Tervezési minták OO programozási nyelvekben

A tervezési minták az objektumorientált programozás (OOP) alapvető építőkövei, amelyek segítenek a gyakori problémák megoldásában a szoftverfejlesztés során. Ezek a minták olyan bevált megoldások, amelyeket tapasztalt fejlesztők dolgoztak ki, hogy megkönnyítsék az újrahasznosítható, karbantartható és skálázható kódok írását. A beadandóm során a következő tervezési mintákra fogok kitérni: MVC, MVVM, MVP, Factory és Singleton.

Model-View-Controller azaz MVC tervezési minta:

Az MVC minta az alkalmazások logikáját három fő komponensre bontja: Model, View és Controller. Mindegyik komponensnek meghatározott szerepe van, amelyeket külön kezelve egyszerűbbé válik az alkalmazások fejlesztése és karbantartása. Eredetileg grafikus felhasználói felületekhez (GUI) tervezték, de napjainkban webes és mobilalkalmazások fejlesztésében is széles körben alkalmazzák.

Controller:

A Controller a View és a Model közötti kapcsolatot biztosítja. Fő feladata, hogy feldolgozza a felhasználói interakciókat (például kattintások, adatbevitel), és ezek alapján utasításokat adjon a Modelnek vagy frissítéseket kezdeményezzen a View számára. A Controller maga nem végez adatmanipulációt, hanem meghatározza, hogyan reagáljon az alkalmazás az adott felhasználói műveletekre.

View:

A View a felhasználói felületért felelős komponens, amely a Model által biztosított adatokat jeleníti meg vizuális formában. A View nem közvetlenül kommunikál a Modellel, hanem általában a Controlleren keresztül. Ez a komponens biztosítja, hogy a felhasználó számára mindig a megfelelő információ jelenjen meg.

Model:

A Model az alkalmazás adatainak kezeléséért, tárolásáért és validálásáért felelős réteg. Ez a komponens tartalmazza az üzleti logikát is, amely az adatok manipulálását és a szabályok alkalmazását végzi. Gyakran közvetlen kapcsolatban áll az adatbázisokkal vagy más külső adatforrásokkal.

Singleton:

A Singleton minta biztosítja, hogy egy osztálynak csak egy példánya legyen és globális hozzáférést nyújt ehhez a példányhoz. Ezt a mintát gyakran használják abban az esetekben, amikor egy központi vezérlő objektumra van szükség, például egy konfigurációs beállítások kezelésére vagy egy adatbázis-kapcsolat megosztására.

Factory

A factory minta olyan tervezési minta, amely lehetővé teszi, hogy objektumokat hozzunk létre anélkül, hogy azok konkrét osztályait ismernénk. Ez a minta különösen hasznos, ha az osztályok példányosításának logikája bonyolult vagy esetleg változó.

Model-View-ViewModel (MVVM) minta

A MVVM minta a GUI alkalmazások fejlesztésére szolgál, különösen népszerű a WPF és mobil alkalmazások köreiben. A tervezési minta célja, hogy az adatokat és az üzleti logikát elkülönítse a megjelenítéstől, miközben a két réteg között adat kapcsolatot biztosít.

Model:

Az alkalmazás üzleti logikáját és adatait kezeli, hasonlóan az MVC mintában található “Model”-hez.

View:

A felhasználó számára látható felület, amely adatkapcsolatot használ a ViewModel-hez.

ViewModel:

Közvetítő a View és a Model réteg között, kezeli az adatkapcsolakat, a felhasználói interakciókatt és az üzleti logika hívásait.

Model-View-Presenter (MVP)

Az MVP minta a korábban említett MVC tervezési mintából alakult ki. Az MVP modell főleg olyan alkalmazásokban nyújt jelentős előnyöket az MVC-hez hasonlóan, ahol komplex adathalmazokon kell műveleteket végezni és ezek eredményeit a felhasználó elé tárni. A minta sajátosságából eredően könnyedén alá vethetőek Unit testeknek.

Model:
A model réteg az alkalmazás által megjelenített és feldolgozott adatok reprezentációja.

View:
A view réteg megjeleníti a modelben tárolt adatokat a felhasználó számára, illetve a felhasználói interakció során bekövetkező eseményeket továbbítja a prezenter felé.

Prezenter:
A prezenter összegyűjti az adatokat, illetve formázza a nézet számára feldolgozható módon.