# Tervezési minták egy OO programozási nyelvben

## MVC, mint modell-nézet-vezérlő minta és néhány másik tervezési minta

### 1. Bevezetés

A tervezési minták olyan bevált megoldások, amelyek gyakran előforduló problémák kezelésére kínálnak általános módszereket objektumorientált programozásban. Ezek a minták nem konkrét kódok, hanem egyfajta receptet nyújtanak a szoftverstruktúrák kialakításához.

**A tervezési minták jellemzői**

Általánosság: Többféle helyzetben is alkalmazhatóak.

Újrahasznosíthatóság: A minta nem az adott problémára íródott, így más kontextusban is használható.

Olvashatóság: Segíti a fejlesztők közötti kommunikációt, hiszen a minták elnevezései általában közismertek.

**A tervezési minták kategóriái**

A mintákat három fő kategóriába sorolhatjuk:

1. Kreációs minták: Az objektumok létrehozására koncentrálnak, például a Singleton és Factory Method.
2. Szerkezeti minták: Az osztályok és objektumok közötti kapcsolatok kialakítását segítik, például az Adapter vagy a Decorator.
3. Viselkedési minták: Az osztályok és objektumok közötti kommunikációra és interakcióra fókuszálnak, például az Observer és Strategy.

### 2. MVC minta (Model-View-Controller)

Az MVC egy szerkezeti minta, amely az alkalmazások működését három összetevőre bontja:

* Modell: Kezeli az adatok tárolását és üzleti logikát.
* Nézet: A felhasználói felület, amely az adatokat megjeleníti.
* Vezérlő: Fogadja a felhasználói bemeneteket, és azok alapján irányítja az alkalmazás működését.

**Az MVC felépítése és működése**

1. A felhasználó interakcióba lép a nézettel (például egy gombot megnyom).
2. A vezérlő feldolgozza az eseményt, és szükség esetén módosítja a modellt.
3. A modell változásai frissítik a nézetet, amely újra megjeleníti az adatokat.

**MVC példa egy webes alkalmazásban**

1. Modell: Adatbázis-kezelő osztály, amely egy webshop termékeit tartalmazza.
2. Nézet: HTML/CSS oldalak, amelyek a termékeket megjelenítik.
3. Vezérlő: PHP vagy JavaScript kód, amely feldolgozza a vásárlói kereséseket vagy kosárba helyezéseket.

**MVC előnyei**

* Jobb karbantarthatóság: Az egyes rétegek külön kezelhetőek és tesztelhetőek.
* Újrafelhasználhatóság: A nézet vagy a modell módosítása nem befolyásolja a vezérlőt.
* Egységes logika: Az üzleti logika nem keveredik a megjelenítéssel.

**MVC hátrányai**

* Az alkalmazás bonyolultsága nőhet kisebb projekteknél.
* Nagyobb tanulási görbét jelent kezdők számára.

### 3. Singleton minta

A Singleton egy kreációs tervezési minta, amely biztosítja, hogy egy adott osztályból csak egyetlen példány jöhessen létre. Ezt gyakran használják globális erőforrások kezelésére.

**Példák a használatra**

* Konfigurációs osztályok: Egy alkalmazás beállításai csak egyszer kerülnek inicializálásra.
* Naplózás (Logging): Egyetlen naplóobjektum kezeli a rendszer logjait.

**Singleton előnyei**

* Egyértelmű globális hozzáférés biztosítása.
* Csökkenti a memóriahasználatot, mert csak egy példány jön létre.

**Singleton hátrányai**

* Globális állapotot hoz létre, amely nehezíti a tesztelést.
* A párhuzamos programozásnál gondosan kell kezelni a példányosítást.

### 4. Observer minta

Az Observer egy viselkedési minta, amelyben egy objektum (alany) képes értesíteni több más objektumot (megfigyelők) az állapotváltozásairól.

**Observer működése**

1. A megfigyelők regisztrálják magukat az alanynál.
2. Az alany állapotváltozás esetén értesíti a megfigyelőket.

**Példák használatra**

* UI frissítése: Egy alkalmazás több felhasználói felületének szinkronban tartása.
* Eseményvezérelt rendszerek: Például hírlevél-rendszerek.

**Observer előnyei**

* Csökkenti a függőséget az osztályok között.
* Könnyen bővíthető, új megfigyelők hozzáadása egyszerű.

**Observer hátrányai**

* Teljesítményproblémák jelentkezhetnek sok megfigyelő esetén.

### 5. Strategy minta

A Strategy minta lehetővé teszi, hogy egy osztály különböző algoritmusokat használjon, amelyeket futásidőben cserélhet.

**Strategy felépítése**

1. Egy absztrakt osztály vagy interfész meghatározza az algoritmus struktúráját.
2. Több konkrét implementáció biztosítja az egyes stratégiákat.
3. A fő osztály egy konkrét stratégiát használ a futás során.

**Strategy példa egy webáruházban**

Fizetési módok: CreditCardPayment, PayPalPayment, BankTransfer.

Az ügyfél kiválasztja a preferált stratégiát, a rendszer pedig ennek megfelelően jár el.

**Strategy előnyei**

* Könnyen bővíthető, új stratégiák egyszerűen hozzáadhatók.
* Az osztályok és az algoritmusok közötti kapcsolat tiszta és szétválasztott.

**Strategy hátrányai**

* Az osztályok száma jelentősen megnőhet.
* Nem mindig szükséges, ha az algoritmusok nem változnak gyakran.

### 6. Összegzés

A tervezési minták nélkülözhetetlen eszközök az objektumorientált szoftverfejlesztésben.

Az MVC, Singleton, Observer és Strategy minták mind különböző problémákra nyújtanak optimális megoldást.

Ezek a minták elősegítik a kód újrafelhasználhatóságát, karbantarthatóságát és átláthatóságát.

### 7. Források

* Design Patterns: Elements of Reusable Object-Oriented Software
* Refactoring.Guru: <https://refactoring.guru>