ارث بری از ساختارها

در سی‌شارپ، ارث‌بری از ساختارهای داده‌ای مانند لیست‌ها به توسعه‌دهندگان این امکان را می‌دهد که ویژگی‌ها و عملکردهای موجود را گسترش دهند و به نیازهای خاص خود پاسخ دهند. به عنوان مثال، می‌توان یک کلاس جدید ایجاد کرد که از کلاس لیست ارث‌بری می‌کند و متدها یا ویژگی‌های جدیدی را برای مدیریت داده‌ها اضافه می‌کند. این رویکرد نه تنها باعث افزایش قابلیت استفاده مجدد از کد می‌شود، بلکه به توسعه‌دهندگان اجازه می‌دهد تا ویژگی‌های خاصی را که در لیست‌های استاندارد موجود نیستند، به راحتی پیاده‌سازی کنند.

بازنویسی متدها (Method OverRiding)

بازنویسی متدها **(Method Overriding)** در زبان برنامه‌نویسی سی‌شارپ به قابلیت یک کلاس فرزند (مشتق) برای ارائه پیاده‌سازی جدیدی از یک متد تعریف‌شده در کلاس والد (پایه) اشاره دارد. زمانی که یک متد در کلاس پایه به‌صورت مجاز با استفاده از کلیدواژه **virtual** تعریف می‌شود، کلاس‌های فرزند می‌توانند این متد را با استفاده از کلیدواژه **override** بازنویسی کنند.

مثال : کلاس **Person** را بصورت زیر در نظر بگیرید:

Person.cs > ...

```
class Person
{
    public void sayhi()
    {
        Console.WriteLine("Hello from person");
    }
}
```

این کلاس را بعنوان کلاس والد در نظر می گیریم ، کلاس بعدی که تعریف می شود کلاسی بنام **Student** است که در مرحله اول کلاسی خالی است.

```
Student.cs > ...
class Student : Person { }
```

بهنگام نمونه سازی از این کلاس براحتی می توانید متد **sayhi** را فراخوانی کنید.

```
Program.cs
Student ob = new Student();
ob.sayhi();
```

```
PROBLEMS  OUTPUT  TERMINAL  ...  powershell

PS C:\p1000> dotnet run
Hello from person
```

برای بازنویسی این متد در کلاس **Student** باید ابتدا آن را بصورت **virtual** در کلاس والد و سپس بصورت **override** در کلاس فرزند تعریف نمود:

```
Person.cs > ...
class Person
{
    public virtual void sayhi()
    {
        Console.WriteLine("Hello from person");
    }
}
```

و کلاس دانشجو بصورت زیر خواهد بود :

```
Student.cs > ...
class Student : Person
{
    public override void sayhi()
    {
        // base.sayhi();
        Console.WriteLine("Hello from student");
    }
}
```

نکته : بهنگام بازنویسی بهتر است اول متد کلاس والد فراخوانی و سپس دستورات جدید به کلاس فرزند اضافه شوند.

کلاس های Abstract

کلاس **abstract** در سی‌شارپ به کلاس‌هایی اطلاق می‌شود که نمی‌توانند به‌طور مستقیم از آنها شیء ایجاد کرد و معمولاً به‌عنوان یک الگوی عمومی برای کلاس‌های مشتق‌شده عمل می‌کنند. این کلاس‌ها می‌توانند شامل متدهای **abstract** (انتزاعی) باشند که فقط امضای آنها تعریف شده و نیاز است که در کلاس‌های فرزند پیاده‌سازی شوند. همچنین، کلاس‌های **abstract** می‌توانند متدها و ویژگی‌های عادی را نیز شامل شوند. هدف از استفاده از کلاس‌های**abstract** ، فراهم کردن یک ساختار پایه برای کلاس‌های مشتق‌شده است تا توسعه‌دهندگان بتوانند رفتارهای مشترک را تعریف کرده و پیاده‌سازی‌های خاص را در کلاس‌های فرزند انجام دهند.

Person.cs > ...

```
abstract class Person
{
    public void sayhi()
    {
        Console.WriteLine("Hello from person");
    }
}
```

کلاس‌هایsealed

در سی‌شارپ به کلاس‌هایی گفته می‌شود که امکان وراثت از آنها وجود ندارد، به این معنی که نمی‌توانند پایه‌ای برای کلاس‌های دیگر شوند. با استفاده از کلیدواژه **sealed**در تعریف کلاس، توسعه‌دهندگان می‌توانند از وراثت ناخواسته جلوگیری کنند و تضمین کنند که پیاده‌سازی خاصی از یک کلاس به هیچ‌وجه تغییر نخواهد کرد. این ویژگی به‌ویژه در مواقعی کاربرد دارد که یک کلاس به‌طور کامل پیاده‌سازی شده و هیچ‌گونه تغییری در رفتار آن مطلوب نیست.

```
Student.cs > ...
sealed class Student : Person { }
```

تمرین ۱ : با استفاده از ارث بری از کلاس لیست ، برنامه ای بنویسید که دوعدد را در لیست قرار داده و آنها را در خروجی نمایش دهد.

```
Test.cs > ...
class Test : List<int>{ }
```

```
Program.cs
Test ob = new Test();
ob.Add(12);ob.Add(14);
foreach (int num in ob)
{
    Console.WriteLine(num);
}
```

مثال : با استفاده از ارث بری کلاسی بنام دانشجو را از نوع دیکشنری تعریف کنید که نام و نمره دانشجویان را در خود ذخیره کند و متدی بنام **addStudent** به کلاس فوق اضافه نمایید که نام دانشجویی را بعنوان پارامتر گرفته و نمره ای تصادفی برای او ایجاد کرده و در دیکشنری ذخیره سازی نماید.

```
Student.cs > ...
class Student : Dictionary<string, int>
{
    public void addStudent(string name)
    {
        Random r = new Random();
        this.Add(name, r.Next(0, 20));
    }
}
```

مثال : متدی بنام **showAllStudents** به کلاس فوق اضافه نمایید که نام و نمره دانشجویان را در خروجی نمایش دهد.

```
class Student : Dictionary<string, int>
    public void showAllStudents()
    {
        foreach (string s in this.Keys)
        {
            Console.WriteLine("{0} {1}",s,this[s]);
        }
    }
}
```

تمرین ۱ : متدی بنام **showPassNames** به کلاس **Student** اضافه نمایید تا نام دانشجویانی که نمره بالاتر از ۱۰ گرفته اند را در خروجی نمایش دهد.

تمرین ۲ : متدی بنام **updateMark** به کلاس **Student** اضافه نمایید تا نام و نمره دانشجویی را بعنوان گرفته و نمره او را تغیر دهد.

تمرین ۳ : متدی بنام **nameList** بنویسید که نام های موجود در کلاس **Student** را در لیستی قرار داده و آن را بازگرداند.

تمرین ۴ : متدی بنام **findMark** به کلاس **Student** اضافه نمایید که نام دانشجویی را بعنوان پارامتر گرفته و نمره او را بازگرداند.

نمرین ۵ : متدی بنام **maxMark** به کلاس **Student** اضافه نماید که نام دانشجویی که بزرگترین نمره را گرفته است را بازگرداند.

تمرین ۶ : متدی بنام **showAvg** به کلاس **Student** اضافه نمایید که معدل کل دانشجویان را بازگرداند.