

Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерного проектирования
Кафедра Информатики
Дисциплина «Программирование»

ОТЧЕТ

к лабораторной работе №4

на тему:

**«КОНСТРУКТОРЫ. СТАТИЧЕСКИЕ ЧЛЕНЫ КЛАССА. ШАБЛОН
ПРОЕКТИРОВАНИЯ SINGLETON.»**

БГУИР 6-05-0612-02 17

Выполнил студент группы 353502
ХАРИТОНЧИК Денис Сергеевич

(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

В рамках проекта была создана директория SingletonClasses, в которой размещаются два файла с классами TransportCompany и Tariff. Эти классы представляют важные компоненты системы управления транспортной компанией.

Класс TransportCompany является важным компонентом системы управления транспортной компанией. Он обладает рядом функциональных возможностей, которые обеспечивают эффективное управление данными и операциями компании. Одной из ключевых особенностей класса является его способность хранить информацию о транспортной компании, такую как её название и тарифы на услуги перевозки. Это позволяет легко получать доступ к этой информации из различных частей системы, что упрощает её использование и обеспечивает её единообразие. Кроме того, класс предоставляет методы для расчета общего дохода компании на основе массы перевозимого груза и установленных тарифов. Это позволяет компании эффективно управлять своими финансами и анализировать её прибыльность. Это позволяет компании быстро реагировать на изменяющиеся рыночные условия и оптимизировать свою прибыльность. Реализация шаблона Singleton обеспечивает наличие единственного экземпляра класса в системе. Это гарантирует консистентность данных и предотвращает нежелательные дублирования объектов, что повышает надежность и эффективность работы системы. Ниже приведен листинг кода класса TransportCompany.

```

public class TransportCompany
{
    private string companyName;
    private Tariff tariff;
    // приватное статическое поле для хранения единственного экземпляра класса
TransportCompany
    private static TransportCompany _instance;
    // приватный конструктор для создания экземпляра класса TransportCompany
    private TransportCompany(string companyName, Tariff tariff)
    {
        this.companyName = companyName;
        this.tariff = tariff;
    }
    // публичный статический метод для получения единственного экземпляра
класса TransportCompany
    public static TransportCompany GetInstance(string companyName, Tariff
tariff)
    {
        if (_instance == null)
        {
            _instance = new TransportCompany(companyName, tariff);
        }
        return _instance;
    }
    // метод для расчета общего дохода компании для заданных масс груза
    public double ComputeTotalEarnings(double[] cargo)
    {
        double totalEarning = 0;
        foreach (double weight in cargo)
        {
            totalEarning += tariff.CalculateEarnings(weight);
        }
        return totalEarning;
    }
    // метод для изменения тарифа компании
    public void ChangeCompanyTariff(double newPrice)
    {
        tariff.ChangeTariff(newPrice);
    }
}

```

Класс Tariff представляет тарифные ставки для перевозки грузов в транспортной компании. Он обеспечивает хранение информации о тарифах и расчет дохода от перевозки грузов. Основные возможности класса включают хранение тарифов на перевозку грузов, реализацию шаблона Singleton для обеспечения единственного экземпляра класса в системе, а также методы для расчета дохода от перевозки грузов и изменения тарифов. Использование класса Tariff в системе обеспечивает эффективное управление тарифами на

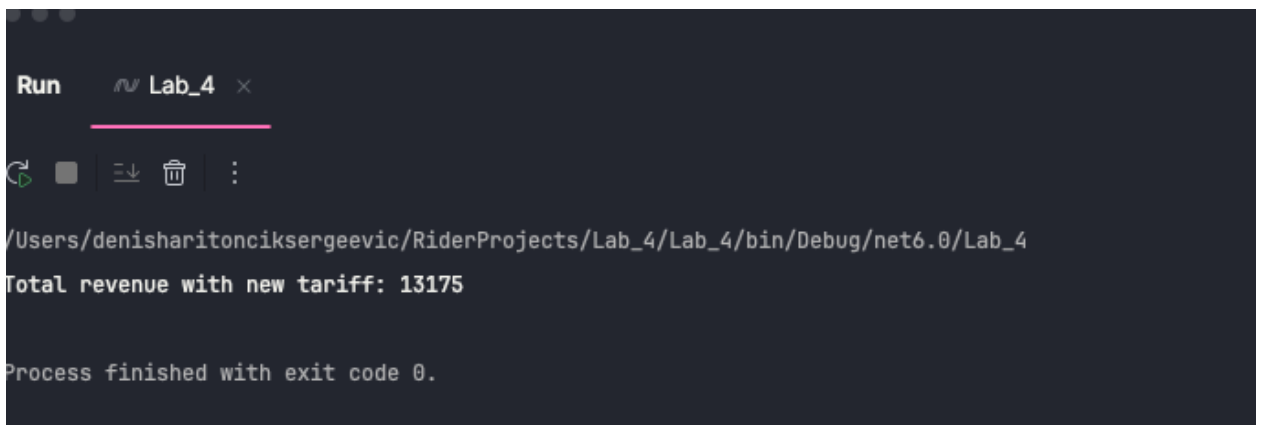
перевозку грузов, что позволяет компании адаптироваться к изменяющимся рыночным условиям и оптимизировать свою прибыльность. Ниже приведен листинг кода класса Tariff.

```
public class Tariff
{
    private double priceOneTon;
    // приватный статический экземпляр класса Tariff
    private static Tariff _instance;
    // приватный конструктор для создания экземпляра класса Tariff
    private Tariff(double priceOneTon)
    {
        this.priceOneTon = priceOneTon;
    }
    // публичный статический метод для получения единственного экземпляра
    класса Tariff
    public static Tariff GetInstance(double priceOneTon)
    {
        if (_instance == null)
        {
            _instance = new Tariff(priceOneTon);
        }
        return _instance;
    }
    // метод для расчета дохода для заданного веса груза
    public double CalculateEarnings(double cargo)
    {
        return cargo * priceOneTon;
    }
    // метод для изменения тарифа
    public void ChangeTariff(double newPrice)
    {
        this.priceOneTon = newPrice;
    }
}
```

Класс Program представляет точку входа в приложение. Внутри метода Main создаются экземпляры классов Tariff и TransportCompany, используя методы GetInstance, которые обеспечивают единственный экземпляр каждого класса. Далее создается массив cargo, содержащий данные о массе груза. Затем вызывается метод ComputeTotalEarnings экземпляра класса TransportCompany для расчета общего дохода компании от перевозки заданных грузов. Далее изменяется тариф компании с помощью вызова метода ChangeCompanyTariff экземпляра класса TransportCompany. После этого повторно вызывается метод ComputeTotalEarnings для расчета общего дохода компании с новым тарифом. Наконец, результат выводится на консоль

с помощью метода `WriteLine`. Использование класса `Program` обеспечивает запуск и управление работой приложения, включая создание необходимых объектов и выполнение операций с ними. Ниже приведен листинг кода класса `Program`. На рисунке 1 изображена работа консольного приложения.

```
class Program
{
    static void Main(string[] args)
    {
        // создание экземпляра класса Tariff (единственный экземпляр)
        Tariff tariff = Tariff.GetInstance(100);
        // создание экземпляра класса TransportCompany (единственный экземпляр)
        TransportCompany company = TransportCompany.GetInstance("My Company",
tariff);
        double[] cargo = { 10, 35, 40 };
        double totalEarning = company.ComputeTotalEarnings(cargo);
        company.ChangeCompanyTariff(155);
        totalEarning = company.ComputeTotalEarnings(cargo);
        Console.WriteLine($"Total revenue with new tariff: {totalEarning}");
    }
}
```



```
Run Lab_4 x
/Users/denisharitonciksergeevic/RiderProjects/Lab_4/Lab_4/bin/Debug/net6.0/Lab_4
Total revenue with new tariff: 13175
Process finished with exit code 0.
```

Рисунок 1 – Процесс работы консольного приложения

ВЫВОД

В ходе выполнения лабораторной работы были изучены основные принципы создания и использования классов в языке программирования C#. Особое внимание было уделено разработке классов для выбранной предметной области, включая класс-контейнер, управляющий контейнеризируемым классом, и сам контейнеризируемый класс. Целью работы было ознакомление с концепциями конструкторов, статических членов класса и шаблона проектирования Singleton. В ходе выполнения задания были изучены и применены эти концепции для разработки структурированных и эффективных классов. Результатом выполнения лабораторной работы является создание нескольких классов с соответствующей структурой и функциональностью. Каждый класс обладает набором полей, методов и свойств, а также может содержать статические методы и применять шаблон проектирования Singleton для обеспечения единственного экземпляра класса в системе.