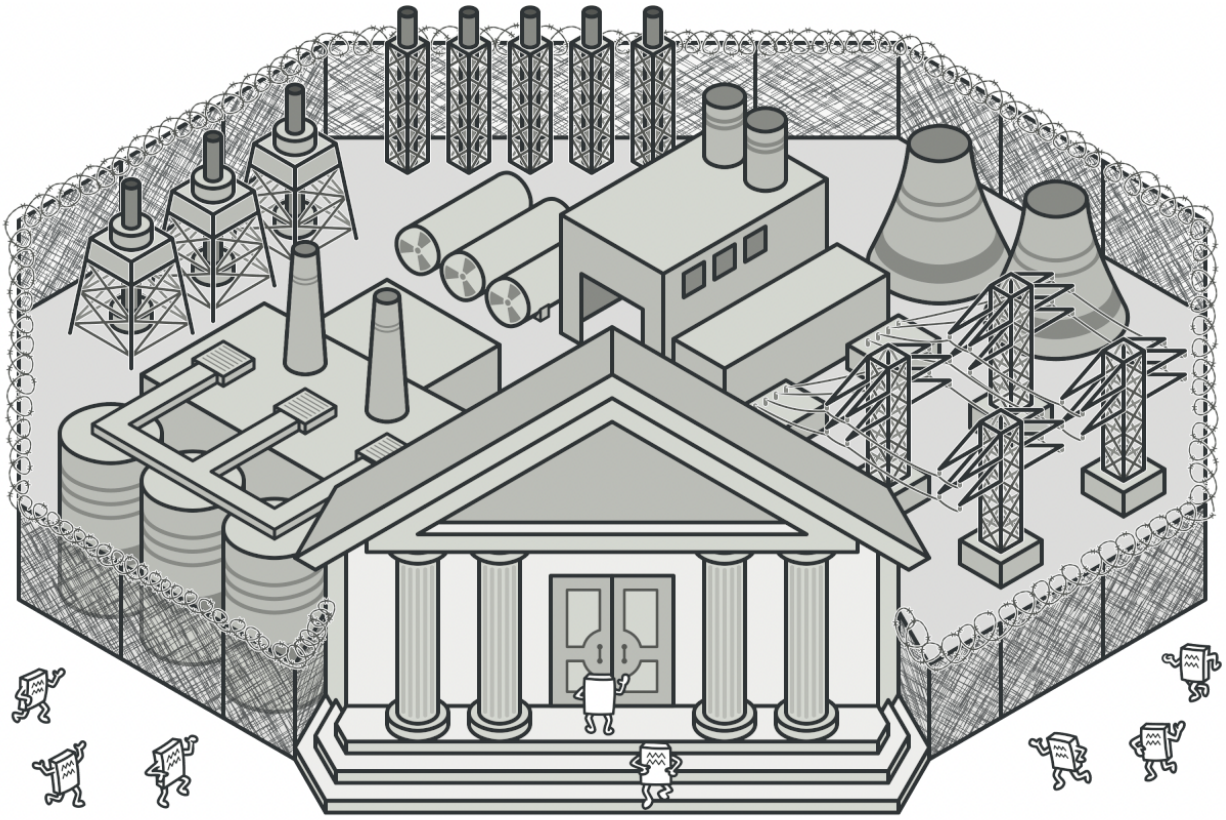
Facade

Structural design pattern



Nona Ghazizadeh

Shayan Mohammadizadeh

مقدمه

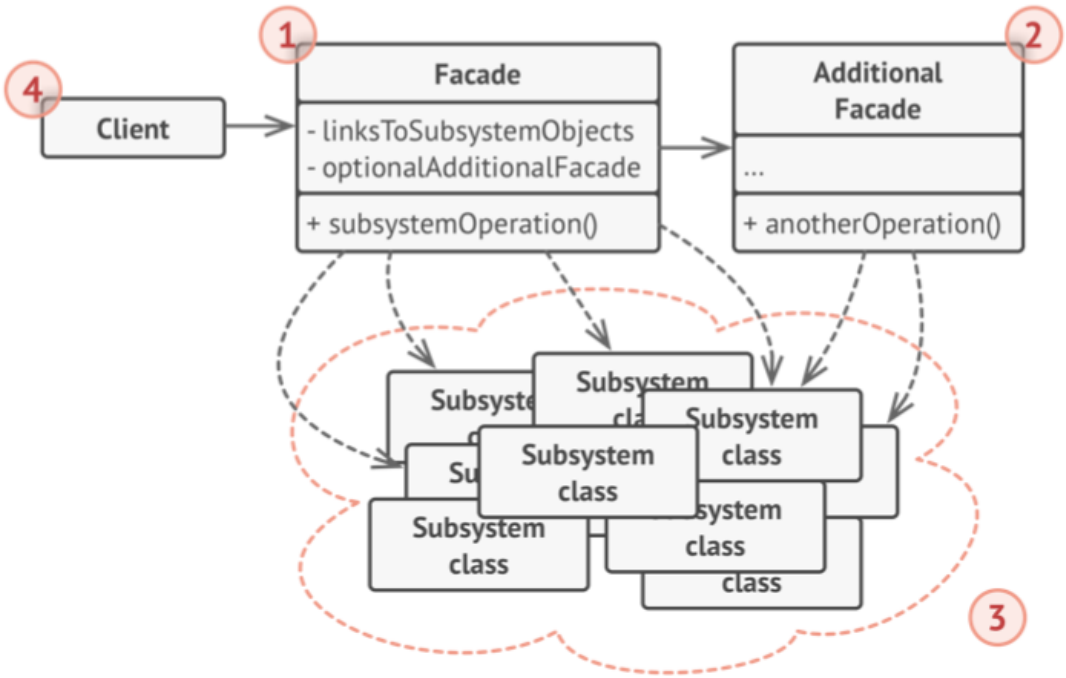
الگوی طراحی facade که در لغت به معنی ظاهر بیرونی یک ساختمان بزرگ است، دقیقا راه حلی متناسب با این معنی برای مشکل ما عرضه می‌کند. Facade یک الگوی طراحی ساختاری است که با منزوی کردن یک سیستم پیچیده با کلاس‌های گوناگون و طراحی یک در برای راه ارتباطی مشتری با این سیستم، متدهای مورد نیاز مشتری را بدون درگیر کردن وی با پیچیدگی سیستم در دسترس او قرار می‌دهد.

کاربرد

فرض کنید سایتی طراحی کرده‌ایم که در آن کاربر با وارد کردن مدل گوشی، قیمت آن گوشی در فروشگاه مورد نظر را پیدا کند. در ابتدا سایت ما ناشناخته بود و تنها تعداد کمی از فروشندگان را در سایت خود داشتیم، پس با طراحی یک متد برای هر فروشنده و در دسترس قرار دادن این متدها برای باقی مهندسان کاری کردیم که آنها برای پیدا کردن قیمت هر فروشنده با صدا زدن تابع آن فروشنده به خواسته خود برسند. اما با معروف شدن سایت با یک سیستم حجیم مواجه شدیم که هر مشتری (مهندس دیگر) با بخشی از آن کار داشت.

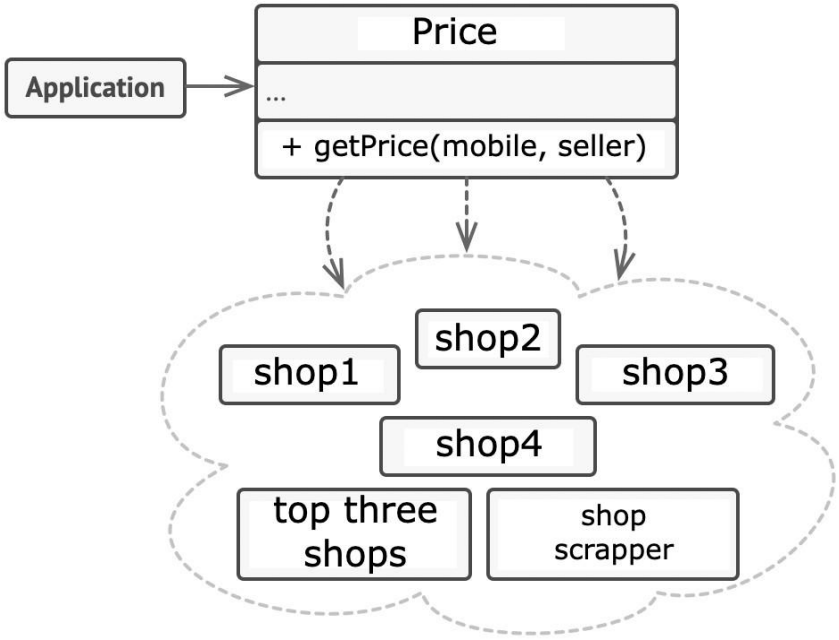
برای حل این مشکل تصمیم گرفتیم با منزوی کردن این سیستم و طراحی یک interface برای آن، کاری کنیم هر مشتری با وارد کردن نام یک فروشنده و یک مدل موبایل، به قیمت آن برسد. این دقیقا همان الگوی facade است. از طرفی با این کار می‌توانیم قابلیت‌های جدیدی به سیستم خود (از جمله دریافت کمترین قیمت از بین فروشندگان) اضافه کنیم و یا حتی دسترسی به برخی فروشندگان را محدود کنیم.

ساختار



1. facade با توجه به ورودی و خواسته کاربر به سیستم‌های زیرین رفته و خروجی خواسته شده را برمی‌گرداند. این قسمت به عملکردهای سیستم‌های زیرین شناخت دارد.
2. برای حجیم نشدن خود facade نیز می‌توان یک interface دیگری طراحی نمود که با سیستم‌های زیرین در ارتباط است.
3. تمامی عملکردهای اصلی در اینجا قرار دارند که عمدتا پیچیده و سخت فهم هستند.
4. مشتری تنها با interface طراحی شده ارتباط دارد و هیچ خبری از پیچیدگی سیستم‌های زیرین ندارد و در نتیجه کاملا از آنها مستقل است.

سودو کد




```
Class Shop1
// ...

Class Shop2
// ...

Class Shop3
// ...

Class Shop4
// ...

Class TopThreeShops
// ...
```

```
Class ShopScrapper
// ...

Class Price:
   Func getPrice:
      shop1 = Shop1.getShop()
      scrappedShop1 = ShopScrapper(shop1)
      ...
      Return result

Class Application:
   price = new Price()
   print(price.getPrice())
```

خوبی(ها) و بدی(ها)

😀 منزوی کردن سیستم پیچیده و نامفهوم که سبب ساده‌سازی استفاده برای مشتری می‌شود.

😀 در صورتی که کاربر بخواهد از کتابخانه یا framework دیگری استفاده کند تغییر برای او بسیار آسان‌تر خواهد بود.

😀 جداسازی سیستم و مشتری باعث پایداری بیشتر کد و دیباگ راحت‌تر آن می‌شود.

😢 ممکن است خود interface طراحی شده به یک سیستم با پیچیدگی حتی بیشتر از سیستم اصلی تبدیل شود.