عنوان: **اصول طراحی شی گرا SOLID - #بخش سوم اصل LSP**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۷/۰۶ ۲۰:۴۰
آدرس: www.dotnettips.info
برچسب‌ها: OOP, SOLID Principals, Object Oriented Design, Liskov substitution

بخش‌های پیشین : اصول طراحی شی گرا SOLID - #بخش اول اصل SRP
اصول طراحی شی گرا SOLID - #بخش دوم اصل OCP

اصل 3 (L – LSP – Liskov substitution principle
اصل **LSP** میگوید : **"زیر کلاس‌ها باید بتوانند جایگزین نوع پایه‌ی خود باشند".**
مقایسه با جهان واقعی :

شغل یک پدر تجارت املاک است درحالی که پسرش دوست دارد فوتبالیست شود.
یک پسر هیچگاه نمیتواند جایگزین پدرش شود، با اینکه که آنها به یک سلسله مراتب خانوادگی تعلق دارند.
در یک مثال عمومی‌تر بررسی میکنیم :
به طور معمول زمانی که ما در مورد اشکال هندسی صحبت میکنیم ، مستطیل را یک کلاس پایه برای مربع میدانیم. به کد زیر توجه کنید :

```
public class Rectangle
{
    public int Width { get; set; }
    public int Height { get; set; }
}

public class Square:Rectangle
{
    //codes specific to
    //square will be added
}
```

و میتوان گفت :

```
Rectangle o = new Rectangle();
o.Width = 5;
o.Height = 6;
```

بسیار خوب، اما با توجه به LSP باید قادر باشیم مستطیل را با مربع جایگزین کنیم. سعی میکنیم این کار را انجام دهیم :

```
Rectangle o = new Square();
o.Width = 5;
o.Height = 6;
```

موضوع چیست؟ مگر مربع می‌تواند طول و عرض نا برابر داشته باشد؟! امکان ندارد.
خوب این به چه معنی است؟ به این معنی که ما نمیتوانیم کلاس پایه را با کلاس مشتق شده جایگزین کنیم و باز هم این معنی را میدهد که ما داریم اصل LSP را نقض میکنیم.
آیا ما میتوانیم طول و عرض را در کلاس Square طبق کد زیر دوباره نویسی کنیم؟

```
public class Square : Rectangle
{
    public override int Width
    {
        get{return base.Width;}
        set
        {
            base.Height = value;
            base.Width = value;
        }
    }
    public override int Height
    {
        get{return base.Height;}
        set
        {
            base.Height = value;
            base.Width = value;
        }
    }
}
```

باز هم اصل LSP نقض میشود چون ما داریم رفتار خاصیت‌های طول و عرض در کلاس مشتق شده را تغییر میدهیم. ولی با توجه به کد بالا یک مستطیل نمیتواند طول و عرض برابر داشته باشد چون در صورت برابری دیگر مستطیل نیست.

اما راه حل چیست؟

یک کلاس انتزاعی (abstract) را به شکل زیر ایجاد و سپس دوکلاس Square و Rectangle را از آن مشتق میکنیم :

```
public abstract class Shape
{
    public virtual int Width { get; set; }
    public virtual int Height { get; set; }
}
```

همکنون ما دو کلاس مستقل از هم داریم. یکی Square و دیگری Rectangle که هر دو از کلاس Shape مشتق شده اند.

حالا میتوانیم بنویسیم :

```
Shape o = new Rectangle();
o.Width = 5;
o.Height = 6;

Shape o = new Square();
o.Width = 5; //both height and width become 5
o.Height = 6; //both height and width become 6
```

زمانی که ما در مورد اشکال هندسی صحبت میکنیم ، هیچ قاعده‌ی خاصی جهت اندازه‌ی طول و عرض نیست. ممکن است برابر باشند یا نباشند.

در قسمت بعدی اصل ISP را مورد بررسی قرار خواهیم داد.

نظرات خوانندگان

نویسنده: فدورا
تاریخ: ۱۳۹۲/۰۷/۰۹ ۱۰:۳۸

سلام.
خیلی ممنون از بابت مقالات آموزشی خوبتون.
فقط سوالی برای من تو این بخش سوم پیش اومد و اون هم اینکه بعد از تعریف کلاس abstract تعریف کلاس‌های rectangle و square به چه شکل شد؟ لطفا کد اون کلاس‌ها رو هم اضافه کنید.
با تشکر

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۷/۰۹ ۱۳:۴۸

ممنون.
کلاس های Rectangle و Square هر دو به همون شکل باقی میمونند با این تفاوت که هر دو از کلاس Shape مشتق شده اند و میتوانند خاصیت‌های Width و Height را طبق نیاز خود دوباره نویسی کنند (override). کلاس Restangle:

```
public class Rectangle : Shape
{
    //شما میتوانید خاصیت‌ها طول و عرض در کلاس پایه را در صورت نیاز دوباره نویسی کنید
}
```

کلاس Square :

```
public class Square : Shape
{
    //دوباره نویسی کردن خاصیت‌های طول و عرض در کلاس پایه جهت برابر کردن طول و عرض مربع
    public override int Width
    {
        get{return base.Width;}
        set
        {
            base.Height = value;
            base.Width = value;
        }
    }
    public override int Height
    {
        get{return base.Height;}
        set
        {
            base.Height = value;
            base.Width = value;
        }
    }
}
```

که با توجه به کدهای بالا ، کلاسهای مشتق شده‌ی Square و Restangle میتوانند جایگزین کلاس پایه خود یعنی Shape شوند :

```
Shape o = new Rectangle();
o.Width = 5;
o.Height = 6;

Shape o = new Square();
o.Width = 5; //طول و عرض هر دو برابر 5 میشوند
o.Height = 6; //عرض و طول هر دو برابر 6 میشوند
```