

Лабораторна робота № 5 ІНТЕРФЕЙСИ

В C#.

Мета: Набути умінь і навичок створення та реалізації інтерфейсів C# у середовищі Microsoft Visual Studio 2022.

Короткі теоретичні відомості

Інтерфейси - це ще один інструмент реалізації поліморфізму в C#. Інтерфейс являє собою набір методів (властивостей, подій, індексаторів), реалізацію яких повинен забезпечити клас, який реалізує інтерфейс.

Інтерфейс може містити тільки сигнатури (ім'я та типи параметрів) своїх членів. Інтерфейс не може містити конструктори, поля, константи, статичні члени. Створювати об'єкти інтерфейсу неможливо.

Оголошення інтерфейсу

Інтерфейс - це одиниця рівня класу, він оголошується за межами класу, за допомогою ключового слова `interface`:

```
interface ISomeInterface
{
    // Тіло інтерфейсу
}
```

* Імена інтерфейсів прийнято давати, починаючи з префіксу «I», щоб відразу відрізнити де клас, а де інтерфейс.

Усередині інтерфейсу оголошуються сигнатури його членів, модифікатори доступу вказувати не потрібно:

```
interface ISomeInterface
{
    string SomeProperty { get; set; } // Властивість

    void SomeMethod(int a); // Метод }
```

Реалізація інтерфейсу

Щоб вказати, що клас реалізовує інтерфейс, необхідно, так само, як і при спадкуванні, після імені класу і двокрапки вказати ім'я інтерфейсу:

```
class SomeClass : ISomeInterface // реалізація інтерфейсу ISomeInterface
{
    // Тіло класу
}
```

Клас, який реалізує інтерфейс, повинен надати реалізацію всіх членів інтерфейсу:

```
class SomeClass : ISomeInterface
{
    public string SomeProperty
    {
        get
        {
            // Тіло get аксесор
        }
        set
        {
            // Тіло set аксесор
        }
    }
    public void SomeMethod(int a)
    {
        // Тіло методу
    }
}
```

Приклад. Є класи геометричних фігур Прямокутник і Окружність. У обох класів повинні бути методи обчислення периметра і площі. Ці методи ми представимо інтерфейсом:

```
interface IGeometrical // оголошення інтерфейсу
```

```

{    void
GetPerimeter();    void
GetArea();
}

class Rectangle : IGeometrical // реалізація інтерфейсу
{    public double Width { get; set;
}    public double Height { get; set;
}    public void GetPerimeter()
{
    Console.WriteLine("(Width + Height) * 2 = " + (Width + Height) * 2);
}
public void GetArea()
{
    Console.WriteLine("Width * Height = " + Width * Height);
}
}

class Circle : IGeometrical // реалізація інтерфейсу
{    public double Radius { get; set;
}    public void GetPerimeter()
{
    Console.WriteLine("2 * pi * Radius = " + 2 * Math.PI * Radius);
}

    public void GetArea()
    {
        Console.WriteLine("pi * Radius^2 = " + Radius * Radius * Math.PI);
    }
} class

Program
{    static void Main(string[]
args)
{
    List<IGeometrical> figures = new List<IGeometrical>();
figures.Add(new Rectangle() { Width = 2, Height = 4 });

```

```

    figures.Add(new Circle() { Radius = 3 });
    foreach (IGeometrical f in figures)
    {
        f.GetPerimeter();
        f.GetArea();
    }
    Console.ReadLine();
}

```

У підсумку, ми можемо об'єднати в один список об'єкти класів, які реалізують один і той же інтерфейс.

Інтерфейси дуже схоже на абстрактні класи і є заміною множинному успадкуванню, яке є в мові C++, в C# від нього відмовилися і внесли інтерфейси. У C# клас може реалізовувати відразу кілька інтерфейсів. Це і є головною відмінністю використання інтерфейсів і абстрактних класів. Крім того, абстрактні класи можуть містити всі інші члени, яких не може бути в інтерфейсі, і не всі методи і властивості в абстрактному класі повинні бути абстрактними.

Якщо клас реалізовує декілька інтерфейсів, вони розділяються комами:

```

interface IDrawable
{
    void
    Draw();
} interface
IGeometrical
{
    void
    GetPerimeter(); void
    GetArea();
} class Rectangle : IGeometrical,
IDrawable
{
    public void
    GetPerimeter()

```

```

{
    Console.WriteLine("(a + b) * 2");
}

public void GetArea()
{
    Console.WriteLine("a * b");
}

public void Draw()
{
    Console.WriteLine("Rectangle");
}
} class Circle : IGeometrical,
IDrawable
{
    public void GetPerimeter()
    {
        Console.WriteLine("2 * pi * r");
    }

    public void GetArea()
    {
        Console.WriteLine("pi * r ^ 2");
    }

    public void Draw()
    {
        Console.WriteLine("Circle");
    }
}

```

Тут був оголошений інтерфейс IDrawable, який надає метод для малювання об'єкта. Цей інтерфейс може реалізовувати, наприклад, клас Image. Класи Image і Circle зовсім різні сутності, і вони не мають спільного базового класу, але ми можемо створити список покажчиків на інтерфейс IDrawable, і працювати з такими об'єктами, як з однотипними (з однаковим інтерфейсом).

Цей приклад з IDrawable більш наочно відображає можливості, які надають інтерфейси.

Реалізація інтерфейсів по замовчанню

Починаючи з версії C# 8.0, інтерфейси підтримують реалізацію методів і властивостей за замовчуванням. Навіщо це потрібно? Припустимо, ми маємо купу класів, які реалізують деякий інтерфейс. Якщо ми додамо у цей інтерфейс новий метод, ми будемо змушені реалізувати цей метод у всіх класах, що використовують цей інтерфейс. Інакше подібні класи просто не компілюватимуться. Тепер замість реалізації методу у всіх класах нам достатньо визначити його реалізацію за умовчанням в інтерфейсі. Якщо клас не реалізує метод, застосовуватиметься реалізація за умовчанням.

```
IMovable tom = new Person();
Car tesla = new Car();
tom.Move(); // Walking
tesla.Move(); // Driving interface

IMovable
{
    void Move() =>
    Console.WriteLine("Walking");
}

class Person : IMovable { }
class Car : IMovable
{
    public void Move() =>
    Console.WriteLine("Driving");
}
```

У цьому випадку інтерфейс IMovable визначає стандартну реалізацію для методу Move. Клас Person не реалізує цей метод, тому він застосовує стандартну реалізацію на відміну від класу Car, який визначає свою реалізацію для методу Move.

Варто зазначити, що хоча для об'єкта класу Person ми можемо викликати метод Move - адже клас Person застосовує інтерфейс IMovable, проте ми не можемо написати так:

```
Person tom = new Person();
```

```
tom.Move();    // Помилка - метод Move не визначений в класі Person
```

Завдання до лабораторної роботи

Реалізувати інтерфейси і наслідування для класів згідно індивідуального завдання:

1. interface Видання(Шифр, Автор, Назва, Рік, видавництво) → abstract class Книга → class Довідник, class Енциклопедія.
2. interface Abiturient → abstract class Student → class Student Of Faculty.
3. interface Робітник → class Інженер → class Керівник.
4. interface Учбовий Заклад → class Коледж, class Університет.
5. interface Споруда → abstract class Громадська Споруда → class Театр.
6. interface Mobile → abstract class Mobile → class Siemens Model.
7. interface Корабель → abstract class Військовий Корабель → class Авіаносець.
8. interface Лікар → class Хірург → class Нейрохірург.
9. interface Корабель → abstract class Грузовий Корабель → class Танкер.
10. interface Диск → abstract class Директорія → class Файл.
11. interface Робітник → class Інженер → class Керівник.
12. interface Учбовий Заклад → class Коледж, class Університет.
13. interface Споруда → abstract class Громадська Споруда → class Театр.
14. interface Корабель → abstract class Грузовий Корабель → class Танкер.
15. interface Лікар → class Хірург → class Нейрохірург.
16. interface Mobile → abstract class Mobile → class Siemens Model.

17. interface Корабель → abstract class Військовий Корабель → class
Авіаносець.

18. interface Лікар → class Хірург → class Нейрохірург.

19. interface Корабель → abstract class Грузовий Корабель → class
Танкер.

20. interface Диск → abstract class Директорія → class Файл.

Код програми(варіант 10):

```
using System;
namespace Lab5
{
    interface IDiskSystem
    {
        string Path { get; set; }
        void Delete();

        void CopyTo(string path);

        void Cut();

        void Paste(string path);
    }

    class Directory: IDiskSystem
    {
        public string Path { get; set; }

        public void Up(string path)
        {
            Path = Path + "/" + path;
        }

        public void Down()
        {
            Path = Path.Substring(0, Path.IndexOf("/"));
        }

        public void Delete()
        {
            Console.WriteLine("Видалили директорію: " + Path);
        }

        public void CopyTo(string path)
```



```

    {
        Console.WriteLine("Копіювали " + Path + " в " + path);
    }

    public void Cut()
    {
        Console.WriteLine("Вилучили " + Path);
    }

    public void Paste(string path)
    {
        Console.WriteLine("Вставили " + Path + " в " + path);
    }
}

class File : IDiskSystem
{
    public string Path { get; set; }

    public void Delete()
    {
        Console.WriteLine("Видалили файл: " + Path);
    }

    public void CopyTo(string path)
    {
        Console.WriteLine("Копіювали " + Path + " в " + path);
    }

    public void Cut()
    {
        Console.WriteLine("Вилучили " + Path);
    }

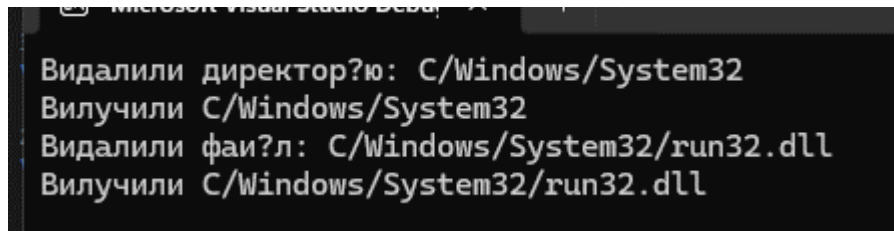
    public void Paste(string path)
    {
        Console.WriteLine("Вставили " + Path + " в " + path);
    }
}

class Program
{
    static void Main(string[] args)
    {
        IDiskSystem[] file_system = new IDiskSystem[] { new
Directory(), new File() };
        file_system[0].Path = "C/windows/System32";
        file_system[1].Path = "C/windows/System32/run32.dll";
        foreach (IDiskSystem elem in file_system)
        {
            elem.Delete();
            elem.Cut();
        }
    }
}

```

```
}  
}
```

Результат роботи програми:



```
Видалили директор?ю: C/Windows/System32  
Вилучили C/Windows/System32  
Видалили фай?л: C/Windows/System32/run32.dll  
Вилучили C/Windows/System32/run32.dll
```

Рисунок 5.1 – Результат роботи програми з реалізацією інтерфейсу

Контрольні питання.

1. Що являє собою інтерфейс?

Інтерфейс - це тип даних, який описує набір методів, властивостей та подій, які повинен реалізувати клас.

2. Як реалізується інтерфейс класом?

Клас використовує ключове слово `implements`, щоб оголосити, що він реалізує інтерфейс. Клас повинен надати реалізацію для всіх членів інтерфейсу.

3. Чому інтерфейс можна назвати заміною абстрактному класу?

Інтерфейси схожі на абстрактні класи, тому що вони не містять реалізації. Однак інтерфейси більш гнучкі, тому що клас може реалізувати кілька інтерфейсів, а не один абстрактний клас.

4. Які елементи інтерфейсу можна реалізувати за замовченням?

З C# 8.0 інтерфейси можуть мати реалізацію за замовчуванням для методів, властивостей та подій.

5. Як в класі реалізувати декілька інтерфейсів?

Клас використовує ключове слово `implements` і перераховує всі інтерфейси, які він реалізує, через кому. Або ж як при звичайному наслідуванні - через двокрапку.

6. В яких випадках потрібно використовувати перетворення типів для створення посилання на інтерфейс?

Перетворення типів потрібне, коли змінна зберігає посилання на базовий клас, а вам потрібно використовувати його як інтерфейс.