1. Abstract Class (کلاس انتزاعی)

یک کلاس انتزاعی کلاسی است که نمی‌توان از آن مستقیماً نمونه‌سازی کرد. این نوع کلاس معمولاً شامل متدهای تعریف‌نشده (انتزاعی) است که باید در کلاس‌های مشتق‌شده پیاده‌سازی شوند.

کلاس‌های انتزاعی بیشتر به‌عنوان الگو یا پایه‌ای برای کلاس‌های دیگر استفاده می‌شوند. این کلاس‌ها ممکن است شامل متدهای پیاده‌سازی‌شده باشند، اما باید حداقل یک متد انتزاعی داشته باشند. متدهای انتزاعی فقط امضای متد (نام و پارامترها) را دارند و هیچ کدی در بدنه آن‌ها نیست.

نکات:

نمی‌توان یک شیء از کلاس انتزاعی ساخت.

کلاس مشتق‌شده باید متدهای انتزاعی را پیاده‌سازی کند.

Csharp Copy code

```
abstract class Animal
{
    public abstract void
MakeSound(); // متدی انتزاعی
}

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

class Program
{
    static void Main()
    {
        Animal myDog = new Dog();
        myDog.MakeSound(); // خروجی:
Woof! Woof!
    }
}
```

2. Sealed Class (کلاس مهروموم شده)

کلاسی که با کلمه کلیدی sealed تعریف شده باشد، نمی‌تواند به‌عنوان کلاس پایه استفاده شود (یعنی نمی‌توان از آن ارث‌بری کرد).

کلاس‌های انتزاعی بیشتر به‌عنوان الگو یا پایه‌ای برای کلاس‌های دیگر استفاده می‌شوند. این کلاس‌ها ممکن است شامل متدهای پیاده‌سازی‌شده باشند، اما باید حداقل یک متد انتزاعی داشته باشند. متدهای انتزاعی فقط امضای متد (نام و پارامترها) را دارند و هیچ کدی در بدنه آن‌ها نیست.

نکات:

نمی‌توان یک شیء از کلاس انتزاعی ساخت.کلاس مشتق‌شده باید متدهای انتزاعی را پیاده سازی کرد.

```
sealed class MathOperations
{
    public int Add(int a, int b)
    {
        return a + b;
    }
}

// این کد خطا خواهد داد:
// class AdvancedMath :
MathOperations {}

class Program
{
    static void Main()
    {
        MathOperations math = new
MathOperations();

Console.WriteLine(math.Add(2,
3)); //  خروجی:5
    }
}
```

3. Partial Class (کلاس جزئی)

کلاس‌های جزئی به شما اجازه می‌دهند که تعریف یک کلاس را به چند فایل تقسیم کنید. این ویژگی معمولاً برای مرتب‌سازی کد در پروژه‌های بزرگ استفاده می‌شود.

سناریو کاربردی:

پروژه‌های بزرگ: برای تفکیک منطقی بخش‌های مختلف یک کلاس.

ویژوال استودیو: هنگام تولید کد خودکار (مانند طراحی فرم‌ها).

قواعد:

تمام بخش‌های کلاس باید در یک فضای نام (Namespace) قرار بگیرند.

فقط در زبان‌های سی‌شارپ و برخی زبان‌های دیگر مانند VB.NET استفاده می‌شود.

File1.cs

Csharp Copy code

```
partial class Person
{
    public string FirstName { get;
set; }
}
```

File2.cs

Csharp Copy code

```
partial class Person
{
    public string LastName { get;
set; }

    public void PrintFullName()
    {

Console.WriteLine($"{FirstName}
{LastName}");
    }
}

class Program
{
    static void Main()
    {
        Person person = new Person
        {
            FirstName = "John",
            LastName = "Doe"
        };
        person.PrintFullName(); //
خروجی: John Doe
    }
}
```

4. Polymorphism (چند ریختی)

چندریختی به توانایی استفاده از یک متد یا خاصیت به شکل‌های مختلف اشاره دارد. این مفهوم معمولاً با ارث‌بری و متدهای مجازی پیاده‌سازی می‌شود.

چندریختی دو نوع دارد:

1. چندریختی در زمان کامپایل: به آن Overloading گفته می‌شود.
2. چندریختی در زمان اجرا: به آن Overriding گفته می‌شود.

Csharp Copy code

```
class MathOperations
{
    public int Multiply(int a, int
b)
    {
        return a * b;
    }

    public double Multiply(double a,
double b)
    {
        return a * b;
    }
}

class Program
{
    static void Main()
    {
        MathOperations math = new
MathOperations();

Console.WriteLine(math.Multiply(2,
3));        //  خروجی: 6

Console.WriteLine(math.Multiply(2.5,
4.2));    //  خروجی: 10.5
    }
}
```

5. Overriding (بازنویسی متد)

بازنویسی متد به معنای تغییر رفتار متد تعریف‌شده در کلاس پایه در کلاس مشتق‌شده است. از کلیدواژه‌های virtual و overrideاستفاده می‌شود.

بازنویسی متد زمانی استفاده می‌شود که بخواهید رفتار پیش‌فرض یک متد تعریف‌شده در کلاس پایه را تغییر دهید. برای این کار:

کلاس پایه باید از virtual برای متد خود استفاده کند.

کلاس مشتق‌شده باید با override متد را بازنویسی کند.

نکته مهم: اگر بخواهید رفتار کلاس پایه حفظ شود، می‌توانید از base استفاده کنید.

Csharp Copy code

```
class Parent
{
    public virtual void
ShowMessage()
    {
        Console.WriteLine("Message
from Parent class");
    }
}

class Child : Parent
{
    public override void
ShowMessage()
    {
        Console.WriteLine("Message
from Child class");
    }
}

class Program
{
    static void Main()
    {
        Parent obj = new Child();
        obj.ShowMessage(); // خروجی:
Message from Child class
    }
}
```

6. Array (آرایه)

آرایه می‌تواند یک‌بعدی، دوبعدی یا چندبعدی باشد. همچنین می‌توان آرایه‌ای از اشیاء تعریف کرد.

آرایه مجموعه‌ای از عناصر هم‌نوع است که در حافظه پشت سر هم ذخیره می‌شوند.

Csharp Copy code

```
class Person
{
    public string Name { get; set; }
}

class Program
{
    static void Main()
    {
        Person[] people = {
            new Person { Name =
"Alice" },
            new Person { Name =
"Bob" }
        };

        foreach (var person in
people)
        {

Console.WriteLine(person.Name);
        }
        // خروجی:
        // Alice
        // Bob
    }
}
```