OOPs Concepts in C# : Inheritance and Abstract

By

Narasimha Rao T

Microsoft.Net FSD Trainer

Professional Development Trainer

tnrao.trainer@gmail.com

1. Access Modifier

Access modifiers control the **visibility** and **accessibility** of classes, methods, and other members.

Types:

Modifier	Description
`public`	Accessible from anywhere.
`private`	Accessible only within the containing class.
`protected`	Accessible within the containing class and by derived classes.
`internal`	Accessible within the same assembly.
`protected internal`	Accessible within the same assembly and by derived classes.

2. Inheritance

Allows a class to inherit members from another class.

Syntax:

```
class Parent {
    public void Greet() => Console.WriteLine("Hello");
}

class Child : Parent {
    public void Play() => Console.WriteLine("Playing");
}
```

- **Single Inheritance** is supported in C# (one base class).
- Use `: BaseClass` syntax to inherit.

3. Usage of `base` Keyword

Used to **refer to the base class** from a derived class.

Use Cases:

- Calling **base class constructor**
- Calling **base class method or property**

Example

```
class Parent {
    public virtual void Show() => Console.WriteLine("Parent");
}

class Child : Parent {
    public override void Show() {
        base.Show(); // Call base class method
        Console.WriteLine("Child");
    }
}
```

4. Abstract Classes & Methods

Used to define a base class with **incomplete implementation**.

Abstract Class:

- Cannot be instantiated.
- Can have **abstract** and **non-abstract** members.
- Used as a base class.

Abstract Class Example

```
abstract class Animal {
    public abstract void MakeSound(); // Abstract method
    public void Sleep() {
        Console.WriteLine("Sleeping...");
    }
}
```

5. Abstract Method:

- Declared in abstract class.
- No body (implementation) in base class.
- Must be **overridden** in derived class.

```
class Dog : Animal {
    public override void MakeSound() {
        Console.WriteLine("Bark");
    }
}
```

6. Interfaces

Definition:

An interface defines a contract. A class or struct that implements an interface must implement all of its members.

Syntax:

```
interface IShape {
    void Draw();
}

class Circle : IShape {
    public void Draw() {
        Console.WriteLine("Drawing Circle");
    }
}
```

Q & A

Narasimha Rao T
tnrao.trainer@gmail.com