**اللگوی Facade یال نمای بیرونی**

**نمونه دنیای واقعی :**

**فرض کنید قصد خرید برای منزل دارید و هیچ فروشگاهی در سطح شهر وجود ندارد و محصول مورد نیاز خودتان را باید از کارخانه یا باغ ها و ... تهیه کنید. برای این کار باید شما انرژی زمان هزینه زیادی را صرف کنید.**

**این فعالیت در دنیای امروز به راحتی قابل انجام است و شما تنها به مراجعه به فروشگاه های اطراف محل زندگیتان می توانید همه محصول مورد نظر را تهیه کنید. این فروشگاه ها نقش الگوی Facade را بازی میکنند و شما را از Subsystem ها جدا میکنند.مثلا برای تهیه لبنیات نیازی نیست شما مستقیم به کارخانه تولید لبنیات بروید که نقش SubSystem را ایفا میکنند ، فروشگاه ها با این SubSystem ها در ارتباط است.**

**نکات:**

1. **پیاده سازی ساده**
2. **عدم محدودیت در تعداد ( میتوان به ازای هر package یا SubSystem یک facade تولید کرد.)**
3. **واسط Facade به واسط subsystem یا package ها وابستگی ندارد**
4. **از هر facade تنها به یک نمونه نیاز داریم ( میتوان از الگوی singleton برای الگوی Facade استفاده کرد ویا Facade از جنس static باشند)**
5. **Facade ممکن است چندین پیاده سازی داشته باشند که میتوان این کلاس را abstract تعریف کرد.**
6. **ایجاد لایه بندی برای Façade**
7. **ساده سازی کار با سیستم شلوغ**

مزایا

1. برقراری اتصال سست در برنامه

Client با subsystem یا پیکج ها اتصال محکم ( مستقیم ) ندارد.

1. لایه بندی سیستم

زمانی که Client و SubSystem ها بصورت جداگانه توسعه داده می شوند و مستقل از هم هستند.میتوان لایه بندی کرد که به صورت مستقیم از هم استفاده نکنند. وابستگی های پیچیده ای که client و Subsystem ها میتوانند باهم داشته باشند را با این الگو از بین برده میشود.

1. کلاینت ها از جزئیات سطح پایین SubSystem ها خبر ندارند.
2. کاهش وابستگی ها
3. کاهش وابستگی بین SubSystem ها ( حتی subsystem ها برای ارتباط باهم از Facade های هم استفاده کنند(.
4. اجبار استفاده از Facade برای Client وجود ندارد.

چه زمانی از الگوی Facade استفاده کنیم:

1. زمانی که SubSystemهای زیادی داریم و Client ها برای کارکردن با این SubSystem ها مجبور است کلاس های زیادی را استفاده کند یا objectهای زیادی را تولید کند وبخواهیم Client ها با object کمتری کار کنند.
2. زمانی که میخواهیم Client ها در برابر تغییرات و توسعه SubSystem ها آسیب نبینند.
3. زمانی که چند SubSystem پیچیده داریم و میخواهیم واسط کار با آنها را برای Client ساده کنیم.
4. زمانی که میخواهیم لایه بندی را بر روی سیستم خود اعمال کنیم.

الگوی های مرتبط

1. Singleton
2. Abstract Factory