عنوان: **الگوی سردر Facade Pattern**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۵/۰۱/۲۳ ۱۳:۰
آدرس: www.dotnettips.info
گروه‌ها: Design patterns

یکی از الگوهای ساختاری Gang Of Four، استفاده از الگوی Facade است که پیچیدگی‌های یک سیستم را مخفی می‌سازد و با ارائه یک پیاده سازی ساده‌تر، استفاده از آن و تست آن را راحت‌تر می‌سازد. این الگو یک کلاس یا یک سیستمی را با متدها و رویدادهایی ساده، در اختیار ما قرار می‌دهد و در یک لحظه، تنها با یک کلاس واحد سر و کله می‌زنیم. احتمالا بسیاری از شما از این الگو استفاده کرده‌اید، ولی شاید با اسم آن آشنا نبوده‌اید.

کار این کلاس در واقع ترکیب کلاس‌ها و کتابخانه‌های کاری مشخص است که نیاز به ارتباط با یکدیگر را دارند. به عنوان مثال یک برنامه کتابخانه، برای وظیفه‌ای چون امانت یک کتاب نیاز است تا چندین کلاس مختلف را با یکدیگر به کار بگیرد که این وظایف شامل موارد زیر می‌باشند:

بررسی وجود کتاب
بررسی تعداد موجود یک کتاب در کتابخانه
بررسی وضعیت امانی کتاب (آیا کتاب در دست کسی از قبل امانت است؟ یا کتاب برای امانت آزاد است؟)
در صورتی که کتابی بیش از زمان مورد نظر در دست کسی امانت است، با یک پیامک از او بخواهیم که کتاب را بازگرداند.
تمامی موارد بالا تنها قسمتی از انجام یک عمل ساده هستند که در یک گروه جای می‌گیرند؛ ولی در واقع از چندین کلاس جدا مثل کلاس کتاب، امانت، سیستم پیامکی و ... استفاده شده است . الگوی Facade به ما کمک می‌کند تا پیچیدگی و تعداد خطوط اجرا را در سطوح بالاتر مخفی سازیم و تنها با صدا زدن یک یا چند متد ساده، کار را به اتمام برسانیم. این کار باعث کاهش کد و خوانایی برنامه در سطوح بالاتر می‌شود.

در کد زیر ما قصد داریم نمونه‌ای از اجرای این الگو را ببینیم. ابتدا سه کلاس اطلاعاتی زیر را ایجاد می‌کنیم:
کلاس کتاب Book:

```
class Book
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int Quantity { get; set; }
        public bool IsLoanable { get; set; }
    }
```

کلاس کاربر User

```
class User
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public  string CellPhoneNumber { get; set; }
    }
```

کلاس امانت Loan

```
class Loan
    {
       public DateTime ExpiredDate { get; set; }
        public User User { get; set; }
    }
```

بعد از آن سه کلاس را برای مدیریت کتاب، مدیریت امانت و مدیریت پیامکی، ایجاد می‌کنیم:

کلاس کتاب BookManager:

```
class BookManager
    {
        public Book GetBook(int id)
        {
                return new Book()
                {
                    Id=id,
                    Title = "a book",
                    Quantity = 3,
                    IsLoanable = true
                };
        }
    }
```

کلاس بالا یک کتاب را با موجودی سه عدد بازمی‌گرداند که خاصیت IsLoanable آن می‌گوید کتاب را برای بردن به خانه امانت می‌دهند؛ ولی اگر کتاب به فرض یک کتاب مرجع باشد باید False برگرداند تا از امانت آن خودداری شود.

کلاس LoanManager برای مدیریت امانت‌ها

```
public int IsLoan(int bookId)
        {
            return 2;
        }

        public List<Loan> GetLoans(int bookId)
        {
            return new List<Loan>()
            {
                new Loan()
                {
                    ExpiredDate = DateTime.Now.AddDays(-15),
                    User = new User()
                {
                    Id = 2,
                    Title = "User1",
                    CellPhoneNumber = "342342424"
                }
                },
                new Loan()
                {
                    ExpiredDate = DateTime.Now.AddDays(5),
                    User = new User()
                        {
                            Id = 56,
                            Title = "User56",
                            CellPhoneNumber = "324324324324"
                        }
                }
            };
        }
```

این کلاس شامل دو متد است که اولین متد آن کد کتاب را دریافت می‌کند و تعداد افرادی را که در حال حاضر نسخه‌های مختلف آن را به امانت برده‌اند، برمی‌گرداند. در کد بالا عدد دو بازگردانده می‌شود که میگوید از نسخه‌های موجود این کتاب در کتابخانه، دوتای آن‌ها به امانت برده شده‌اند. در متد دوم، کد کتاب داده شده و امانت‌های فعلی آن کتاب که همان دو عدد بالا می‌باشد را برگشت می‌دهد.

در نهایت سومی کلاس مدیریتی برای پیامک هاست:

```
class SmsManager
    {
        public void SendMessage(string number)
        {
            Console.WriteLine("please take back the book to the library : "+number);
        }
    }
```

و در کلاس Facade داریم

```
class FacadeBookLoan
    {
        private readonly BookManager _bookManager;
        private readonly LoanManager _loanManager;
        private readonly SmsManager _smsManager;
        public FacadeBookLoan()
        {
            _bookManager = new BookManager();
            _loanManager=new LoanManager();
            _smsManager=new SmsManager();
        }

        public int IsLoanable(int bookId)
        {
            var book = _bookManager.GetBook(2);
            if (book == null)
                return -2;
            if (!book.IsLoanable)
                return -1;
            var howManyBookIsLoaned = _loanManager.IsLoan(bookId);

            if(howManyBookIsLoaned>0) ManageLoaners(bookId);

            return book.Quantity - howManyBookIsLoaned;
        }

        private void ManageLoaners(int bookId)
        {
            var loans = _loanManager.GetLoans(bookId);

            foreach (var loan in loans)
            {
                if (loan.ExpiredDate > DateTime.Now)
                {
                    _smsManager.SendMessage(loan.User.CellPhoneNumber);
                }
            }
        }
    }
```

در این کلاس متد IsLoanable چک می‌کند که آیا کتاب قابل امانت هست یا خیر. در اینجا مرحله به مرحله، وجود کتاب و قابلیت امانت کتاب بررسی شده و در صورت نتیجه، کد وضعیت -1 یا -2 بازگردانده می‌شوند و در مرحله بعد تعداد نسخه‌های آن کتاب که در حال حاضر در دست امانت هستند، بررسی می‌شوند. اگر کتابی در درست امانت باشد، متد خصوصی صدا زده شده و به کاربرانی که بیش از مدت معینی یک نسخه از کتاب را داشته‌اند، پیامک می‌زند که کتاب را بازگردانید؛ چرا که نسخه‌ها در حال کاهش هستند و در مرحله بعد تعداد نسخه‌های موجود از آن کتاب را در کتابخانه باز می‌گرداند که در این مثال تنها یک نسخه از آن کتاب در کتابخانه موجود است و دو تای آن‌ها در دست امانت هستند که یکی از امانت دارها 15 روز است کتاب را در تاریخ معینی تحویل نداده است.

کد بدنه اصلی برنامه:

```
var myfacade=new FacadeBookLoan();
var loansCount= myfacade.IsLoanable(2);
Console.WriteLine(loansCount > 0 ? "you can loan the book" : "you can't loan the book");
```

خروجی برنامه:

```
please take back the book to the library : 324324324324
you can loan the book
```