بسم تعالی

تعدادی از مفاهیم سی شارپ

**کلاس‌های انتزاعی (Abstract Classes)**

* الگویی برای کلاس‌های دیگر: کلاس انتزاعی به عنوان یک الگو یا طرح کلی برای کلاس‌های مشتق شده عمل می‌کند. این کلاس‌ها نمی‌توانند نمونه‌سازی شوند و حداقل یک متد انتزاعی (بدون پیاده‌سازی) دارند.
* تعریف متدهای انتزاعی: متدهای انتزاعی در کلاس انتزاعی تعریف می‌شوند، اما پیاده‌سازی آن‌ها در کلاس‌های مشتق شده الزامی است. این امر باعث ایجاد یک قرارداد بین کلاس پایه و کلاس‌های مشتق شده می‌شود.
* کلاس انتزاعی می‌تواند شامل متدهای غیرانتزاعی (با پیاده‌سازی) نیز باشد.

**مثال:**

public abstract class Animal

{

public abstract void MakeSound(); // متد انتزاعی

public void Eat() // متد غیرانتزاعی

{

Console.WriteLine("Eating...");

}

}

**کلاس‌های مهر و موم شده (Sealed Classes)**

* جلوگیری از مشتق‌گیری: کلاس مهر و موم شده کلاسی است که نمی‌توان از آن کلاس دیگری مشتق کرد. این ویژگی برای جلوگیری از تغییرات ناخواسته در رفتار یک کلاس استفاده می‌شود.

**مثال:**

public class Animal

{

Public virtual void MakeSound()

{

Consol.WriteLine("Some generic animal sound");

}

{

public sealed class Dog : Animal

{

public override void MakeSound()

{

Console.WriteLine("Woof!");

}

}

**کلاس‌های جزئی (Partial Classes)**

* تقسیم تعریف کلاس: کلاس جزئی کلاسی است که تعریف آن در چندین فایل مختلف تقسیم می‌شود. این ویژگی برای مدیریت بهتر کدهای بزرگ و پیچیده و همچنین برای ایجاد کتابخانه‌های قابل گسترش استفاده می‌شود.

**مثال:**

// Part1.cs

public partial class Person

{

public string FirstName { get; set; }

public string LastName { get; set; }

}

// Part2.cs

public partial class Person

{

public int Age { get; set; }

}

**چندریختی (Polymorphism)**

* رفتار متفاوت بر اساس نوع داده: چندریختی به توانایی یک متد یا عملگر برای رفتار متفاوت بر اساس نوع داده‌ای که روی آن اعمال می‌شود، گفته می‌شود.
* **چندریختی زمان اجرا (Runtime Polymorphism):** هنگامی رخ می‌دهد که یک متد در کلاس‌های مشتق شده بازنویسی شود.
* **چندریختی زمان کامپایل (Compile-time Polymorphism):** هنگامی رخ می‌دهد که متدها یا عملگرها با پارامترهای مختلف سربارگذاری شوند.

**مثال:**

// چندریختی زمان کامپایل

public class MathOperations

{

public int Add(int a, int b) => a + b;

public double Add(double a, double b) => a + b; // Overloading

}

// چندریختی زمان اجرا

public abstract class Animal

{

public abstract void MakeSound();

}

public class Dog : Animal

{

public override void MakeSound() => Console.WriteLine("Woof!");

}

**بازنویسی (Overriding)**

* پیاده‌سازی مجدد متد: بازنویسی به معنای پیاده‌سازی مجدد یک متد در کلاس مشتق شده با همان نام و پارامترها است. این کار زمانی انجام می‌شود که بخواهید رفتار پیش‌فرض متد کلاس پایه را تغییر دهید.

**نکته:**  
برای بازنویسی یک متد، باید از کلیدواژه‌های override در کلاس مشتق‌شده و virtual یا abstract در کلاس پایه استفاده کنید.

**مثال:**

public abstract class Animal

{

public abstract void MakeSound();

public virtual void Sleep()

{

Console.WriteLine("Animal is sleeping...");

}

}

public class Cat : Animal

{

public override void MakeSound()

{

Console.WriteLine("Meow!");

}

public override void Sleep()

{

Console.WriteLine("Cat is sleeping...");

}

}

**آرایه‌ها (Arrays)**

* مجموعه‌ای از عناصر هم‌نوع: آرایه مجموعه‌ای از متغیرهای هم‌نوع است که با استفاده از یک اندیس به عناصر آن دسترسی پیدا می‌کنیم.
* برای دسترسی به عناصر و طول آرایه می‌توان از اندیس‌ها و خاصیت Length استفاده کرد.

**مثال ۱: مقداردهی عناصر آرایه به صورت جداگانه**

int[] numbers = new int[3]; // آرایه‌ای با سه خانه

numbers[0] = 10;

numbers[1] = 20;

numbers[2] = 30;

Console.WriteLine($"عدد اول: {numbers[0]}");

Console.WriteLine($"عدد دوم: {numbers[1]}");

Console.WriteLine($"عدد سوم: {numbers[2]}");