دیزاین پترن FACTORY

CREATIONAL DESIGN PATTERNS

یکی از زیرشاخه های الگوهای طراحی، Creational Design Patterns است. این الگو راهی جهت ساخت و ایجاد اشیاء (object) از کلاس ها را ارائه میدهد. ساخت یک شیء با استفاده از new کردن آن کلاس در اصطلاح hard code کردن راه حل خوبی نیست و بهتر است از الگوهای طراحی Creational استفاده کرد.

FACTORY DESIGN PATTERN

این پترن یکی از پرکاربردترین الگوهای طراحی از نوع Creational است.

هدف factory pattern این است که یک interface یا abstract class بسازید ولی فرزندان تصمیم میگیرند که کدام شیء را ایجاد کنند.

میزان استفاده : 1 2 3 4 5 زیاد

فرض کنید به جای اینکه یک شیء از یک کلاس را خودتان با استفاده از دستور new ایجاد کنید، آن را از کارخانه تولید کننده اشیاء دریافت کنید و با آن کار کنید.

این پترن شبیه به یک آژانس استخدام منابع انسانی است. باید شخصی را جهت انجام استخدام افراد بگذارید که اطلاعات مورد نیاز برای استخدام رده ها و سمت های مورد نیاز را

بدانند. در نتیجه وقتی یک کارمند جدید میخواهند استخدام کنند، فقط از طریق آن بخش اقدام به استخدام میکنند. دیگر واحدی که نیاز به یک کارمند جدید دارد نیازی ندارد تا از تمامی جزئیات کارمند جدیدالاستخدام اطلاعی داشته باشد چون تمام مواردی که مورد نیاز به بررسی بوده توسط واحد منابع انسانی بررسی و تایید شده.

ایجاد یک شی توسط الگوی طراحی کارخانه هم به همین شکل است. شما نیازی نیست تا اطلاعات زیادی در رابطه با نحوه ساخت یک شیء جدید بدانید. بلکه این وظیفه را به کلاسی که از پیش نیازها آگاه است سپرده اید و صرفا یک شیء درخواست کرده اید.
فرض کنید اینترفیسی تحت عنوان IShape به این صورت داریم.

```
public interface IShape
{
    void Draw();
}
```

این اینترفیس یک متد جهت رسم شکل دارد.

همچنین سه کلاس مستطیل، مربع و دایره را میخواهیم بصورتی که این interface را implement کند بصورت زیر مینویسیم.

```
public class Rectangle : IShape
{
    public int Width { get; set; }
    public int Height { get; set; }
    public void Draw()
    {
        Console.WriteLine("Rectangle Draw()");
    }
}
```

```
public class Square : IShape
{
    public int Width { get; set; }
    public void Draw()
    {
        Console.WriteLine("Square Draw()");
    }
}
```

```
public class Circle : IShape
{
    public int Radius { get; set; }
    public void Draw()
    {
        Console.WriteLine("Circle Draw()");
    }
}
```

یک enum جهت تعیین نوع شکل ها بصورت زیر مینویسیم.

```
public enum ShapesEnum
{
    RECTANGLE = 10,
    SQUARE = 20,
    CIRCLE = 30
}
```

```
}
```

کلاس کارخانه ی تولید اشیاء را با نام ShapeFactory بصورت زیر برنامه نویسی میکنیم.

```
internal class ShapeFactory
{
    public static IShape GetShape(ShapesEnum shape)
    {
        switch (shape)
        {
            case ShapesEnum.RECTANGLE:
                return new Rectangle();
            case ShapesEnum.SQUARE:
                return new Square();
            case ShapesEnum.CIRCLE:
                return new Circle();
            default:
                return null;
        }
    }
}
```

همان طور که مشاهده میشود، کلاس کارخانه به این صورت عمل میکند که با توجه به نوع شکل درخواستی که از ورودی متد به عنوان پارامتر ارسال میشود، یک شیء جدید ایجاد میکند و آن را از نوع اینترفیس IShape باز میگرداند.

حالا متد main را جهت استفاده از کارخانه به این صورت مینویسیم.

```
static void Main(string[] args)
{
    try
    {
        var circle = ShapeFactory.GetShape(ShapesEnum.CIRCLE);
        circle.Draw();

        var rectangle = ShapeFactory.GetShape(ShapesEnum.RECTANGLE);
        rectangle.Draw();

        var square = ShapeFactory.GetShape(ShapesEnum.SQUARE);
        square.Draw();
    }
    catch (Exception ex)
    {
        ShowError(ex.Message);
    }
    Console.ReadLine();
}
```

با استفاده از کارخانه شکل سازی، میتوانیم دایره، مستطیل و مربع خود را دریافت کنیم و متد draw را جهت رسم شکل آن صدا بزنیم.

هولوسن
با من یاد بگیر

آموزش های بیشتر در وبسایت هولوسن : https://holosen.net