MVC Tervezési Minta

Az **MVC (Model-View-Controller)** minta az alkalmazások kódját három részre osztja: **Model**, **View** és **Controller**. A **Model** felelős az adatokért és a háttérben futó logikáért, például az adatok tárolásáért és kezeléséért. A **View** az adatok megjelenítését és a felhasználói bevitel rögzítését végzi. Ez az a rész, amit a felhasználó lát. A **Controller** közvetít a Model és a View között, feldolgozza a felhasználói parancsokat, és elindítja a megfelelő műveleteket. Ez az elkülönítés megkönnyíti a kód karbantartását és fejlesztését, mivel a különböző részek önállóan módosíthatók.

Az MVC nagy előnye, hogy lehetővé teszi a párhuzamos munkát: egy csapat dolgozhat a View-n, míg egy másik a Modelen, anélkül hogy zavarnák egymást. Emellett könnyebb hibát keresni és tesztelni is. Viszont kisebb projektekben az MVC néha túl bonyolult lehet. A modern fejlesztésben az MVC különböző változatait, például az MVVM-et (Model-View-ViewModel) vagy az MVP-t (Model-View-Presenter) is használják, hogy jobban illeszkedjenek az adott igényekhez.

Bridge Tervezési Minta

A **Bridge minta** segít elválasztani az absztrakciót a megvalósítástól. Ez akkor hasznos, ha a funkciók és a megvalósítások külön fejlődnek. Az absztrakció határozza meg, hogy mit kell csinálni, például egy alakzat rajzolását, míg a megvalósítás azt, hogy hogyan történjen ez meg, például képernyőn vagy nyomtatóval. A kettő egy "híd" interfész segítségével kapcsolódik össze.

Ez a minta különösen akkor előnyös, ha egy alkalmazásnak többféle kombinációt kell támogatnia, például különböző alakzatokat és megjelenítési módokat. A Bridge minta csökkenti a duplikációt és javítja a kód karbantarthatóságát. Ugyanakkor bevezet némi plusz bonyolultságot, ezért inkább nagyobb, összetettebb projektekhez javasolt.

Factory Tervezési Minta

A **Factory minta** olyan tervezési megközelítés, amely egy központi osztályra bízza az objektumok létrehozását. Ahelyett, hogy a klienskód maga példányosítaná az objektumokat (pl. egy Kör vagy Téglalap), egy gyár (pl. ShapeFactory) dönt arról, hogy milyen típusú objektumot hozzon létre. Ez megkönnyíti az új objektumtípusok hozzáadását anélkül, hogy a meglévő kódot módosítani kellene.

A Factory minta különösen hasznos, ha az objektum pontos típusa csak futásidőben derül ki, vagy ha az objektum létrehozása bonyolultabb előkészítést igényel. Ugyanakkor túl sok gyár bevezetése a kód bonyolultságát növelheti. A Factory Method és az Abstract Factory minták ennek továbbfejlesztett változatai, amelyek még specifikusabb igényeket szolgálnak ki, például kapcsolódó objektumok csoportos létrehozását.

https://en.wikipedia.org/wiki/Model–view–controller

https://en.wikipedia.org/wiki/Bridge_pattern

https://en.wikipedia.org/wiki/Factory_method_pattern