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 №8
на тему:
«ПОЛИМОРФИЗМ»
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 4. Предметная область: Аэропорт. Касса аэропорта имеет список тарифов на различные направления. Тариф содержит название направления и стоимость перевозки. На некоторые направления предоставляется фиксированная скидка. В классе аэропорт реализовать метод добавления нового тарифа и метод поиска направления с максимальной стоимостью.

2 ВЫПОЛНЕНИЕ РАБОТЫ

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

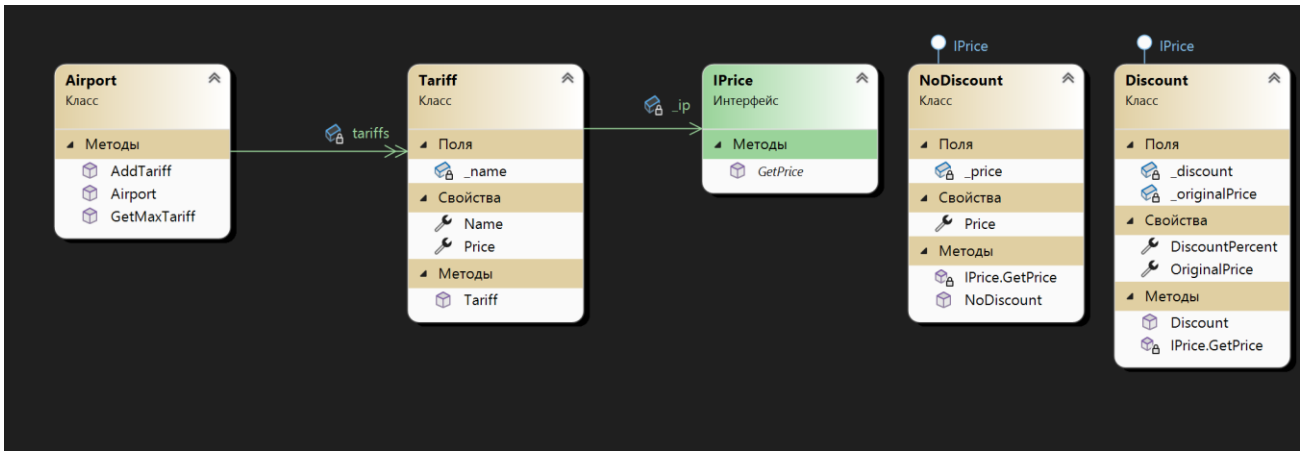

Рисунок 1 – Диаграмма классов

Для выполнения задания был создан интерфейс IPrice, где определён метод GetPrice, который должен будет возвращать общую стоимость со скидкой или без неё.

```
internal interface IPrice
{
    decimal GetPrice();
}
```

Сперва рассмотрим один из классов, который реализует интерфейс IPrice. Discount содержит метод GetPrice, который считает общую стоимость для тарифа со скидкой.

```
namespace Polymorphism
{
    internal class Discount : IPrice
    {
        private decimal _originalPrice;
```

```
                private decimal _discount;
                public decimal OriginalPrice
                {
                    get { return _originalPrice; }
                    set
                    {
                        if (value < 0)
                            throw new ArgumentException("Цена не может быть
отрицательной");
                        _originalPrice = value;
                    }
                }

                public decimal DiscountPercent
                {
                    get { return _discount; }
                    set
                    {
                        if (value < 0 || value > 100)
                            throw new ArgumentException("Скидка должна быть в
пределах 0 - 100 ");
                        _discount = value;
                    }
                }

                public Discount(decimal price, decimal discount)
                {
                    OriginalPrice = price;
                    DiscountPercent = discount;
                }

                decimal IPrice.GetPrice()
                {
                    return OriginalPrice * (100 - DiscountPercent) / 100;
                }
            }
        }
```

Класс NoDiscount также реализует интерфейс IPrice и содержит метод GetPrice, который считает общую стоимость тарифа, но уже без скидки.

```
        namespace Polymorphism
        {
            internal class NoDiscount : IPrice
            {
                private decimal _price;

                public decimal Price
                {
                    get
                    {
                        return _price;
                    }
                    set
                    {
                        if (value < 0)
                            throw new ArgumentException("Цена не может быть
отрицательной");
                        _price = value;
                    }
                }
                public NoDiscount(decimal price)
```

```
            {
                Price = price;
            }

            decimal IPrice.GetPrice()
            {
                return _price;
            }
        }
    }
```

Взглянем на реализацию класса Tariff, у которого есть поля Name и Price, которые отвечают за хранение направление полета и его стоимости соответственно.

```
        internal class Tariff
        {
            private IPrice _ip;
            string _name;
            public string Name {
                get
                {
                    return _name;
                }
                set
                {
                    if (string.IsNullOrWhiteSpace(value))
                        throw new ArgumentException("Название не может быть
пустым");
                    _name = value;
                }
            }
            public decimal Price {
                get
                {
                    return _ip.GetPrice();
                }
            }
            public Tariff(string name, decimal price, decimal discount = 0)
            {
                if (price < 0)
                    throw new ArgumentException("Цена не может быть
отрицательной");
                if (discount == 0)
                {
                    _ip = new NoDiscount(price);
                }
                else
                {
                    _ip = new Discount(price, discount);
                }

                Name = name;
            }
        }
    }
```

В классе Airport создается список всех тарифов, а также методы для добавления тарифа и вычисления направления самого дорогого полета.

```
namespace Polymorphism
{
    internal class Airport
    {
        private List<Tariff> tariffs;

        public Airport()
        {
            tariffs = new List<Tariff>();
        }
        public void AddTariff(Tariff tariff)
        {
            tariffs.Add(tariff);
        }

        public Tariff GetMaxTariff()
        {
            if (tariffs.Count == 0)
                throw new InvalidOperationException("Нет добавленных
тарифов");
            Tariff maxTariff = tariffs[0];
            for (int i = 1; i < tariffs.Count; i++)
            {
                if (maxTariff.Price < tariffs[i].Price)
                {
                    maxTariff = tariffs[i];
                }
            }
            return maxTariff;
        }

    }
}
```

В классе Program создаётся объект класса Airport и демонстрируется корректность выполнения работы.

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            // 1. Демонстрация работы через интерфейсные ссылки
            Console.WriteLine("=== Демонстрация ===");

            IPrice noDiscountStrategy = new NoDiscount(5000);
            IPrice discountStrategy = new Discount(4000, 10);

            Console.WriteLine($"Цена без скидки:
{noDiscountStrategy.GetPrice()} руб.");
            Console.WriteLine($"Цена со скидкой:
{discountStrategy.GetPrice()} руб.");

            // 2. Демонстрация паттерна Strategy в работе аэропорта
            Console.WriteLine("\n=== Работа аэропорта ===");

            Airport airport = new Airport();

            airport.AddTariff(new Tariff("Москва", 5000));
            airport.AddTariff(new Tariff("Санкт-Петербург", 4000, 10));
            airport.AddTariff(new Tariff("Сочи", 3000));
```

```
                    airport.AddTariff(new Tariff("Калининград", 3500, 15));

                    Tariff maxTariff = airport.GetMaxTariff();
                    Console.WriteLine($"Самый дорогой тариф: {maxTariff.Name} -
{maxTariff.Price} руб.");

                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Произошла ошибка: {ex.Message}");
                }
            }
        }
```

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

```
=== Демонстрация ===
Цена без скидки: 5000 руб.
Цена со скидкой: 3600 руб.

=== Работа аэропорта ===
Самый дорогой тариф: Москва – 5000 руб.
```

Рисунок 2 – Результат работы программы

ВЫВОД

В ходе лабораторной работы были изучены базовые принципы работы с интерфейсами и освоен принцип работы с шаблоном проектирования Strategy.