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 №4

на тему:

**«Конструкторы. Статические члены класса.**

**Шаблон проектирования Singleton.»**

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 - Тариф. В классе хранить информацию об оплате за перевозку одной тонны грузов (не зависит от направления - класс Тариф), масса перевезенных грузов, наименование фирмы. Реализовать метод для подсчета общей выручки фирмы. Реализовать возможность изменения (увеличения и уменьшения) тарифа.

# 2 Выполнение работы

Для выполнения задания в проект была добавлена папка Classes, в которой находятся классы TrackCompany и Tariff (cм. рисунок 1).

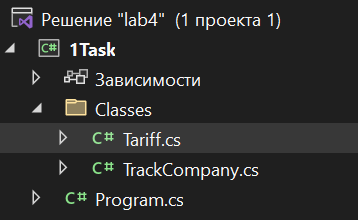


Рисунок 1 – Обозреватель решений

Сперва был реализован класс Tariff, в котором имеется свойство costTone, что отвечает за стоимость перевозки за одну тонну груза. Для costTone были реализованы методы UppCost и DownCost, которые позволяют поднимать и снижать стоимость тарифа.

namespace \_1Task.Classes

{

public class Tariff

{

public double costTone { get; private set; }

public Tariff(double costValue) =>

costTone = costValue;

public void UppCost(double value) =>

costTone += value;

public void DownCost(double value) =>

costTone = (costTone > value) ? costTone - value : 0;

}

}

Рассмотрим реализацию класса TrackCompany, в котором мы применили шаблон проектирования Singleton, который гарантирует, что для класс будет создан только один объект. Помимо этого в классе реализован метод profitCompany, который вычислят прибыль компании по установленному тарифу.

namespace \_1Task.Classes

{

public class TrackCompany

{

private static TrackCompany? instance;

public Tariff tariff { get; private set; }

public string nameCompany { get; private set; }

public double toneWeight { get; private set; }

public TrackCompany(string nameCompany, double toneWeight, double costTariffValue)

{

this.nameCompany = nameCompany;

this.toneWeight = toneWeight;

tariff = new Tariff(costTariffValue);

}

public TrackCompany(string nameCompany)

{

this.nameCompany = nameCompany;

toneWeight = 0;

tariff = new Tariff(0);

}

public static TrackCompany GetInstance(string nameCompany, double toneWeight,

double costTariffValue) =>

instance ??= new TrackCompany(nameCompany, toneWeight, costTariffValue);

public static TrackCompany GetInstance(string nameCompany) =>

instance ??= new TrackCompany(nameCompany);

public void UppCost(double value) =>

tariff.UppCost(value);

public void DownCost(double value) =>

tariff.DownCost(value);

public double profitCompany() =>

toneWeight \* tariff.costTone;

}

}

Покажем принцип работы этой программы в классе Program, где, помимо всего, реализован метод CorrectInput в качестве проверки на ввод для типа данных double. Используя шаблон проектирования Singleton мы создаём объект нашего класса TrackCompany и реализуем принцип работы программы используя оператор выбора switch.

namespace \_1Task

{

internal class Program

{

static void CorrectInput(out double value)

{

string? str = null;

while (true)

{

str = Console.ReadLine();

if (double.TryParse(str, out value))

{

value = double.Parse(str);

break;

}

Console.WriteLine("re-Input value: ");

}

}

static void Main(string[] args)

{

string? nameCompany = null, str = null;

double costTariff = 0, toneWeight = 0;

Console.WriteLine("Input name your company:");

nameCompany = Console.ReadLine();

Console.WriteLine("Input cost tariff:");

CorrectInput(out costTariff);

Console.WriteLine("Input the weight of the transported:");

CorrectInput(out toneWeight);

TrackCompany myCompany = TrackCompany.GetInstance(nameCompany, toneWeight, costTariff);

while (true)

{

Console.WriteLine("Select method or other number to exit:\n" +

"1 - Get name my company\n2 - Get cost tariff\n" +

"3 - Get weight of the transported\n4 - Get profit my company\n" +

"5 - Upp tariff\n6 - Down tariff");

str = Console.ReadLine();

if (str != "1" && str != "2" && str != "3" && str != "4" &&

str != "5" && str != "6") Environment.Exit(0);

switch (int.Parse(str))

{

case 1:

{

Console.WriteLine("Name company: " + myCompany.nameCompany + "\n");

break;

}

case 2:

{

Console.WriteLine("The cost of tariff: " +

myCompany.tariff.costTone + "\n");

break;

}

case 3:

{

Console.WriteLine("The weight of the transported: " +

myCompany.toneWeight + "\n");

break;

}

case 4:

{

Console.WriteLine("Profit company: " +

myCompany.profitCompany() + "\n");

break;

}

case 5:

{ double value = 0;

Console.WriteLine("Input value:");

CorrectInput(out value);

myCompany.UppCost(value);

break;

}

case 6:

{

double value = 0;

Console.WriteLine("Input value:");

CorrectInput(out value);

myCompany.DownCost(value);

break;

}

}

}

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 2).

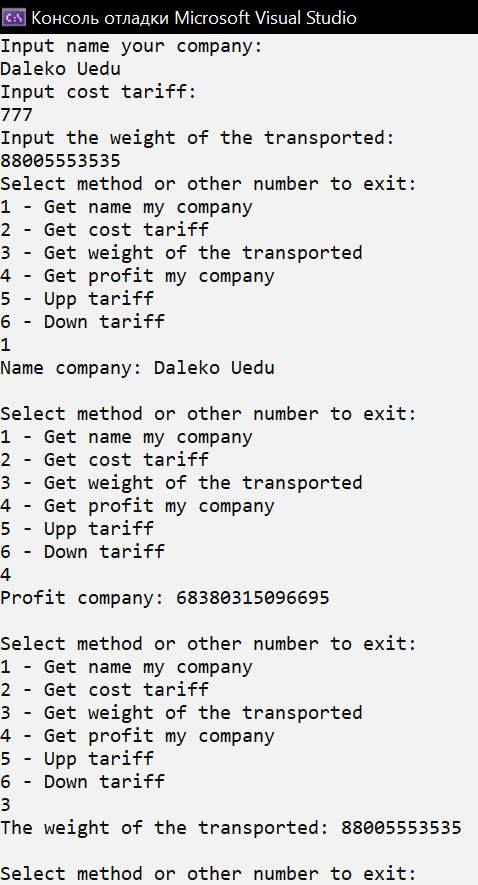


Рисунок 2 – Результат работы программы

# Вывод

В ходе лабораторной работы были изучены назначения конструкторов в классе, перегрузка методов класса, возможности членов класса с модификатором static, шаблоны проектирования Singleton.