Gang-of-Four tervezési minták 6

# Struktúrális minták

## Facade („Homlokzat”)

* Egységes interfészt definiál egy alrendszer interfészeinek halmazához.
* **Probléma**
  + Kód széleskörű objektumokkal rendelkezik, amik egy library-hez vagy keretrendszerhez tartozik.
  + Normális esetben az összes objektumot inicializálni kellene, nyomon követni a függőségeket, a metódusokat a megfelelő sorrendben végrehajtani és így tovább.
  + Ennek eredményeként az osztályok üzleti logikája szorosan összekapcsolódik a harmadik féltől származó osztályok megvalósítási részleteivel, ami megnehezíti a megértést és a karbantartást.
* **Megoldás**
  + Csak a tényleges funkciókat tartalmazza.
  + Praktikus, ha integrálni kell az alkalmazást integrálni kell egy library-vel, ami sok funkcióval rendelkezik, de csak egy kis részére van szükség.
    - **Példa:** Egy alkalmazás, ami rövid videókat tölt fel egy platformra, ami egy összetett videókonvertáló könyvtárat használ.
    - Tehát azon az osztályon belül eléri azt a metódust, amivel lehet konvertálni és azt hozzácsatoljuk a konvertáló library-vel, akkor megvan az első facade.

### Facade struktúra

A képen diagram látható

Automatikusan generált leírás

1. A facade hozzáférést biztosít egy adott részhez az alrendszer funkcionalitásához.
   1. Tudja hogyan kell a kéréseket működtetni.
2. Egy kiegészítő facade osztály „szennyezés” megelőzésére létrehozható.
3. A komplex alrendszer több különböző objektumból áll.
   1. Ahhoz, hogy ezek mindegyike értelmes tevékenységet végezzen, tudni kell az alrendszer megvalósításának részleteit, hogy például az objektumokat milyen sorrendben kell inicializálni.
   2. Az alrendszer osztályai nem tudnak a facade létezéséről.
   3. A rendszeren belül működnek és közvetlenül egymással dolgoznak.
4. A kliens a facade-t használja ahelyett, hogy közvetlenül hívnál az alrendszer objektumait.

### Facade használati esetek

* Akkor használjuk, ha korlátozott, de egyszerű interfészre van szüksége egy összetett alrendszerhez.
* Akkor használjuk, ha egy alrendszert rétegekre akarunk strukturálni.

### Facade implementációja

1. Nézzük meg, hogy lehet-e egyszerűbb interfészt biztosítani, mint amit egy meglévő alrendszer biztosít.
2. Interfész implementálása az új facade osztályban.
3. A facade-nek át kell irányítania a kódból érkező hívásokat az alrendszer megfelelő objektumaihoz.
4. Innentől kezdve a kódban csak a facade-en keresztül kommunikáljon az alrendszer.
   1. Mostantól a kód védve van az alrendszer kódjának bármilyen változásától.
   2. Ha egy alrendszer új verzióra frissül, csak a facade kódot kell módosítani.

### Facade előnye és hátránya

* **Előny**
  + Elszigetelhető a kód az alrendszer komplexitásától.
* **Hátrány**
  + „god object” lehet belőle

## Proxy

* Lehetővé teszi egy másik objektum helyettesítőjének vagy placeholder-jének/helytartójának biztosítását.
* A proxy ellenőrzi az eredeti objektumhoz való hozzáférést, lehetővé téve, hogy valamit elvégezzen, mielőtt vagy miután a kérés eljut az eredeti objektumhoz.
* **Probléma**
  + Miért akarjuk ellenőrizni az objektumhoz való hozzáférést?
    - Van egy hatalmas objektum, ami rengeteg rendszererőforrást fogyaszt és időnként szükség van rá, de nem mindig.
* **Lusta megoldás**
  + Csak akkor hozzuk létre az objektumot, amikor tényleg szükség van rá.
  + Végre kellene hajtani néhány késleltetett inicializálási kódot, de ez kód duplikációt okozna.
* **Megoldás**
  + Hozzunk létre egy új proxy osztályt, aminek interfésze megegyezik az eredeti service objektummal.
  + Ezután frissíti az alkalmazást, hogy átadja a proxy objektumot az eredeti objektum összes kliensének.
  + A klienstől érkező kérés fogadásakor a proxy létrehoz egy valódi service objektumot és mindent átad neki.
* **Haszna**
  + Ha valamit az osztály alapvető logikája előtt vagy után kell végrehajtani, a proxy lehetővé teszi, hogy ezt az osztály megváltoztatása nélkül tegye.
  + Mivel a proxy ugyanazt az interfészt valósítja meg, mint az eredeti osztály, átadható bármely olyan kliensek, ami valódi szolgáltatásobjektumot vár.

### Proxy struktúra

https://wiki.sch.bme.hu/Tervez%C3%A9si\_mint%C3%A1k\_(Sznik%C3%A1k)#Proxy

## Decorator

## Flyweight trükkök

## Flyweight a .NET osztályokban (String, Type).