Tervezési Minták Dokumentáció – Connect 4

Készítette: Demeter Attila László

Neptun kód: JY133Z

Tárgy: Programozási technológia

Bevezetés

A Connect 4 (Négy a sorban) játék fejlesztése során törekedtem az objektumorientált elvek (OOP) és a tiszta kód (Clean Code) betartására. Bár a projekt mérete kicsi, több tervezési minta (Design Pattern) elve is megjelenik benne, akár közvetlen implementáció, akár architekturális döntés formájában.

Az alábbiakban bemutatom a projektben alkalmazott vagy azzal rokonítható 3 tervezési mintát.

1. MVC (Model-View-Controller) Architekturális Minta

Leírás

Az MVC minta célja, hogy szétválassza az alkalmazás üzleti logikáját (Model), a felhasználói felületet (View) és a vezérlést (Controller). Ez teszi lehetővé, hogy a logika független legyen a megjelenítéstől, így később könnyen cserélhető legyen például a konzolos felület egy grafikusra.

Alkalmazása a projektben

A kódomban határozottan szétválasztottam a felelősségeket:

- Model (Modell): A Connect4 osztály.
  - Ez az osztály felel a játék állapotáért (tábla tömb), a szabályokért (gravitáció, nyerésellenőrzés) és az adatok integritásáért. Nem tartalmaz System.out.println parancsokat (kivéve a debug/logolást), nem kommunikál közvetlenül a felhasználóval.
- View & Controller (Nézet és Vezérlő): A Main osztály.
  - Ez az osztály felel a kapcsolattartásért a felhasználóval (Scanner), az inputok bekéréséért és a tábla kirajzolásáért a konzolra. Ez hívja meg a Modell metódusait a játékos utasításai alapján.

Ez a szétválasztás teszi lehetővé a kód átláthatóságát és tesztelhetőségét.

2. Strategy (Stratégia) Minta

Leírás

A Stratégia minta lehetővé teszi, hogy egy algoritmuscsaládot definiáljunk, és ezeket cserélhetővé tegyük. A kliens a futás során döntheti el, melyik algoritmust (stratégiát) használja.

Alkalmazása a projektben

A projektben ez a minta az AI (Mesterséges Intelligencia) működésében érhető tetten a Connect4 osztály aiLepes() metódusánál.

- Jelenleg egy "Mohó Stratégiát" (Greedy Strategy) alkalmazok: a gép megkeresi az első érvényes oszlopot, és oda lép.
- A tervezésnek köszönhetően ez a logika könnyen kiszervezhető lenne egy külön AiStrategy interfészbe, amelynek lehetnének EasyAi, MediumAi vagy HardAi implementációi. A Connect4 osztálynak nem kellene tudnia, melyik fut, csak meghívná a közös interfész metódusát. A jelenlegi aiLepes() metódus ennek az elvnek az alapköveit fekteti le: a döntéshozatal elszigetelt a játék többi részétől.

3. Singleton (Egyke) Minta

Leírás

A Singleton minta biztosítja, hogy egy osztályból csak egyetlen példány létezzen a program futása során, és ehhez globális hozzáférést biztosít. Gyakran használják erőforrás-kezelésre, konfigurációra vagy naplózásra.

Alkalmazása a projektben

A projektben a Naplózás (Logging) megvalósításánál használtam ezt a mintát.

A Connect4 osztályban található sor:
Java
```
private static final Logger logger =
Logger.getLogger(Connect4.class.getName());
```

A Java beépített java.util.logging.Logger osztálya a Singleton mintát követi. Amikor a getLogger-t meghívjuk, a rendszer nem hoz létre minden alkalommal

új naplózó alrendszert, hanem a már meglévő, központi példányt adja vissza az adott osztályhoz rendelve. Ez biztosítja, hogy a naplófájlok írása (I/O műveletek) kontrolláltan, egy központi helyről történjen, elkerülve az erőforrás-ütközéseket.

Összegzés

A fenti minták alkalmazása segítette a kód karbantarthatóságát. Az MVC elvnek köszönhetően a játékszabályok könnyen tesztelhetők (JUnit), a Singleton alapú naplózás segíti a hibakeresést, a Stratégia elvű AI pedig lehetőséget ad a jövőbeli fejlesztésekre.