abstract class

در زبان **C#**، یک **کلاس انتزاعی** **(Abstract Class)**کلاسی است که نمی‌توان از آن **نمونه‌سازی** کرد و ممکن است شامل **متدهای انتزاعی** (بدون پیاده‌سازی) و همچنین متدهای معمولی باشد. کلاس‌های انتزاعی برای تعریف یک پایه مشترک برای کلاس‌های مشتق استفاده می‌شوند.

```
;using System


namespace AbstractClassExample
}
    // تعریف کلاس انتزاعی
    abstract class Car
    }
        public string Brand { get; set; }


        // متد انتزاعی
        ;()public abstract void StartEngine


        // متد انتزاعی
        ;()public abstract void Drive


        // متد معمولی
        ()public void DisplayInfo
        }
```

```
        ;Console.WriteLine($"This is a car of brand: {Brand}")
                {
                  {

// کلاس مشتق برای ماشین بنزینی
                class GasolineCar : Car
                }
            ()public override void StartEngine
                }
;Console.WriteLine("Starting gasoline car engine with a roar!")
                {

            ()public override void Drive
                }
        ;Console.WriteLine("Driving the gasoline car.")
                {
                  {

// کلاس مشتق برای ماشین برقی
                class ElectricCar : Car
                }
            ()public override void StartEngine
                }
;Console.WriteLine("Starting electric car engine silently!")
                {

            ()public override void Drive
```

```
        }
    ;Console.WriteLine("Driving the electric car.")
        {
    {

class Program
    }
static void Main(string[] args)
    }
    // ایجاد یک شیء از ماشین بنزینی
;()Car gasolineCar = new GasolineCar
;"gasolineCar.Brand = "Toyota
;()gasolineCar.DisplayInfo
;()gasolineCar.StartEngine
;()gasolineCar.Drive

;()Console.WriteLine

    // ایجاد یک شیء از ماشین برقی
;()Car electricCar = new ElectricCar
;"electricCar.Brand = "Tesla
;()electricCar.DisplayInfo
;()electricCar.StartEngine
;()electricCar.Drive
        {
    {
  {
```

sealed class

یک **کلاس مهر و موم شده (Sealed Class)** به این معناست که کلاس نهایی است و دیگر قابل گسترش نیست. این ویژگی برای جلوگیری از ارث‌بری ناخواسته یا بهبود کارایی برنامه استفاده می‌شود

```
;using System


namespace SealedClassExample
}
    // تعریف کلاس مهر و موم شده
    sealed class FinalCar
    }
        ()public void Start
        }
            ;Console.WriteLine("Car is starting...")
        {

        ()public void Stop
        }
            ;Console.WriteLine("Car has stopped.")
        {
    {
```

```
// این کد خطا می‌دهد چون FinalCar مهر و موم شده است.
*/
class SportsCar : FinalCar
}
()public void Boost
}
;Console.WriteLine("Boost activated!")
{
{
/*

class Program
}
static void Main(string[] args)
}
;()FinalCar myCar = new FinalCar
;()myCar.Start
;()myCar.Stop
{
{
```

{

partial class

در زبان **C#**، کلمه کلیدی **`partial`** به شما این امکان را می‌دهد که یک کلاس، ساختار (**struct**) یا اینترفیس (**interface**) را به چندین فایل جداگانه تقسیم کنید. این قابلیت برای زمانی مفید است که بخواهید یک کلاس بسیار بزرگ یا پیچیده را مدیریت کنید، یا وقتی ابزارهای خودکار (مانند طراحی‌های ویژوال استودیو) بخشی از کد کلاس را تولید می‌کنند.

ساختار کلی :

```
File1.cs //

partial class MyClass

}

()public void Method1

}

;Console.WriteLine("This is Method1.")

{

{


File2.cs //

partial class MyClass

}

()public void Method2

}

;Console.WriteLine("This is Method2.")
```

```
{
  {


Program.cs //
class Program
}
static void Main(string[] args)
}
MyClass obj = new MyClass();
obj.Method1();
obj.Method2();
{
  {
```

polymorohism

یکی از ویژگی‌های اصلی برنامه‌نویسی **پلی‌مورفیسم (Polymorphism)** شیءگرا است و به معنای توانایی موجودیت‌ها برای نمایش رفتارهای مختلف ، پلی‌مورفیسم به دو شکل زیر پیاده‌سازی C# در شرایط مختلف است. در زبان می‌شود:

1. **پلی‌مورفیسم زمان کامپایل (Compile-Time Polymorphism)**:
   - با استفاده از **متدهای overload** و **اپراتورهای overload** پیاده‌سازی می‌شود.
   - به عنوان **Static Polymorphism** نیز شناخته می‌شود.
2. **پلی‌مورفیسم زمان اجرا (Run-Time Polymorphism)**:
   - با استفاده از **متدهای virtual، override** و **abstract** پیاده‌سازی می‌شود.

نیز شناخته می‌شود **Dynamic Polymorphism**به عنوان ○.

```
using System;

namespace PolymorphismExample
{
    class Calculator
    {
        // متد برای جمع دو عدد
        public int Add(int a, int b)
        {
            return a + b;
        }

        // متد برای جمع سه عدد
        public int Add(int a, int b, int c)
        {
            return a + b + c;
        }
    }

    class Program
    {
```

```
static void Main(string[] args)
}
;()Calculator calc = new Calculator
;Console.WriteLine(calc.Add(10, 20)) // خروجی:
۳۰
;Console.WriteLine(calc.Add(10, 20, 30)) // خروجی:
۶۰
{
{
{
```

Overriding

Overridingدر C# یکی از مفاهیم کلیدی برنامه‌نویسی شیءگرا است که به شما اجازه می‌دهد رفتار متدهای کلاس پایه **(Base Class)** را در کلاس مشتق **(Derived Class)**بازنویسی کنید. این ویژگی برای **پلی‌مورفیسم زمان اجرا (Run-Time Polymorphism)**ضروری است

```
;using System

namespace OverridingExample
}
// کلاس پایه
class Animal
}
()public virtual void Speak
```

```
}
;Console.WriteLine("Animal makes a sound.")
{
  {

// کلاس مشتق: سگ
class Dog : Animal
}
()public override void Speak
}
;Console.WriteLine("Dog barks.")
{
  {

// کلاس مشتق: گربه
class Cat : Animal
}
()public override void Speak
}
;Console.WriteLine("Cat meows.")
{
  {
```

```
class Program
}
static void Main(string[] args)
}
// کلاس پایه
;Animal myAnimal

// استفاده از کلاس مشتق: Dog
;()myAnimal = new Dog
.Dog barks :خروجی // ;()myAnimal.Speak

// استفاده از کلاس مشتق: Cat
;()myAnimal = new Cat
.Cat meows :خروجی // ;()myAnimal.Speak
{
 {
  {
Array
```

آرایه (Array) در C#

یک **مجموعه از مقادیر هم‌نوع** است که در حافظه به صورت C# آرایه در پشت سر هم ذخیره می‌شوند. آرایه به شما این امکان را می‌دهد که مقادیر متعددی را در یک متغیر ذخیره کنید و با استفاده از **ایندکس** به عناصر آن دسترسی پیدا کنید.

```
;using System

namespace ArrayExample
}
class Program
}
static void Main(string[] args)
}
// تعریف و مقداردهی آرایه
;{ ۵۰ ,۴۰ ,۳۰ ,۲۰ ,۱۰ } = int[] numbers

// دسترسی به عناصر آرایه
;Console.WriteLine(numbers[0]) // خروجی: ۱۰
;Console.WriteLine(numbers[3]) // خروجی: ۴۰

// تغییر مقدار یک عنصر
;numbers[2] = 100
;Console.WriteLine(numbers[2]) // خروجی: ۱۰۰
{
 {
  {
```