

Academy .NET

Interfacce

Sommario

- Interfacce - concetti di base
- Come implementare le interfacce in C #?
- Interfacce e polimorfismo
- Riferimenti alle interfacce: "is", "as"
- Interfacce come parametri
- Costruire tipi personalizzati
- IConvertible, IEnumerator, ICloneable, IComparable
- Interfaccia System.Collections
- ArrayList, Hashtable, Queue, SortedList e Stack

Interfacce – Concetti di base

- Un'interfaccia non è altro che una raccolta denominata di **membri astratti** semanticamente correlati
- Le interfacce possono avere membri statici, tipi annidati, astratti, membri virtuali, proprietà ed eventi.
- Le interfacce non specificano una classe base o un modificatore di accesso.
- Forniscono **polimorfismo**
- Le interfacce **non** implementano alcun metodo

Interfacce – Concetti di base

- Le interfacce permettono di organizzare architetture generiche.
- I tipi che implementano l'interfaccia potranno accedere alle funzionalità definite nell'interfaccia ed utilizzarle
- L'interfaccia impone a ogni tipo che la implementa di esporre **membri pubblici** specifici che verranno utilizzati in un certo modo.

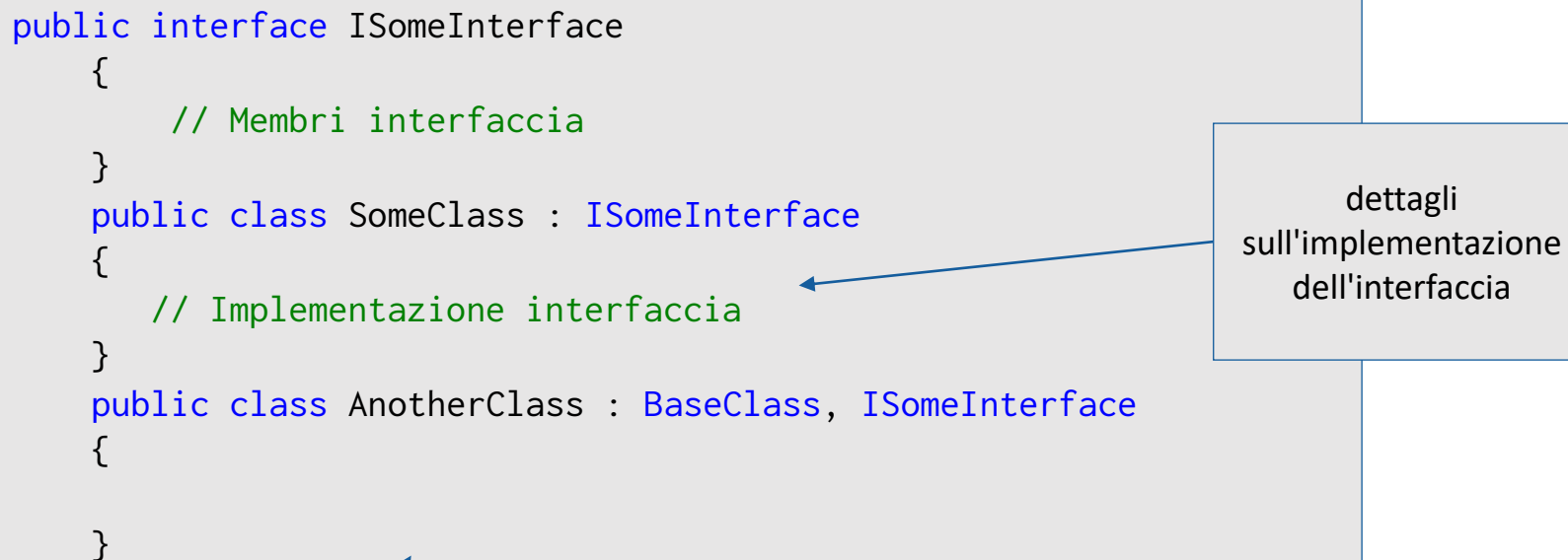
Implementare un'interfaccia

- Una classe che implementa un'interfaccia può implementare esplicitamente un membro di tale interfaccia
- Le interfacce possono essere implementate da **classi** e **struct**

Interfaccia - Esempio

```
public interface ISomeInterface
{
    // Membri interfaccia
}
public class SomeClass : ISomeInterface
{
    // Implementazione interfaccia
}
public class AnotherClass : BaseClass, ISomeInterface
{
}
```

dettagli
sull'implementazione
dell'interfaccia



deriva da BaseClass e
implementa l'interfaccia

Definizione Interfaccia - Esempio

```
public interface IAge
{
    int Age {get; }
    string Name {get;}
}
```

Implementazione Interfaccia - Esempio

```
class Person : IAge
{
    private string firstName;
    private string lastname = ".....";
    private int yearBorn;
    public int Age
    {
        get { return DateTime.Now.Year - yearBorn; }
        set { yearBorn = value; }
    }
    public string Name
    {
        get { return firstName + " " + lastname; }
        set { firstName = value; }
    }
}
```

Implementazione con
: **nomeInterfaccia**

Continua...

Utilizzo Interfaccia - Esempio

```
static void Main(string[] args)
{
    Person p = new Person();
    p.Name = "Geetha";
    p.Age = 1986;
    Console.WriteLine("Name = {0}", p.Name);
    Console.WriteLine("Age = {0}", p.Age);

    IAge[ ] AgeArray = new IAge[2];
    // .....
}
```

Interfaccia – Esempio 2

```
public interface IDimensions
{
    float Length();
    float Width();
}

class Box : IDimensions
{
    float lengthInches;
    float widthInches;

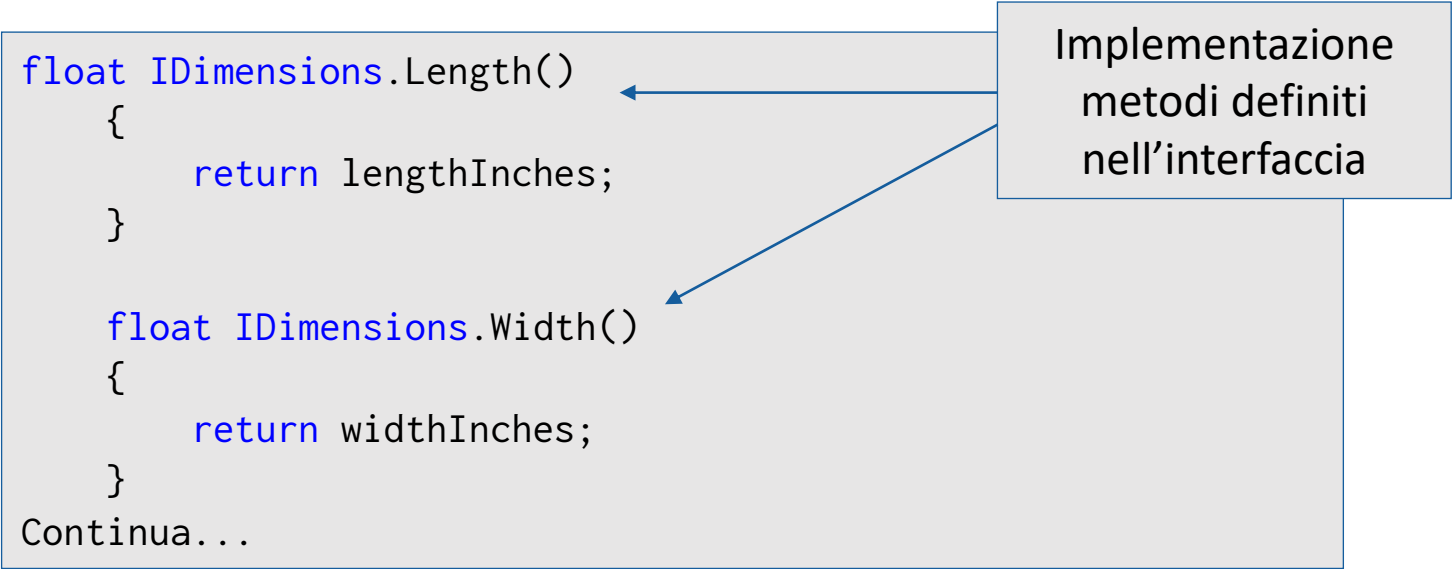
    public Box(float length, float width)
    {
        lengthInches = length;
        widthInches = width;
    }
}

Continua...
```

Interfaccia – Esempio 2

```
float IDimensions.Length()  
{  
    return lengthInches;  
}  
  
float IDimensions.Width()  
{  
    return widthInches;  
}  
Continua...
```

Implementazione
metodi definiti
nell'interfaccia



Interfaccia – Esempio 2

```
public static void Main()
{
    Box myBox = new Box(30.0f, 20.0f);

    IDimensions myDimensions = (IDimensions) myBox;
    /*
    Console.WriteLine("Length: {0}", myDimensions.Length());
    Console.WriteLine("Width: {0}", myDimensions.Width());
    */
}
```

Interfacce – as / is

- Utilizzando la parola chiave "as", è possibile fare riferimento ai membri dell'interfaccia

```
Hexagon h2 = new Hexagon();  
IPoints ipt2; ←  
ipt2 = h2 as IPoints;  
Console.WriteLine(ipt2.GetNumberOfPoints());
```

Interfaccia IPoints

- ipt2 può essere impostato su **null** se una determinata interfaccia non è supportata dall'oggetto
- É anche possibile utilizzare la parola chiave "is"

```
Hexagon h2 = new Hexagon();  
if (h2 is IPoints) Console.WriteLine(ipt2.GetNumberOfPoints());  
else //ERROR...
```

Interfacce e Parametri

Le interfacce possono essere usate come parametri nei metodi:

```
public interface IDraw3D
{
    void Draw3D();
}
```

Supporre ora che la sottoclasse Circle sia stata rivista per fornire un'implementazione aggiuntiva per IDraw

```
public class Circle : Shape, IDraw3D
{
    // .....
    public void Draw3D()
    { Console.WriteLine("Drawing a 3D Circle");
    }
}
```


Interfacce come parametri

Il metodo seguente dichiara l'interfaccia come parametro:

```
public static void DrawIn3D(IDraw3D int3D)
{
    Console.WriteLine("Drawing in 3D:");
    int3D.Draw3D();
}
public static void Main()
{
    ...
    if (s[i] is IDraw3D) DrawIn3D((IDraw3D)s[i]);
    ...
}
```

Domande & approfondimenti

Academy .NET