Tervezési minták a programozásban

A szoftvertechnológia egyik alapköve a tervezési minták alkalmazása. Ezek a minták olyan dokumentált, újrahasznosítható megoldások, amelyek a szoftvertervezés során gyakran ismétlődő problémákra adnak választ. Nem kész kódrészletekről, hanem absztrakt megoldási sablonokról van szó, amelyek elősegítik a kód karbantarthatóságát, tesztelhetőségét, valamint a rugalmasságát. Az alábbi projektem során alkalmazott három meghatározó mintát mutatom be részletesebben.

Első ilyen tervezési minta a **Singleton (egyke).**

A Singleton a létrehozási minták családjába tartozik. Elsődleges célja, hogy az adott osztályból az alkalmazás teljes életciklusa alatt pontosan egyetlen példány jöjjön létre.

Bizonyos rendszerelemeknél (pl. naplózók, konfigurációs modulok vagy adatbázis-kezelők) kritikus fontosságú, hogy ne létezzen több párhuzamos példány, mert azok egymás működését zavarhatnák. A minta megvalósítása során az osztály konstruktora privát elérést kap, így megakadályozva a külső példánymódosítást. Az egyetlen objektumot az osztály egy statikus metódusán keresztül érhetjük el, amely az első híváskor létrehozza a példányt, minden további esetben pedig ugyan azt adja vissza.

Második ilyen tervezési minta a **Strategy (stratégia).**

A Straregy egy viselkedési minta, amely lehetővé teszi, hogy egy objektum viselkedését vagy algoritmusát futásidőben válasszuk meg, anélkül, hogy a kliens kódot módosítani kellene.

A minta alapelve az egységbe zárás. Az algoritmusokat külön osztályokba emeljük ki, amelyek egy közös interfészt valósítanak meg. Ezáltal a szoftver megfelel az Open/Closed elvnek, amellyel nyitott bővítésre, de zárt a módosításra.

Harmadik ilyen tervezési minta a **MVC (Model-View-Controller) architektúra.**

Az MVC egy összetett tervezési minta, amely az alkalmazást három jól elkülöníthető felelőségi körre osztja.

Model (Modell): Az alkalmazás magja. Tartalmazza az adatokat, azok állapotát és az üzleti logikát. Teljesen független.

View (Nézet): Feladata az adatok vizuális megjelenítése a felhasználó számára. Nem hoz döntéseket, csak adatokat közvetít.

Controller (Vezérlő): A híd a két oldal között. Fogadja a felhasználói bemenetet, feldolgozza azt a Modellel, majd utasítja a Nézetet a frissítésre.

Összegezve a fent leírtakat, a tervezési minták alkalmazása biztosítja, hogy a programom ne csupán egy működő szoftver, hanem egy professzionális alapokon, skálázható alkalmazás legyen. A felelőségi körök elválasztása, a példányosítás kontrolálása és az algoritmusok rugalmassága együttesen teszik lehetővé a szoftver jövőbeni továbbfejlesztését.