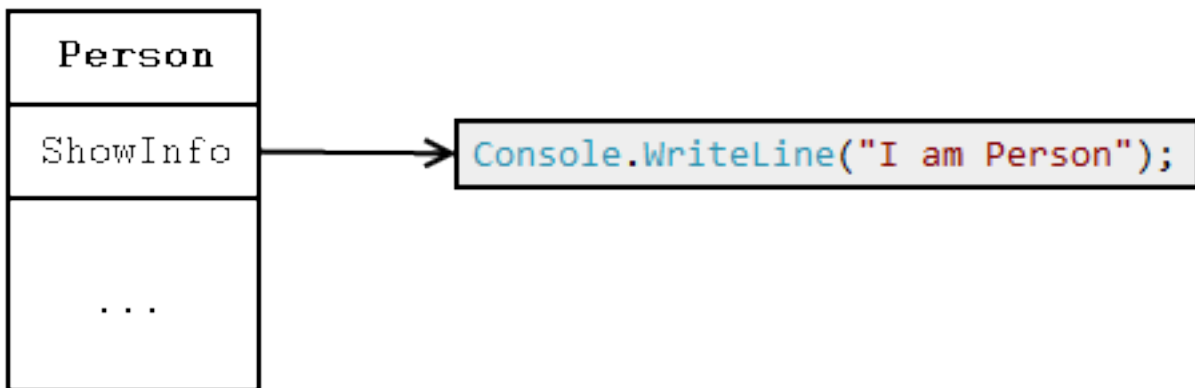


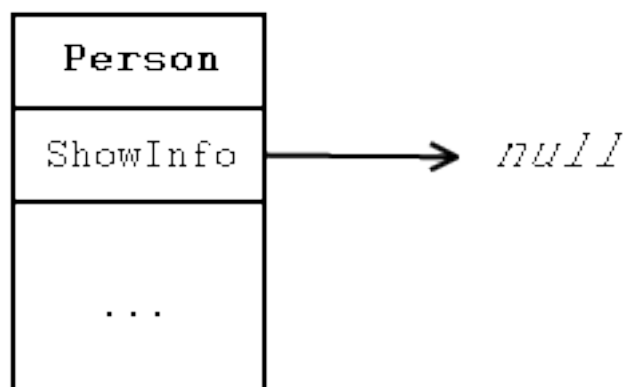
وقتی از کلاسی وهله‌ای را ایجاد می‌کنید، در واقع جدولی از اشاره گرها را به پیاده سازی متدهای آن ایجاد خواهید کرد. تصویر زیر مفهوم این جمله را بیان می‌کند.



متدها به روش‌های مختلفی تعریف می‌شوند و هر کدام رفتارهای مختلفی را در زمان ارث بری از خود نشان می‌دهند. روش استفاده استاندارد از آن‌ها مانند شکل بالاست ولی در صورتیکه بخواهید این روش را تغییر دهید می‌توانید به آن‌ها کلمات کلیدی اضافه کنید.

#### Abstract methodها

abstract متدها به هیچ جایی اشاره نمی‌کنند. مانند شکل زیر:



در صورتی که کلاسهای شما دارای اعضای abstract باشند، باید خودشان نیز abstract باشند. شما نمی‌توانید از این کلاس‌ها وهله‌ایی ایجاد نمایید؛ ولی می‌توانید از آن‌ها در ارث بری سایر کلاسها استفاده کنید و از سایر کلاس‌ها یک وهله ایجاد نمایید.

```
public abstract class Person
{
    public abstract void ShowInfo();
}

public class Teacher : Person
{
    public override void ShowInfo()
    {
        Console.WriteLine("I am a teacher!");
    }
}

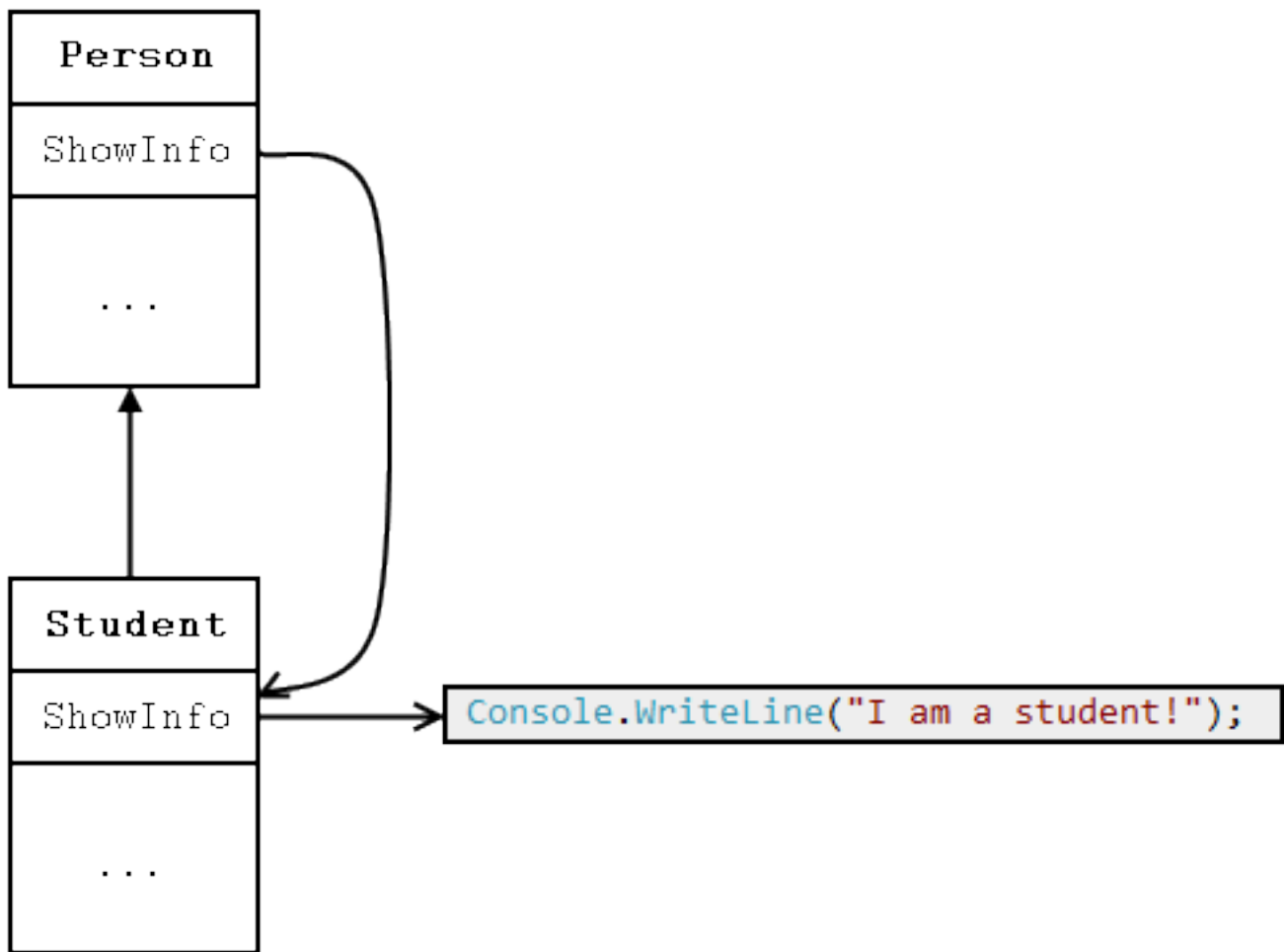
public class Student : Person
{
    public override void ShowInfo()
    {
        Console.WriteLine("I am a student!");
    }
}
```

در مثال بالا رفتار متد ShowInfo() به پیاده سازی آن در کلاس مربوطه بستگی دارد.

```
Person person = new Teacher();
person.ShowInfo();    // Shows 'I am a teacher!'

person = new Student();
person.ShowInfo();    // Shows 'I am a student!'
```

همانگونه که مشاهده می‌کنید متد ShowInfo() کلاس‌های Teacher و Student از کلاس Person ارث بری کرده اند ولی زمانی که از آن‌ها درخواست نمایش اطلاعات می‌کنید هر کدام رفتارهای مختلفی از خود نشان می‌دهند. خب چه اتفاقاتی در پشت صحنه رخ می‌دهد در حالیکه آن‌ها از Person ارث بری کرده اند؟ تا قبل از پیاده سازی متد ShowInfo()، اشاره گر به جایی اشاره نمی‌کند. ولی زمانی که به عنوان مثال یک وهله از Student ایجاد می‌کنید، اشاره گر به پیاده سازی واقعی متد ShowInfo() در کلاس Student اشاره می‌کند.



#### Virtual methodها

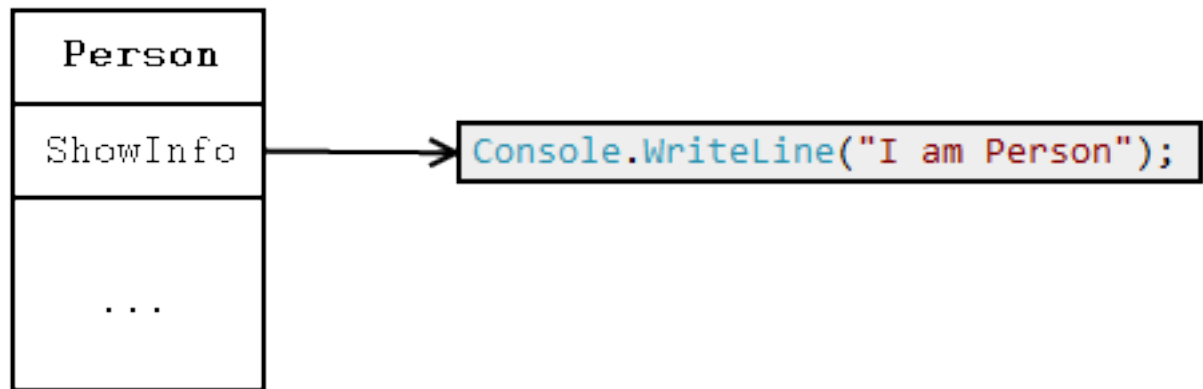
از کلاسهای دارای Virtual method می‌توان یک وهله ایجاد کرد و حتی امکان تحریف Virtual methodهای کلاس پایه در Derived-class وجود دارد. مانند کد زیر:

```

public class Person
{
    public virtual void ShowInfo()
    {
        Console.WriteLine("I am a person!");
    }
}

public class Teacher : Person
{
    public override void ShowInfo()
    {
        Console.WriteLine("I am a teacher!");
    }
}
    
```

در مثال بالا هم عضو `Person.ShowInfo` به جایی اشاره نمی‌کند، پس چرا می‌توانیم از آن یک وهله ایجاد کنیم؟!



باید توجه داشته باشید تصویر بالا با تصویر اول تفاوتی ندارد بدلیل اینکه اینک `virtual method`ها به روش استاندارد پیاده سازی اشاره می کنند. با استفاده از کلمه کلید `virtual` شما به عنوان مثال به کلاس `Person` می گوئید که متد `ShowInfo()` می تواند پیاده سازی های دیگری هم شاید داشته باشد و سایر پیاده سازی های دیگر این متد باید با کلمه کلیدی `override` در کلاس دیگر (`Teacher`) مشخص شوند.

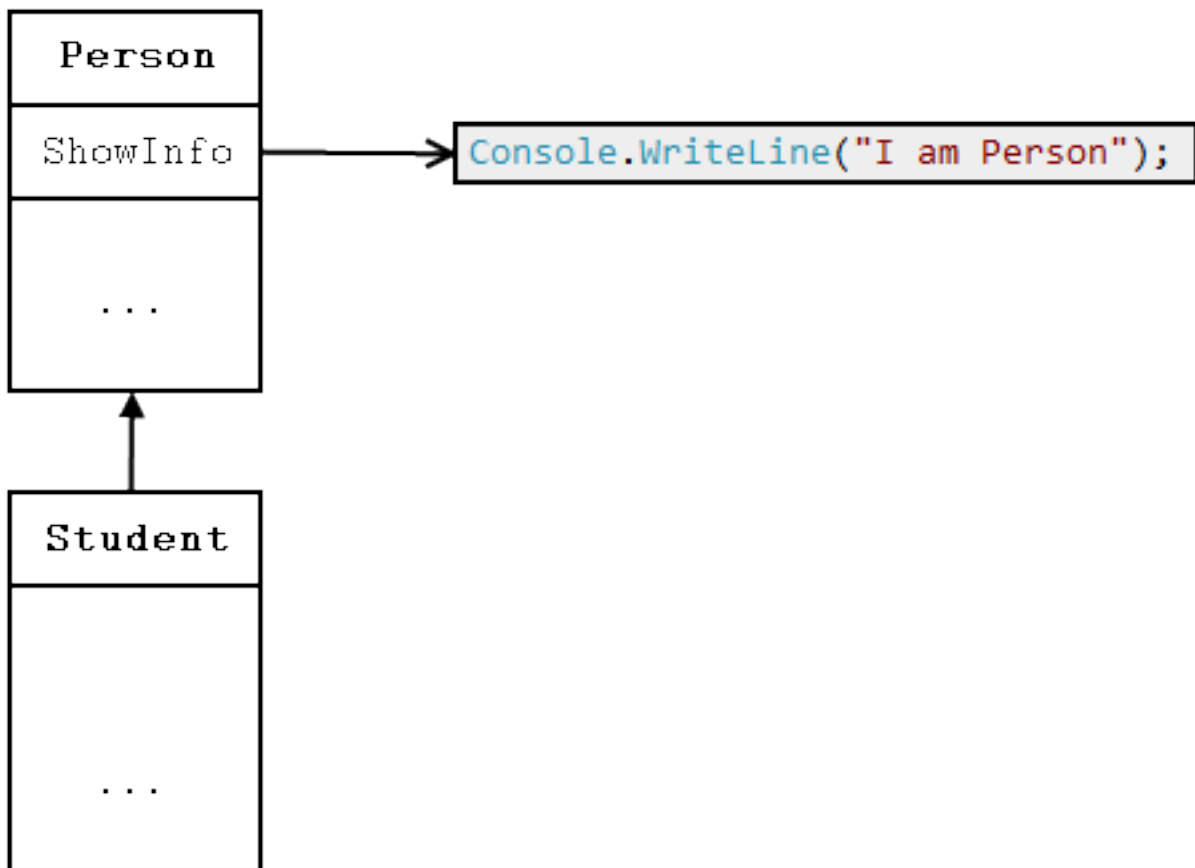
در ضمن در صورتیکه پیاده سازی دیگری از آن متد ارائه نشود از پیاده سازی کلاس پایه استفاده می شود.

```

public class Student : Person
{
}

Person person = new Teacher();
person.ShowInfo();    // Shows 'I am a teacher!'

person = new Student();
person.ShowInfo();    // Shows 'I am a person!'
  
```



#### نکته پایانی:

کلمه کلیدی **new** در متدهای کلاس‌های پایه و **Derived** مانند shadowing عمل می‌کند. برای بیان بهتر به کد زیر توجه کنید:

```

public class Person
{
    public void ShowInfo()
    {
        Console.WriteLine("I am Person");
    }
}

public class Teacher : Person
{
    public new void ShowInfo()
    {
        Console.WriteLine("I am Teacher");
    }
}
    
```

همانطور که ملاحظه می‌کنید متد **ShowInfo()** در کلاس پایه و **Derived** دارای پیاده‌سازی متفاوتی است برای این نوع پیاده‌سازی متد **ShowInfo()** در کلاس **Teacher** از کلمه کلیدی **new** برای عمل shadowing استفاده کرده ایم.

