.Length; ++i)

{

Console.WriteLine($"{i + 1}. " + enumTariffs.ElementAt(i));

}

}

}

}

Рассмотрим реализацию класса TrackCompany, в котором применен шаблон проектирования Singleton, который гарантирует, что для класса будет создан только один объект. Добавление клиента происходит через метод AddClient, который принимает в параметры ник и пароль клиента, при это проверяет, чтобы не существовало клиента с таким же ником. Для создания заказа был создан метод OrderTransportation, если данные введены корректно, то высчитывает сумму заказа используя метод CalculateSumOrder. Для получения общей суммы, потраченной на заказы, был реализован метод GetTotalSumOrders. Метод в параметры принимает ник и пароль клиента. Для получения полного списка клиентов компании реализован метод GetAllClients.

namespace Task1.Classes

{

internal class TrackCompany

{

private static TrackCompany? instance;

public string nameCompany { get; private set; }

private readonly List<Client> listOfClients;

private TrackCompany(string nameCompany)

{

this.nameCompany = nameCompany;

listOfClients = new List<Client>();

}

public void AddClient(string nicknameClient, string passwordClient)

{

bool clientExists = false;

for (int i = 0; i < listOfClients.Count; ++i)

{

if (nicknameClient == listOfClients.ElementAt(i).nicknameClient)

{

Console.WriteLine("The client already exists");

clientExists = true;

}

}

if (!clientExists)

{

listOfClients.Add(new Client(nicknameClient, passwordClient));

Console.WriteLine("The client was successfully added");

}

}

public void OrderTransportation(string nicknameClient, string passwordClient,

double loadWeightKg, Tariff.Tariffs tariff)

{

for (int i = 0; i < listOfClients.Count; ++i)

{

if (nicknameClient == listOfClients.ElementAt(i).nicknameClient)

{

if (passwordClient == listOfClients.ElementAt(i).passwordClient)

{

Console.WriteLine("Order sum: " + CalculateSumOrder(loadWeightKg, tariff));

listOfClients.ElementAt(i).totalSumOrders += CalculateSumOrder(loadWeightKg,

tariff);

Console.WriteLine("Successfully");

break;

}

else

{

Console.WriteLine("Wrong password");

break;

}

}

else if (i == listOfClients.Count - 1)

{

Console.WriteLine("The client was not found");

}

}

}

private static double CalculateSumOrder(double loadWeightKg, Tariff.Tariffs tariff) =>

loadWeightKg \* ((byte)tariff);

public void GetTotalSumOrders(string nicknameClient, string passwordClient)

{

for (int i = 0; i < listOfClients.Count; ++i)

{

if (nicknameClient == listOfClients.ElementAt(i).nicknameClient)

{

if (passwordClient == listOfClients.ElementAt(i).passwordClient)

{

Console.WriteLine(listOfClients.ElementAt(i).totalSumOrders);

break;

}

else

{

Console.WriteLine("Wrong password");

break;

}

}

else if(i == listOfClients.Count - 1)

{

Console.WriteLine("Wrong nickname");

break;

}

}

}

public void GetAllClients()

{

for (int i = 0; i < listOfClients.Count; ++i)

{

Console.WriteLine($"{i + 1}. " + listOfClients.ElementAt(i).nicknameClient);

}

}

public static TrackCompany GetInstance(string nameCompany) =>

instance ??= new TrackCompany(nameCompany);

}

}

Покажем принцип работы этой программы в классе Program. Используя шаблон проектирования Singleton создаём объект нашего класса TrackCompany и реализуем принцип работы программы.

namespace Task1

{

internal class Program

{

static void Main(string[] args)

{

TrackCompany myCompany = TrackCompany.GetInstance("Cargo");

myCompany.AddClient("Artyom", "1234");

myCompany.AddClient("Timka", "timka1958");

Console.WriteLine("\nList of tariffs:");

Tariff.GetAllTariffs();

Console.WriteLine("\nList of clients:");

myCompany.GetAllClients();

Console.WriteLine("\nLet's make orders:");

myCompany.OrderTransportation("Artyom", "1234", 5.04, Tariff.Tariffs.economical);

myCompany.OrderTransportation("Artyom", "1234", 128.99, Tariff.Tariffs.luxury);

myCompany.OrderTransportation("Timka", "timka1958", 666.666666, Tariff.Tariffs.safe);

Console.WriteLine("\nLet's look at the total amount of orders:");

myCompany.GetTotalSumOrders("Artyom", "1234");

Console.WriteLine("\nLet's check out classes:");

myCompany.AddClient("Artyom", "simplePassoword1234");

myCompany.AddClient("Artyom", "1234");

myCompany.OrderTransportation("Artyom", "123", 51, Tariff.Tariffs.fast);

myCompany.OrderTransportation("Artem", "1234", 51, Tariff.Tariffs.fast);

myCompany.GetTotalSumOrders("Artyom", "12345");

myCompany.GetTotalSumOrders("Arteeeeeeeem", "1234");

}

}

}

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

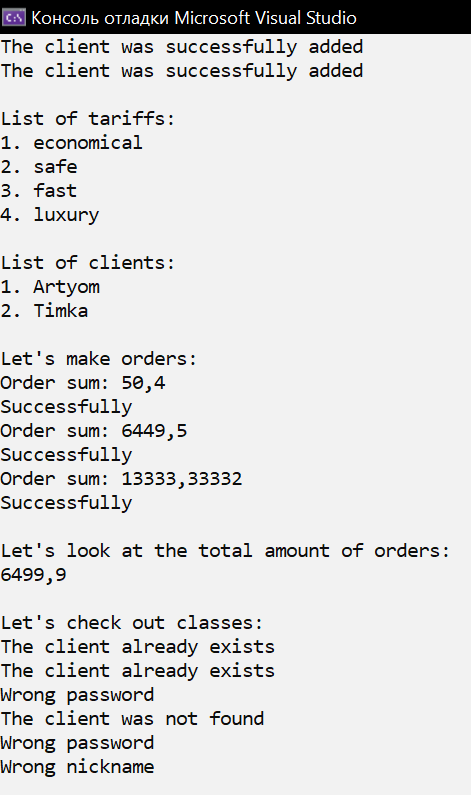


Рисунок 3 – Результат работы программы

# Вывод

В ходе лабораторной работы были изучены принципы построения диаграмм классов, использования типа данных enum и получены навыки проектирования приложения, состоящих из нескольких взаимосвязанных классов.