**Model-View-Controller (MVC) Tervezési Minta**

1. **Model (Adat és üzleti logika)**
   * A *GameBoard* osztály a játék állapotát reprezentálja (a táblát), és felelős a játékszabályok kezeléséért (például érvényes lépések, győzelem vagy döntetlen ellenőrzése).
   * A *Player* osztály a játékosok adatait tárolja (például név, token).
   * A *Connect4Game* osztály tartalmazza az üzleti logikát, mint a játékállapot frissítése, a játékosok váltása, pontszámok kezelése és fájlkezelési műveletek.
2. **View (Megjelenítés és kimenet)**
   * A nézet felelős a játék megjelenítéséért a felhasználó számára. Ez lehet például egy konzol-kimenet (a *showHighScores()* vagy a *toString()* metódusok alapján), de később akár grafikus felhasználói felület (GUI) is alkalmazható.
3. **Controller (Interakció irányítása)**
   * A vezérlő kezeli a felhasználói inputokat és irányítja a játék működését.
   * A *Connect4Game* osztály bizonyos részei már vezérlőként funkcionálnak (például a játékosok váltása és a játékállapot ellenőrzése).

**Előnyök**

* **Modularitás**: Az üzleti logika, megjelenítés és input kezelése külön rétegekben van, ami megkönnyíti a karbantartást és a bővítést.
* **Tesztelhetőség**: Az egyes komponensek külön-külön tesztelhetők. Például a *GameBoard* és a *Connect4Game* osztályok jól tesztelhetők a meglévő JUnit tesztekkel.
* **Kiterjeszthetőség**: Új nézetek (például GUI vagy webes felület) könnyen hozzáadhatók anélkül, hogy a logikát módosítani kellene.

**Fejlesztési javaslatok**

* Különítsd el explicit módon a vezérlőt (például *GameController* osztály).
* A fájlkezelési és pontszámfrissítési logikát érdemes külön osztályba szervezni, például egy *HighScoreManager*-be.
* Gondolj arra, hogy a játékot később hogyan bővítheted GUI-ra vagy hálózati játékra. Az MVC minta segíthet ennek megvalósításában.