Tervezési minták:

A tervezési minták sokszor felmerülő problémák leírásait adják meg úgy, hogy közben megadják a megoldás magját, amit adaptálva a konkrét rendszerhez újra és újra alkalmazhatunk. Ami a kódolásnak az osztály könyvtár, az a tervezésnél a tervezési minták gyűjteménye.

Objektum orientált programok esetében a tervezési minta leírása megadja azokat az egymással kommunikáló objektumokat, osztályokat, amelyek együttes viselkedése az adott problémára megoldás lehet. Persze nemcsak objektum orientált nyelvekre lehet megadni tervezési mintákat, de mi most csak ezen mintákkal fogunk foglalkozni.

Gyakorlatilag a legtöbb jó OO architektúra használ mintákat. De miért is jó mintákat használni? Egyfelől azért, mert a minták által a rendszer egyszerűbb, karbantarthatóbb és újrafelhasználhatóbb lesz, másfelől a minták ismeretével megérteni is könnyebb a rendszert, hiszen a minták a programozói folklór része, ért(het)i az is, aki amúgy az adott kódbázist nem ismeri, de ezáltal egy-egy rész megértése a kódban könnyebbé válik.

Gyártási minták: Factory Method (Virtual Constructor) - Gyártófüggvény: Interfészt definiál egy objektum létrehozásához, de az alosztályokra bízza, melyik osztályt példányosítják. A factory method megengedi az osztályoknak, hogy a példányosítást az alosztályokra ruházzák át.

Abstract Factory (Kit) - Elvont gyár: Kapcsolódó vagy egymástól függő objektumok családjának létrehozására szolgáló interfészt biztosít a konkrét osztályok megadása nélkül.

Builder - Építő: Összetett objektum létrehozását biztosítja az elemeinek létrehozásával és összerakásával.

Prototype - Prototípus: Prototípus példány használatával egy alap objektumot, az újabb objektumok létrehozását pedig ennek a prototípusnak a lemásolásával állítja elő.

Singleton - Egyke: Egy osztályból csak egy példányt engedélyez, és ehhez globális hozzáférést ad.

Szerkezeti minták: Adapter (Wrapper) - Illesztő: Az adott osztály interfészét átalakítja, hogy az kompatibilissé (használhatóvá) váljon más rendszerek részére.

Bridge (Handle, Body) - Híd: Az elvont ábrázolás (absztrakció) és implementáció közötti csatolást lazítja, hogy a kettő egymástól függetlenül legyen változtatható.

Composite - Összetétel: Rész-egész szerkezetek leképezését valósítja meg az objektum-hierarchiára a rész és egész azonos kezelésével.

Decorator (Wrapper) - Díszítő: További felelősségeket/tulajdonságokat csatol dinamikusan az objektumhoz.

Facade - Homlokzat: Egy alrendszerben különböző interfészek egy halmazához egységes interfészt biztosít. A módszerrel magasabb szintű interfészt határozunk meg, amelynek révén az adott alrendszer könnyebben használhatóvá válik.

Flyweight - Pehelysúlyú: Nagy számú apró, megosztott objektum kezelését teszi hatékonnyá. Akkor használatos, ha nagy (nagyságrendileg száz, vagy ezer) számú objektumunk van, amelyek nagyon hasonlóak. Ilyenkor ez a minta csökkentheti a memória felhasználást.

Proxy (Surrogate) - Helyettes: Egy adott objektumot egy helyettesítő objektummal vált fel, amely szabályozza az eredeti objektumhoz történő hozzáférést is. Gyakorlatilag a proxyval egy köztes réteg kerül a kliens és az objektum közé. Így az eredeti objektumhoz való hozzáférést kontrollálhatjuk.

Viselkedési minták: Interpreter - Értelmező: Egy adott nyelv nyelvtanát reprezentálja megadva azt az értelmezőt is, amely értelmezni tudja az adott nyelven leírt mondatokat.

Template Method - Sablonfüggvény: Egy algoritmus vázát definiálja, amelyben bizonyos műveleteket csak absztrakt módon definiál, megvalósításukért csak a származtatott osztályok lesznek felelősek.

Chain of Responsibility - Felelősséglánc: A kérés/parancs küldője és végrehajtója közötti közvetlen csatolást megszünteti úgy, hogy a kérést feltételezhetően kezelni tudó objektumokat láncba állítja, és a kérés addig halad a kiszolgálók láncolatán, amíg azt valamelyik le nem kezeli.

Command (Action, Transaction) - Parancs: Kéréseket objektumként ábrázolja azok sorba állíthatósága, naplózhatósága végett.

Iterator (Cursor) - Iterator: Tároló-objektum elemeinek sorozatos elérését teszi lehetővé a reprezentációtól függetlenül.

Mediator - Közvetítő: Objektumok együttműködési módját változtatja meg egy egységbezáró objektum közbeékelésével, amely által a két objektum közötti eredeti csatolás megszűnik.

Memento (Token) - Emlékeztető: Az egységbe zárás megsértése nélkül kinyeri és elmenti egy objektum belső állapotát, hogy az később ebbe az állapotba visszaállítható legyen.

Observer (Dependents, Publish-Subscribe) - Megfigyelő: Egy objektum állapotát megfigyeli, az objektum állapotának megváltozásáról pedig a tőle függő objektumokat értesíti.

State (Objects for States) - Állapot: Adott objektum számára engedélyezi, hogy belső állapotának megváltozásával megváltoztathassa viselkedését is. Az objektum ekkor látszólag módosítja az osztályát.

Strategy (Policy) - Stratégia: Meghatároz egy algoritmus családot, amelyben a több, azonos feladatú, de különböző algoritmus egységbezárásával azok felcserélhetővé válnak.

Visitor - Látogató: Egy adott objektum-hierarchián végezhető műveletsort reprezentál, a hierarchia bejárásával ezen műveletek elvégezhetőekké válnak a hierarchia elemein annélkül, hogy a a hierarchiát alkotó osztályokat megváltoztatnánk.