Gang Of Four (GoF) minták

GoF névvel egy könyv négy szerzőjét szokták illetni, akik a könyvben a szoftvertervezés jellemző problémáira adnak ismétlődő válaszokat. A könyvük első felében az objektumorientált programozás lehetőségeit és csapdáit veszik át, a második részben a 23 klasszikus szoftver tervezési mintát.
Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides

- Jellemzően C++ és Smalltalk példákat használnak. Létrehozó: (creational)
  - A minták általában a megadott osztályok további kidolgozásával és példányosítással, illetve példányosításra használhatók…
- Szerkezeti: (structural)
  - A minták objektumösszetételekre épülnek
- Viselkedési (behavioral):
  - A minták a megadott osztályok példányai közötti kommunikációra épülnek

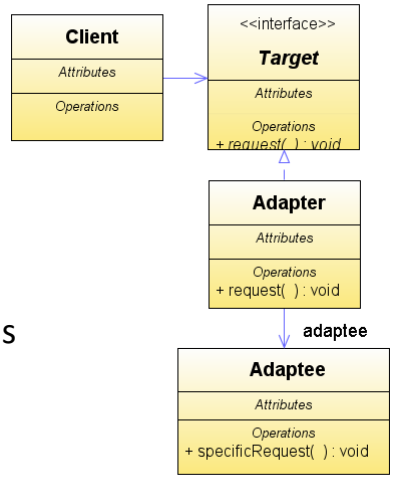
GoF szerkezeti minták

- Adaptor:
- Bridge:
- Composite:
- Decorator:
- Facade:
- Flyweight:
- Proxy:

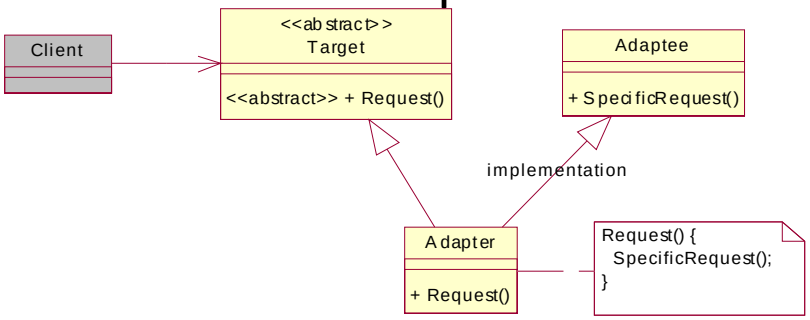
- **Szerkezeti minták: objektum összetételekre épülnek, elősegítik a tervezést azzal, hogy egy egyszerű**
- utat nyitnak az egyes példányok közti kapcsolatok felderítéséhez.
- o Adapter: burkoló osztály, egy osztály felületét átalakítja olyan módon, ahogy azt egy másik
- objektum elvárja.
- o Bridge: elválasztja a definíciót a megvalósítástól, így ezek külön tudnak változni egymástól.
- o Composite: több független objektumból képez egy fa-struktúrájú összetett objektumot, amit
- aztán, mint egyetlen objektumot lehet tovább kezelni.
- o Decorator: egy osztályhoz dinamikusan ad hozzá új műveleteket, kibővítve a funkcionalitást.
- o Facade: több interface számára biztosít egy egységes, egyszerűsített felületet.
- o Proxy: egy felület egy másik osztály felé, amin keresztül lehet kezelni az adott osztályt.
- Jellemzően egy osztályhoz több proxy tartozhat.

Adaptor

- Klasszikus Wrapper osztályszerkezet/Objektum adapter/objektumkompozícióval
- Target: interface, definíció, ezt kapja/használja/várja az ügyfél
- Client: ügyfél, aki használja
- Adapter: olyanná alakítja az Adaptee interfészt, amilyet a kliens szeretne
- Adaptee: ezen végzünk átalakítást



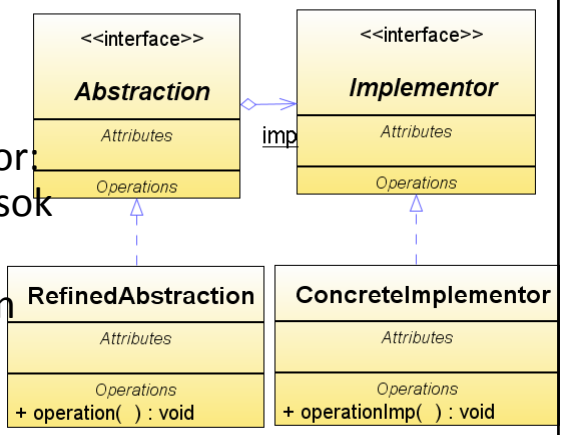
Adaptor



- Adapter osztályszerkezet többszörös öröklődéssel (osztály adapter)
- Target: interface, definíció, ezt kapja/használja/várja az ügyfél
- Client: ügyfél, aki használja
- Adapter: olyanná alakítja az Adaptee interfészt, amilyet a kliens szeretne
- Adaptee: ezen végzünk átalakítást

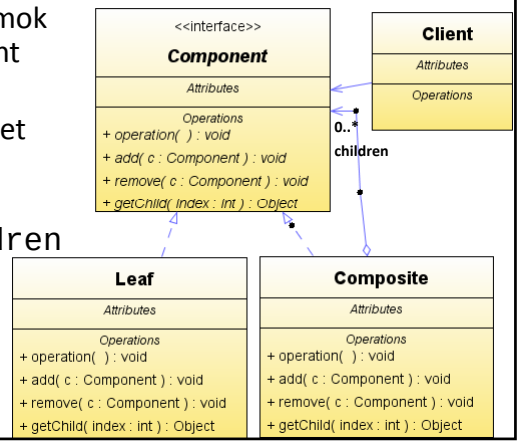
Bridge

- Abstraction-Implementor felületek: a szokásos Wrapper szerkezet
- RefinedAbstraction, ConcreteImplementor: konkrét megvalósítások
- Eredmény: a megvalósítás teljesen függetlenedik a definíciótól



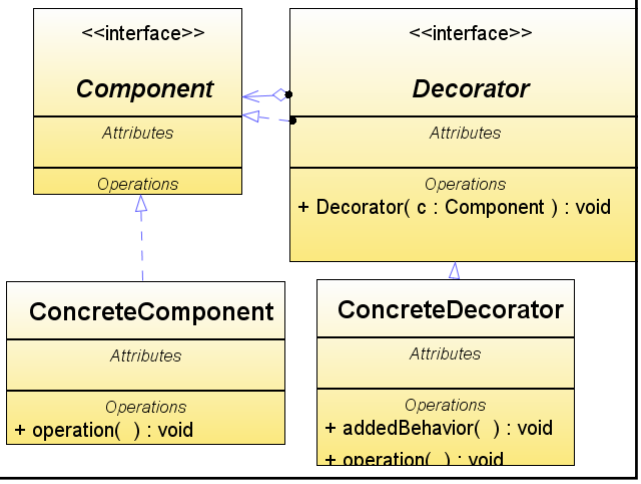
Composite

- Összetett objektumok kezelése, amikor objektumok egy csoportját úgy kezeljük, mintha egyetlen objektum volna.
- A ténylegesen egyszerű (Leaf) és az összetett (Composite) objektumok egyöntetű kezelése (Component felület)
- Az objektumok egy faszerkezetet feszítenek ki
- ```
  operation() {
    for each g in children
    g.operation(); }
  ```



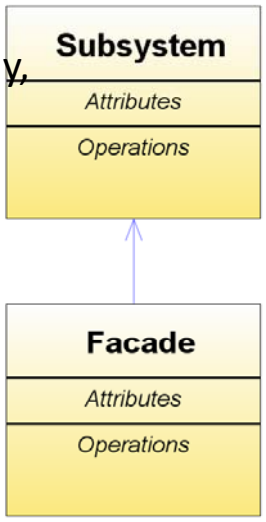
Decorator

- Decorator egy burkolóosztály, amivel új műveleteket adunk az eredeti (Component) osztályhoz.
- Decorator() (konstruktor) paramétere a kibővítendő Component példány.
- Valójában a ConcreteDecorator adja hozzá az új műveleteket.
- Előbb a Component objektumot hozzuk létre, majd ezzel létrehozzuk a Decoratort


  ```

Facade (homlokzat)

- A Facade osztály több különböző osztályhoz biztosít egyszerűsített felületet. Ilyen lehet pl. több osztály, alrendszer, vagy nagyobb lélegzetű objektumkönyvtár.

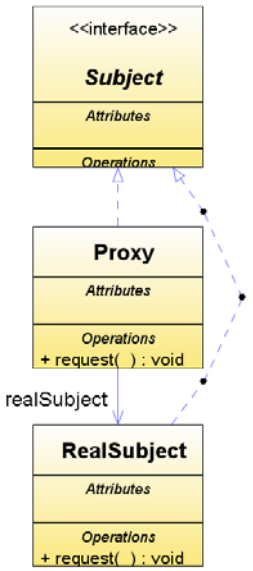


Proxy

- Objektum helyett egy helyettesítő objektumot használ, ami szabályozza az objektumhoz való hozzáférést.
- Pl. Szövegszerkesztő, sok nagyméretű képpel. Csak akkor jelenítjük meg/töltjük be a képeket, ha odagörgetünk
- Egyetlen példányt hozunk létre, és (szükség szerint) proxy példányt.
- Minden műveletet a megvalósított osztálypéldány végez el
- Ha a proxy megszűnik, akkor megszűnhet a megvalósított példány is

Proxy

- Subject: közös interfészt biztosít a Subject és a Proxy számára (ezáltal tud a minta működni, ez a lényeg!)
- RealSubject: a valódi objektum, amit a proxy elrejt
- Proxy: helyettesítő objektum. Tartalmaz egy referenciát a tényleges objektumra, hogy el tudja azt érni. Szabályozza a hozzáférést a tényleges objektumhoz, feladata lehet a tényleges objektum létrehozása és törlése is.



- Távoli Proxy
  - Távoli objektumok lokális megjelenítése átlátszó módon. A kliens nem is érzékeli, hogy a tényleges objektum egy másik címtartományban, vagy egy másik gépen van
- Virtuális Proxy
  - Nagy erőforrás igényű objektumok igény szerinti létrehozása (pl. kép)
- Védelmi Proxy
  - A hozzáférést szabályozza különböző jogok esetén
- Smart Pointer
  - Egy pointer egységbezárása, hogy bizonyos esetekben automatikus műveleteket hajtson végre (pl.:lockolás)

Flyweight (lepkesúly)

- Ha nagyon nagyszámú objektumot kezelünk, akkor fontos, hogy csak kevés felesleges adatot tároljunk
- Nincsenek osztályfüggvényeik, hogy a vtable pointer ne foglaljon helyet
- Pl: egy word-processor karakterei. Nem tárolhatunk minden betűt a formattálásával együtt – csupán egy hivatkozást tárol a formattálásra
- Azonosan formattált szövegszakaszok összefogása egyetlen blokkba