**Лабораторная работа №8. Интерфейсы.**

Задание 1. Интерфейсы Ix, Iy, Iz, содержат объявления методов с одной и той же сигнатурой следующим образом

interface Ix

{

void IxF0(параметр);

void IxF1();

}

interface Iy

{

void F0(параметр);

void F1();

}

interface Iz

{

void F0(параметр);

void F1();

}

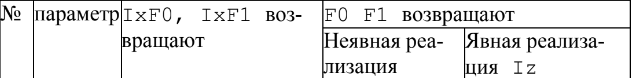
Эти интерфейсы наследуются в классе TestClass, содержащий член w типа параметр и реализуются так, как задано в варианте. В каждом методе задать вывод результата.

Рассмотреть случай

* неявной реализации интерфейсов
* явной реализации интерфейса Iz

В программе должна выполняться:

* неявная неоднозначная реализация методов интерфейсов Iy и Iz,
* вызов функций с явным приведением к типу интерфейса,
* вызов метода для объекта посредством интерфейсной ссылки.





using System;

using System.Collections.Generic;

interface Ix

{

void IxF0(double param);

void IxF1();

}

interface Iy

{

void F0(double param);

void F1();

}

interface Iz

{

void F0(double param);

void F1();

}

class TestClass : Ix, Iy, Iz

{

private double w;

// Неявная реализация методов интерфейсов Ix, Iy и Iz

public void IxF0(double param)

{

Console.WriteLine($"Log(w) = {Math.Log(w)}");

}

public void IxF1()

{

Console.WriteLine($"2/w = {2 / w}");

}

void Iy.F0(double param)

{

Console.WriteLine($"2/w = {2 / w} (неоднозначная реализация)");

}

void Iy.F1()

{

Console.WriteLine("Вызван метод F1() интерфейса Iy");

}

// Явная реализация интерфейса Iz

void Iz.F0(double param)

{

Console.WriteLine($"w^3 = {Math.Pow(w, 3)}");

}

void Iz.F1()

{

Console.WriteLine("Вызван метод F1() интерфейса Iz");

}

public TestClass(double w)

{

this.w = w;

}

}

class Program

{

static void Main(string[] args)

{

double w = 5.0; // Произвольное значение параметра w

TestClass testObject = new TestClass(w);

// Вызов методов через интерфейсные ссылки

Ix ix = testObject;

Iy iy = testObject;

Iz iz = testObject;

ix.IxF0(w);

ix.IxF1();

iy.F0(w);

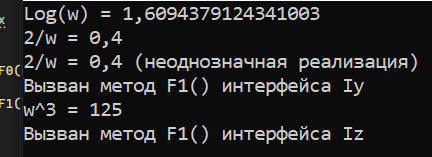
iy.F1();

iz.F0(w);

iz.F1();

}

}



**Задание 2.** Выполнить задания, используя для хранения экземпляров разработанных классов стандартные параметризованные коллекции. Во всех классах реализовать интерфейсы IComparable и IComparer перегрузить операции отношения для реализации сравнения объектов по указанному полю. Результат вывести на экран.

Составить список больных отделения, включив следующие данные: ФИО, болезнь, дата поступления, рабочий стаж. Вывести в новый список информацию о больных, находящихся на лечении больше недели, отсортировав их по ФИО.

using System;

using System.Collections.Generic;

// Класс, представляющий информацию о больном

class Patient : IComparable<Patient>

{

public string FullName { get; set; }

public string Disease { get; set; }

public DateTime AdmissionDate { get; set; }

public int WorkExperience { get; set; }

// Реализация интерфейса IComparable для сравнения по ФИО

public int CompareTo(Patient other)

{

return string.Compare(FullName, other.FullName, StringComparison.OrdinalIgnoreCase);

}

}

class Program

{

static void Main()

{

// Создаем список больных

var patients = new List<Patient>

{

new Patient { FullName = "Иванов Иван Иванович", Disease = "Грипп", AdmissionDate = DateTime.Parse("2024-02-10"), WorkExperience = 5 },

new Patient { FullName = "Петрова Ольга Сергеевна", Disease = "Ангина", AdmissionDate = DateTime.Parse("2024-02-15"), WorkExperience = 8 },

new Patient { FullName = "Сидоров Павел Николаевич", Disease = "Простуда", AdmissionDate = DateTime.Parse("2024-02-12"), WorkExperience = 3 },

// Добавьте еще больных по аналогии

};

// Фильтруем больных, находящихся на лечении больше недели

var hospitalizedMoreThanWeek = patients.FindAll(p => (DateTime.Now - p.AdmissionDate).TotalDays > 7);

// Сортируем по ФИО

hospitalizedMoreThanWeek.Sort();

// Выводим информацию о больных

Console.WriteLine("Список больных, находящихся на лечении больше недели (по ФИО):");

foreach (var patient in hospitalizedMoreThanWeek)

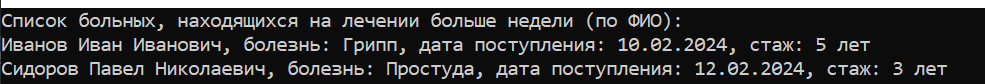
{

Console.WriteLine($"{patient.FullName}, болезнь: {patient.Disease}, дата поступления: {patient.AdmissionDate.ToShortDateString()}, стаж: {patient.WorkExperience} лет");

}

}

}



Контрольные вопросы:

1. **Как описывается интерфейс? Его назначение.**

Интерфейс в C# содержит определения для группы связанных функциональных возможностей, которые должны реализовать неабстрактный класс или структура. Он позволяет определить контракт, который классы и структуры обязаны соблюдать.

1. **Какие члены может содержать интерфейс?**

Назначение интерфейса: Интерфейс позволяет включить в класс поведение из нескольких источников. Это важно, так как C# не поддерживает множественное наследование классов. Также интерфейсы используются для имитации наследования для структур, которые не могут фактически наследовать от другой структуры или класса.

1. **Какие члены может содержать интерфейс?**

Интерфейсы могут содержать методы экземпляра, свойства, события, индексаторы, а также любое сочетание этих четырех типов членов. Они также могут содержать статические конструкторы, поля, константы или операторы1.

1. **Какие спецификаторы допустимы у методов, реализующих интерфейс?**

Методы, реализующие интерфейс, могут иметь любые допустимые спецификаторы доступа, такие как public, protected, internal, private, protected internal или private protected1.

1. **В каких случаях используется явная реализация интерфейса?**

Явная реализация интерфейса используется, когда необходимо разрешить конфликт имен методов, которые реализуют один и тот же интерфейс1.

1. **Как осуществляется наследование интерфейсов?**

Класс или структура может реализовывать несколько интерфейсов, но наследовать только от одного класса.

1. **Можно ли явно реализованные методы объявлять виртуальными?**

Нет, явно реализованные методы нельзя объявлять виртуальными.

1. **Можно ли повторно реализовать интерфейс, указав его имя в списке предков класса наряду с классом-предком?**

Да, можно.

1. **Какие стандартные интерфейсы используются для работы с коллекциями?**

Для работы с коллекциями используются интерфейсы IEnumerable, ICollection, IList, IDictionary и другие.

1. **Чем отличаются интерфейсы IComparable и IComparer?**

IComparable используется для сравнения объектов одного типа, а IComparer позволяет определить пользовательское сравнение для разных типов1.