Bài 6.11: Interface

- ✓ Khái niệm và đặc điểm
- ✓ Cú pháp khai báo interface
- ✓ Triển khai interface
- ✓ Một số interface thường dùng
- ✓ Ví dụ minh họa và bài tập

Khái niệm và đặc điểm

- ✓Interface là một kiểu dữ liệu tham chiếu chứa các định nghĩa của một nhóm các chức năng mà các class non-abstract và struct kế thừa nó phải cung cấp các triển khai chi tiết.
- ✓Trước C# 8, interface giống như abstract class chỉ chứa các thành phần abstract.
- ✓Từ C# 8, một interface có thể chứa các thành phần có triển khai mặc định. Khi một class hay struct kế thừa interface, chúng không phải cung cấp các triển khai cụ thể cho các phương thức có triển khai mặc định trong interface.
- ✓Các triển khai mặc định trong interface nhằm cung cấp chức năng chung nhất cho mọi lớp kế thừa interface đó.
- ✓Ta không thể tạo đối tượng trực tiếp từ một interface. Nhưng biến interface có thể tham chiếu tới đối tượng của lớp kế thừa interface đó.
- ✓Một lớp chỉ được kế thừa trực tiếp từ một lớp khác nhưng có thể kế thừa(triển khai-implement) một hoặc nhiều interface.

Cú pháp khai báo interface

✓Cú pháp khai báo interface:

```
0 references
interface InterfaceName
{
    // interface members
}
```

✓Trong đó:

- ✓ Mọi interface đều bắt đầu với keyword interface. Interface có thể được khai báo trực tiếp trong namespace hoặc trong một lớp nào đó.
- ✓ Phần InterfaceName là tên của interface. Tên interface bắt đầu bằng chữ I(i viết hoa) liền sau đó là danh từ hoặc tính từ. Ví dụ: IComparable, IClonable, IFormatter, IDisposable…
- ✓ Phần interface members có thể chứa khai báo methods, properties, indexers, events. Từ C# 8 trở về sau, interface có thể chứa các hằng số, toán tử, static constructor, các kiểu lồng bên trong nó, các thành phần static fields, methods, properties, indexers, events; các phương thức có triển khai mặc định, cho phép chỉ định tường minh access modifier.
- ✓ Mặc định các thành phần khai báo bên trong interface không có phần thân và access modifier là public.

Ví dụ interface IShape và IPoint

```
0 references
interface IShape
{
    // khai báo thuộc tính abstract,
    // không phải auto-implement properties!
    0 references
    int X { get; set; }
    0 references
    int Y { get; set; }
    // khai báo phương thức abstract
    0 references
    double Area();
    0 references
    double Perimeter();
}
```

```
1 reference
interface IPoint
{
    // khai báo thuộc tính abstract
    0 references
    int X { get; set; }
    0 references
    int Y { get;  set; }
    // phương thức tính khoảng cách
    0 references
    double Distance(IPoint other);
}
```

Một số đặc điểm của interface

- ✓Từ C# 11, một interface có thể khai báo các thành phần là static abstract ngoại trừ các trường dữ liệu.
- ✓Interface không chứa trạng thái của đối tượng do đó nó không hỗ trợ auto properties, instance fields.
- ✓Một interface có thể kế thừa từ 1 hoặc nhiều interface khác.
- ✓Khi interface con ghi đè phương thức trong interface cha, nó phải sử dụng cú pháp triển khai interface tường minh.

Triển khai interface

- ✓Một lớp non-abstract hoặc struct khi kế thừa interface bắt buộc phải triển khai các thành phần chưa triển khai mặc định của interface.
- ✓Cú pháp kế thừa: **class X : BaseClass, Interface1, Interface2,… {…}**
- ✓Trong đó:
  - ✓ X là lớp con sẽ kế thừa lớp cha BaseClass và các interface thứ i.
  - ✓ BaseClass là tên lớp cha, Interface1, Interface2,… là tên các interface cha.
  - ✓ Phần thân lớp cho trong cặp {}.
- ✓Một lớp X có thể kế thừa duy nhất 1 lớp cha trực tiếp nhưng có thể đồng thời triển khai nhiều interface. Nếu trong danh sách lớp, interface cha có lớp cha thì nó phải được đưa lên đầu danh sách.
- ✓Nếu lớp X kế thừa Interface I1, I1 kế thừa I0 thì lớp X phải triển khai mọi phương thức, thuộc tính chưa được triển khai của họ hàng nhà interface I1.

Triển khai interface

- ✓Các thành phần bắt buộc triển khai của interface được triển khai cụ thể trong lớp con bằng cách định nghĩa chi tiết phần thân của thuộc tính, phương thức, bổ sung access modifier public và giữ nguyên thứ tự, kiểu tham số như trong interface.

```
1 reference
interface IShape
{
    // khai báo thuộc tính abstract,
    // không phải auto-implement properties!
    2 references
    int X { get; set; }
    2 references
    int Y { get; set; }
    // khai báo phương thức abstract
    1 reference
    double Area();
    1 reference
    double Perimeter();
}
```

```
class Circle : IShape
{
    private int _x;
    private int _y;
    4 references
    public double Radius { get; set; }
    2 references
    public int X { get => _x; set => _x = value; } // triển khai thuộc tính
    2 references
    public int Y { get => _y; set => _y = value; } // triển khai thuộc tính

    0 references
    public Circle(int x, int y, double radius)
    {
        X = x;
        Y = y;
        Radius = radius;
    }
    // triển khai các phương thức khai báo trong interface
    1 reference
    public double Area()
    {
        return Math.PI * Radius * Radius;
    }

    1 reference
    public double Perimeter()
    {
        return 2 * Math.PI * Radius;
    }
}
```

Ví dụ

```
0 references
class Lesson611
{
    0 references
    static void Main()
    {
        Circle circle = new Circle(3, 5, 12);
        IShape iShape = new Circle(7, 4, 30); // tham chiếu tới đối tượng Circle
        Console.WriteLine($"Chu vi duong tron r = {circle.Radius}: {circle.Perimeter()}");
        Console.WriteLine($"Dien tich duong tron r = {circle.Radius}: {circle.Area()}");
        Console.WriteLine($"Chu vi duong tron r = {((Circle)iShape).Radius}: {iShape.Perimeter()}");
        Console.WriteLine($"Dien tich duong tron r = {((Circle)iShape).Radius}: {iShape.Area()}");
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
Chu vi duong tron r = 12: 75.398223686155
Dien tich duong tron r = 12: 452.38934211693
Chu vi duong tron r = 30: 188.495559215388
Dien tich duong tron r = 30: 2827.43338823081
Press any key to continue . . .
```

Triển khai các interface trùng tên thành phần

- ✓Nếu hai phương thức trong 2 interface giống hệt nhau về dấu hiệu nhận biết và ta chỉ triển khai chúng bởi 1 phương thức public trong lớp con thì biến của cả hai interface này sẽ cùng gọi tới phương thức đó.

```
public interface IControl
{
    1 reference
    void Paint();
}

2 references
public interface ISurface
{
    1 reference
    void Paint();
}

2 references
public class SampleClass : IControl, ISurface
{
    2 references
    public void Paint()
    {
        // do something...
        Console.WriteLine("Paint ");
    }
}
```

```
0 references
class Lesson611
{
    0 references
    static void Main()
    {
        SampleClass obj = new SampleClass();
        IControl control = obj;
        ISurface surface = obj;
        obj.Paint(); // kết quả Paint
        control.Paint(); // kết quả: Paint
        surface.Paint(); // kết quả: Paint
    }
}
```

Triển khai tường minh interface

- ✓Nếu các interface cha của một lớp có cùng tên của thành phần(thuộc tính, phương thức) thì ta cần chỉ rõ đang triển khai cụ thể cho thành phần của interface nào.
- ✓Lúc này ta cần triển khai ít nhất 1 thành phần là tường minh, gồm cả tên interface và dấu chấm phân tách đặt trước tên thành phần.
- ✓Trong cú pháp triển khai interface tường minh **KHÔNG** có access modifier public.
- ✓Các triển khai interface tường minh sẽ **không thể truy cập bởi đối tượng của lớp triển khai các interface đó** mà chỉ có thể được truy cập từ các biến của interface mà nó kế thừa.
- ✓Tương tự, các phương thức, thuộc tính có triển khai mặc định trong interface cũng chỉ có thể được truy cập từ biến interface.

Ví dụ triển khai tường minh interface

```
public interface IControl
{
    2 references
    void Paint();
}

3 references
public interface ISurface
{
    2 references
    void Paint();
}

2 references
public class SampleClass : IControl, ISurface
{
    2 references
    void IControl.Paint()
    {
        // do something...
        Console.WriteLine("IControl.Paint");
    }

    2 references
    void ISurface.Paint()
    {
        // do something...
        Console.WriteLine("ISurface.Paint");
    }
}
```

```
0 references
class Lesson611
{
    0 references
    static void Main()
    {
        SampleClass obj = new SampleClass();
        IControl control = obj;
        ISurface surface = obj;
        control.Paint(); // kết quả: IControl.Paint
        surface.Paint(); // kết quả: ISurface.Paint
        // obj.Paint(); // error
    }
}
```

Một số interface thường dùng

- ✓**IDisposable**: dọn dẹp và giải phóng tài nguyên.
- ✓**IComparable**, **IComparer**: so sánh hai đối tượng dùng trong sắp xếp tập hợp.
- ✓**IEquatable**, **IEqualityComparer**: dùng để đánh giá tính tương đương của hai đối tượng.
- ✓**IEnumerable**: nhằm sử dụng với foreach và LINQ.
- ✓**IQueryable**: cho phép thực hiện truy vấn dữ liệu.
- ✓**IList**, **ICollection**: sử dụng với các collections có thể sửa đổi.
- ✓**IDictionary**: sử dụng để tìm kiếm trong collections.
- ✓**INotifyPropertyChange**: dùng để liên kết dữ liệu tới các lớp giao diện của WPF, winform, silverlight.

Ví dụ: triển khai interface Comparable<T>

```
class Student : IComparable<Student>
{
    2 references
    public string StudentId { get; set; }
    2 references
    public string FullName { get; set; }
    6 references
    public float Gpa { get; set; }

    6 references
    public Student(string id, string fullName, float gpa)...

    0 references
    public int CompareTo(Student other)
    {
        if(Gpa == other.Gpa)
        {
            return 0;
        }
        else if(Gpa > other.Gpa)
        {
            return 1;
        } else
        {
            return -1;
        }
    }

    0 references
    public override string ToString() => $"{StudentId}, {FullName}, {Gpa}";
}
```

Kết quả

```
class Lesson611
{
    0 references
    static void Main()
    {
        Student[] students = new Student[]
        {
            new Student("SV1001", "Hoang Thi Yen", 3.29f),
            new Student("SV1002", "Ngo Diec Phong", 3.34f),
            new Student("SV1003", "Mai Van Hai", 3.14f),
            new Student("SV1004", "Le Tran Dat", 3.47f),
            new Student("SV1005", "Nguyen Hoang Anh", 3.15f),
            new Student("SV1006", "Tran Duc Tuan", 3.55f),
        };
        Console.WriteLine("Truoc khi sap xep: ");
        ShowData(students);
        Console.WriteLine("==============================");
        Console.WriteLine("Sau khi sap xep: ");
        Array.Sort(students);
        ShowData(students);
    }

    2 references
    public static void ShowData(Student[] students)
    {
        foreach (var student in students)
        {
            Console.WriteLine(student);
        }
    }
}
```

```
C:\WINDOWS\system32\cmd.exe
Truoc khi sap xep:
SV1001, Hoang Thi Yen, 3.29
SV1002, Ngo Diec Phong, 3.34
SV1003, Mai Van Hai, 3.14
SV1004, Le Tran Dat, 3.47
SV1005, Nguyen Hoang Anh, 3.15
SV1006, Tran Duc Tuan, 3.55
==============================
Sau khi sap xep:
SV1003, Mai Van Hai, 3.14
SV1005, Nguyen Hoang Anh, 3.15
SV1001, Hoang Thi Yen, 3.29
SV1002, Ngo Diec Phong, 3.34
SV1004, Le Tran Dat, 3.47
SV1006, Tran Duc Tuan, 3.55
Press any key to continue . . .
```

Nội dung tiếp theo

So sánh abstract class vs interface