Tervezési minták egy OO programozási nyelvben

Kovács Kristóf

Z5MP5Y

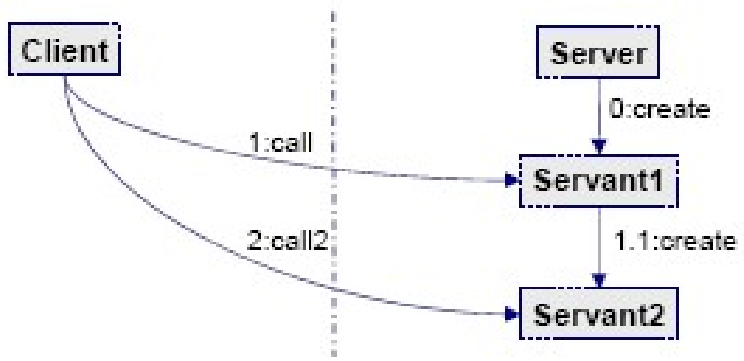
A tervezési minták fogalma Christopher Alexander építésztől származik. Ő kereste meg azokat az építészeti mintákat, amik gyakran megjelentek egy jól felépített ház jellemzésére.

1987-ben Kent Beck és Ward Cunningham kísérletezni kezdett hasonló minták használatával a programozásban, speciális mintanyelvvel. Eredményeiket az OOPSLA (Object-Oriented Programming & System) konferencián mutatták be ugyanebben az évben.

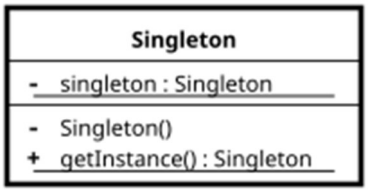
Ezen minták 1994-ben váltak széles körben ismertté. Erich Gramma, Richard Helm, Ralph Johnson és John Vlissides, avagy a GoF (Gang of Four) programozó csapat kiadta új könyvüket, „Programtervezési minták" címmel. Ez a könyv ma is az objektumorientált programozási minták kutatásának az alapja. Huszonhárom mintát mutat be, amely három kategóriába sorolja őket: létrehozási minta, szerkezeti minta, viselkedési minta.

A könyvben található definíció szerint a programtervezési minták: „egymással együttműködő objektumok és osztályok leírásai, amelyek testre szabott formában valamilyen általános tervezési problémát oldanak meg egy bizonyos összefüggésben".

A létrehozási minták (creational patterns) objektumok és osztályok létrehozására valók. Egy példa rá a Factory, ahol egy objektum hozza létre egy másik osztály példányait, inicializálja őket, és opcionálisan az életútjukat is követi. A szerver minden egyes új hívásnál új példányt hoz létre a kiszolgáló servantból.



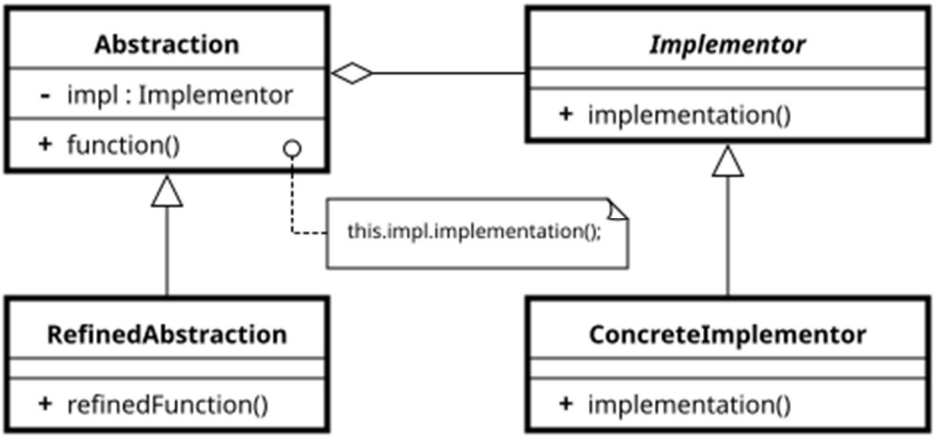
Az Egyke (Singleton) progamtervezési minta egy objektumra korlátozza az osztály létrehozható példányainak számát.



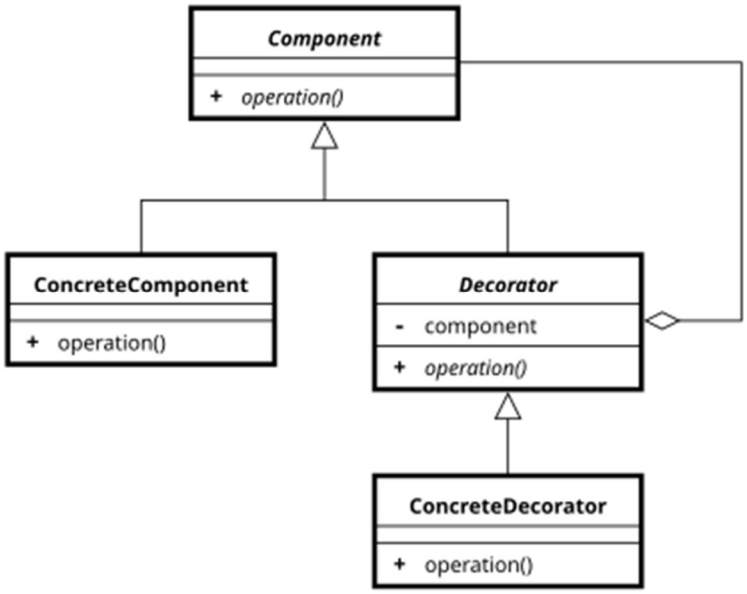

```
public final class Egyke {
   private static final Egyke PELDANY = new Egyke();

   private Egyke() {}

   public static Egyke aPeldany() {
      return PELDANY;
   }
}
```

A szerkezeti minták (structural patterns) egyszerűsítik a szoftverek tervezését az entitások közötti kapcsolatok egyszerű azonosításával.

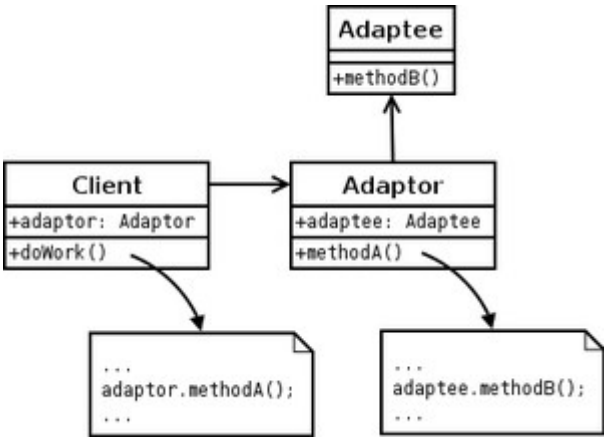
A híd lehetővé teszi az interfész és az implementáció szétválasztását, így a kettő külön változtatható egymástól függetlenül.



A díszítő lehetővé teszi az absztrakció változtatása nélkül további funkciók, felelősségi körök dinamikus hozzáadását.



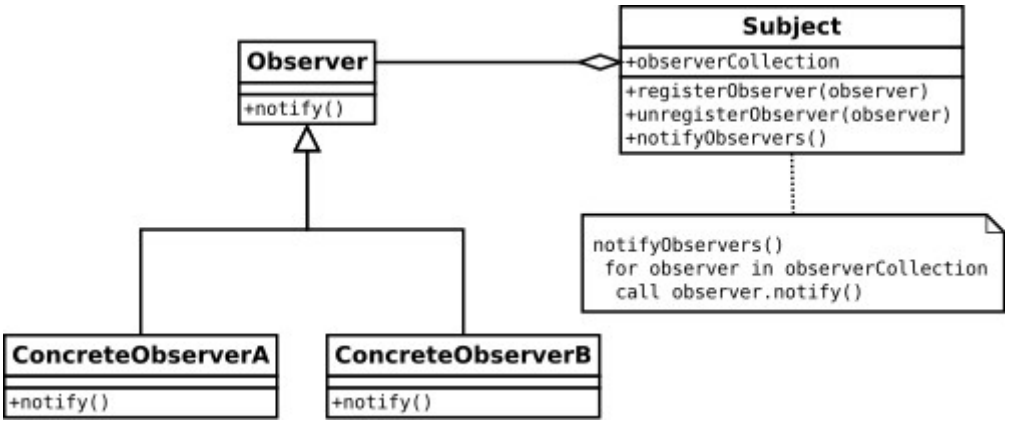
Az illesztő minta egy felület átalakítása, illesztése egy másikra. Összeférhetetlen osztályok együttműködését teszi lehetővé.



A viselkedési minták (Behavioral patterns) elsősorban algoritmusokkal, illetve az osztályok és objektumok közötti kommunikációval, a felelősségi körök kijelölésével foglalkoznak.

Az Iterator lehetővé teszi egy aggregált objektum elemeinek szekvenciális elérését anélkül, hogy láthatóvá tenné a mögöttes reprezentációját.

A Megfigyelő meghatároz ez egy-a-többhöz függőséget objektumok között. Egy adott objektum módosulásáról automatikus értesítő információt küld a tőle függő objektumoknak, amik ezek alapján frissülnek.



A Stratégia egységbe zárt, azonos interfésszel rendelkező algoritmusok dinamikus cserélgetése. Lehetővé teszi, hogy az algoritmus az őt használó kliensektől függetlenül változzon.