عنوان: **آشنایی با Refactoring - قسمت 8**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۲۱ ۰۰:۴۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Refactoring

یکی از اشتباهاتی که همه‌ی ما کم و بیش به آن دچار هستیم ایجاد کلاس‌هایی هستند که «زیاد می‌دانند». اصطلاحا به آن‌ها God Classes هم می‌گویند و برای نمونه، پسوند یا پیشوند Util دارند. این نوع کلاس‌ها اصل SRP را زیر سؤال می‌برند (Single responsibility principle). برای مثال یک فایل ایجاد می‌شود و داخل آن از انواع و اقسام متدهای «کمکی» کار با دیتابیس تا رسم نمودار تا تبدیل تاریخ میلادی به شمسی و ... در طی بیش از 10 هزار سطر قرار می‌گیرند. یا برای مثال گروه بندی‌های خاصی را در این یک فایل از طریق کامنت‌های نوشته شده برای قسمت‌های مختلف می‌توان یافت. Refactoring مرتبط با این نوع کلاس‌هایی که «زیاد می‌دانند»، تجزیه آن‌ها به کلاس‌های کوچکتر، با تعداد وظیفه‌ی کمتر است.
به عنوان نمونه کلاس CustomerService زیر، دو گروه کار متفاوت را انجام می‌دهد. ثبت و بازیابی اطلاعات ثبت نام یک مشتری و همچنین محاسبات مرتبط با سفارشات مشتری‌ها:

```
using System;
using System.Collections.Generic;

namespace Refactoring.Day8.RemoveGodClasses.Before
{
    public class CustomerService
    {
        public decimal CalculateOrderDiscount(IEnumerable<string> products, string customer)
        {
            // do work
            throw new NotImplementedException();
        }

        public bool CustomerIsValid(string customer, int order)
        {
            // do work
            throw new NotImplementedException();
        }

        public IEnumerable<string> GatherOrderErrors(IEnumerable<string> products, string customer)
        {
            // do work
            throw new NotImplementedException();
        }

        public void Register(string customer)
        {
            // do work
        }

        public void ForgotPassword(string customer)
        {
            // do work
        }
    }
}
```

بهتر است این دو گروه، به دو کلاس مجزا بر اساس وظایفی که دارند، تجزیه شوند. به این ترتیب نگهداری این نوع کلاس‌های کوچکتر در طول زمان ساده‌تر خواهند شد:

```
using System;
using System.Collections.Generic;

namespace Refactoring.Day8.RemoveGodClasses.After
{
    public class CustomerOrderService
```

```
    {
        public decimal CalculateOrderDiscount(IEnumerable<string> products, string customer)
        {
            // do work
            throw new NotImplementedException();
        }

        public bool CustomerIsValid(string customer, int order)
        {
            // do work
            throw new NotImplementedException();
        }

        public IEnumerable<string> GatherOrderErrors(IEnumerable<string> products, string customer)
        {
            // do work
            throw new NotImplementedException();
        }
    }
}
```

```
namespace Refactoring.Day8.RemoveGodClasses.After
{
    public class CustomerRegistrationService
    {
        public void Register(string customer)
        {
            // do work
        }

        public void ForgotPassword(string customer)
        {
            // do work
        }
    }
}
```