Tervezési minták
egy objektumorientált programozási nyelvben

Bevezetés

A programozási minták olyan jól bevált megoldások, amelyek bizonyos problémákra általánosan alkalmazható módszert kínálnak. Ezek a minták segítenek a szoftverfejlesztőknek abban, hogy strukturált, könnyen karbantartható és bővíthető programokat készítsenek. Az objektumorientált (OO) programozás egyik nagy előnye, hogy jól alkalmazhatóak benne ezek a tervezési minták, melyek elősegítik a kód újrafelhasználhatóságát és tiszta szerkezetét.

Ebben a dolgozatban a Model-View-Controller (MVC) mintát mutatom be részletesen, majd ismertetem néhány másik gyakori tervezési mintát, és azok alkalmazási területeit.

MVC (Model-View-Controller) minta

Az MVC egy széles körben használt architekturális tervezési minta, amely elválasztja a program logikai részeit három fő komponensre: a Modellt, a Nézetet és a Vezérlőt.

- **Model** (Modell):
  A program adatainak és üzleti logikájának kezelője. A modell tárolja az adatokat és kezeli a velük kapcsolatos műveleteket, például adatbázis lekérdezéseket vagy üzleti szabályok érvényesítését.

- **View** (Nézet):
  Az adatokat megjelenítő réteg, amely a felhasználó felé vizuális megjelenítést biztosít (pl. grafikus felület, weboldal). Csak megjeleníti az adatokat, nem kezel logikát.

- **Controller** (Vezérlő):
  Kapcsolódik a felhasználói interakciókhoz, és kezeli a felhasználói bemeneteket. A vezérlő értesíti a modellt, ha adatváltoztatás történik, és frissíti a nézetet, hogy a változások megjelenjenek.

Az MVC előnyei:

- Elkülöníti az üzleti logikát a megjelenítéstől, így a fejlesztés és karbantartás egyszerűbb.
- Könnyű bővíteni és módosítani az egyes részeket anélkül, hogy az egész rendszert újra kellene írni.
- Lehetővé teszi a párhuzamos fejlesztést (pl. a grafikus felület fejlesztője és az adatkezelő fejlesztő külön dolgozhat).

Példa: Egy webalkalmazásban a Modell lehet az adatbázisból lekért felhasználói adatok kezelése, a Nézet a weboldal HTML oldala, a Vezérlő pedig a gombnyomás vagy űrlapbeküldés kezelését végzi.

Néhány másik tervezési minta

Singleton

A Singleton minta biztosítja, hogy egy osztályból csak egyetlen példány létezzen az alkalmazás futása alatt, és globális hozzáférést biztosít ehhez a példányhoz.

Mikor használjuk? Olyan esetekben, amikor például egy konfigurációs beállításokat tartalmazó osztályra van szükség, amit több helyen is el kell érni.

Factory Method

Ez a minta egy interfészt vagy absztrakt osztályt definiál a példányosításhoz, de az alosztályokra bízza, hogy melyik konkrét osztály példányát hozzák létre.

Mikor használjuk? Ha egy objektum létrehozását szeretnénk elválasztani a használatától, és többféle változat közül kell választani.

Observer

Az Observer minta egy objektum állapotváltozását több más objektumnak is továbbítja automatikusan. Az értesítést azok kapják, akik „feliratkoztak” az eseményekre.

Mikor használjuk? Például grafikus felületekben, amikor egy adat változása után több komponensnek is frissülnie kell.

Decorator

Ez a minta dinamikusan, futási időben bővíti egy objektum képességeit anélkül, hogy az osztályát módosítanánk.

Mikor használjuk? Ha egy objektum viselkedését szeretnénk rugalmasan kiterjeszteni.

Összegzés

A tervezési minták használata jelentősen megkönnyíti a szoftverfejlesztők munkáját, hiszen bevált megoldásokat nyújtanak gyakori problémákra. Az MVC minta segít a kód szerkezetének tisztán tartásában és az egyes rétegek szétválasztásában, míg más minták, mint a Singleton vagy az Observer, speciális helyzetekben biztosítanak hatékony megoldást.

Az objektumorientált programozási nyelvek pedig kifejezetten alkalmasak arra, hogy ezeket a mintákat implementáljuk, ezáltal könnyebben fenntartható, átláthatóbb és rugalmasabb alkalmazásokat hozzunk létre.

Felhasznált irodalom

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley.
- https://refactoring.guru/design-patterns
- https://www.tutorialspoint.com/design_pattern/index.htm

Készítette:

Dörnyei Dániel
KA2XCB