
# دیزاین پترن ABSTRACT 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.

## ABSTRACT FACTORY DESIGN PATTERN

این پترن یکی از زیرشاخه های الگوهای طراحی از نوع Creational است.

گاهی اوقات این پترن را با نام factory of factory هم میشناسند. در آموزش قبلی در رابطه با Factory pattern صحبت کردیم. abstract factory pattern در حقیقت کارخانه ای از کارخانه هاست.

میزان استفاده : 1 2 3 4 5 زیاد

در دیزاین پترن factory گفتیم که 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.

در دیزاین پترن abstract factory در حقیقت در یک سطح بالاتر، برای ساخت اشیاء مشابه و

هم خانواده که میتوانند در یک کارخانه قرار بگیرند، میتوانیم از این الگو استفاده کنیم.
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
}
```

یک اینترفیس با نام IShapeFactory به منظور معرفی رفتار کارخانه شکل سازی بصورت زیر مینویسیم.

```
internal interface IShapeFactory
{
    IShape GetShape(ShapesEnum shape);
}
```

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.
حال فرض کنید اینترفیسی تحت عنوان IColor به این صورت داریم.

```
public interface IColor
{
    void Fill();
}
```

این اینترفیس یک متد جهت رنگ کردن یک شکل را دارد.

همچنین سه کلاس قرمز، سبز و آبی را میخواهیم بصورتی که این interface را implement کند بصورت زیر بنویسیم.

```
public class Red : IColor
{
    public void Fill()
    {
        Console.ForegroundColor = ConsoleColor.Red;
        Console.WriteLine("RED");
    }
}
```

```
public class Green : IColor
{
    public void Fill()
    {
        Console.ForegroundColor = ConsoleColor.Green;
        Console.WriteLine("Green");
    }
}
```

```
public class Blue : IColor
{
    public void Fill()
    {
        Console.ForegroundColor = ConsoleColor.Blue;
        Console.WriteLine("Blue");
```

```
    }
}
```

یک enum جهت تعیین رنگ ها بصورت زیر مینویسیم.

```
public enum ColorsEnum
{
    RED = 10,
    GREEN = 20,
    BLUE = 30
}
```

یک اینترفیس با نام IColorFactory را به منظور معرفی رفتار کارخانه رنگ ها بصورت زیر مینویسیم.

```
internal interface IColorFactory
{
    IColor GetColor(ColorsEnum color);
}
```

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

```
internal class ColorFactory : IColorFactory
{
    public IColor GetColor(ColorsEnum color)
    {
        switch (color)
        {
            case ColorsEnum.RED:
                return new Red();
            case ColorsEnum.GREEN:
                return new Green();
            case ColorsEnum.BLUE:
                return new Blue();
            default:
                return null;
        }
    }
}
```

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

حال نوبت به کلاس اصلی کارخانه ای از کارخانه ها میرسیم. برای نوشتن این کلاس یک اینترفیس بصورت زیر مینویسیم.

```
internal interface IAbstractFactory
{
    IShape GetShape(ShapesEnum shape);
    IColor GetColor(ColorsEnum color);
}
```

کلاس پیاده سازی abstract factory را با همین نام AbstractFactory به شکل زیر مینویسیم.

```
internal class AbstractFactory : IAbstractFactory
```

```
{
    public IColor GetColor(ColorsEnum color)
    {
        var colorFactory = new ColorFactory();
        return colorFactory.GetColor(color);
    }

    public IShape GetShape(ShapesEnum shape)
    {
        var shapeFactory = new ShapeFactory();
        return shapeFactory.GetShape(shape);
    }
}
```

همانطور که مشاهده میکنید، این کلاس از دو متد اصلی GetColor جهت دریافت رنگ و GetShape جهت دریافت شکل ساخته شده است.

با توجه به نوع ورودی هر متد یک کارخانه از نوع درخواستی ایجاد میشود و متد مربوطه اجرا و مقدار درخواستی ایجاد و بازگردانده میشود. زمانی که درخواست ایجاد شکل را میدهیم از GetShape و زمانی که درخواست رنگ کردن را میدهیم از GetColor استفاده باید کرد.

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

```
static void Main(string[] args)
{
    try
    {
        var abstractFactory = new HelperClass.AbstractFactory();

        var red = abstractFactory.GetColor(ColorsEnum.RED);
        red.Fill();
        var circle = abstractFactory.GetShape(ShapesEnum.CIRCLE);
        circle.Draw();

        var green = abstractFactory.GetColor(ColorsEnum.GREEN);
        green.Fill();
        var square = abstractFactory.GetShape(ShapesEnum.SQUARE);
        square.Draw();
    }
    catch (Exception ex)
    {
        ShowError(ex.Message);
    }
    Console.ReadLine();
}
```

با استفاده از کارخانه ای از کارخانه ها، اقدام به دریافت رنگ قرمز، با صدا زدن متد Fill از کلاس رنگ و رسم شکل دایره با صدا زدن متد draw از کلاس شکل، و سپس رنگ سبز جهت رسم شکل مربع میکنیم.


HOLOSEN

# هولوسن
# با من یاد بگیر

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

**برچسب‌ها** دیزاین پترن