

Tervezési Minták Dokumentáció

Amőba (Gomoku) Java Projekt

Ez a dokumentum a projektben alkalmazott szoftvertervezési mintákat (Design Patterns) és architekturális elveket mutatja be.

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

A projekt legfőbb szervező elve az MVC minta, amely elválasztja az üzleti logikát a felhasználói felülettől és a vezérléstől. Ez teszi a kódot tesztelhetővé és átláthatóvá.

Általános leírás

Az MVC minta három részre osztja az alkalmazást:

- **Model (Modell):** Az adatok és az üzleti logika (játékszabályok). Nem tud a megjelenítésről.
- **View (Nézet):** Az adatok megjelenítése a felhasználó számára.
- **Controller (Vezérlő):** Kezeli a felhasználói bemenetet és frissíti a Modellt.

Implementáció a projektben

- **Model (Üzleti Logika):** * A `hu.unipg.amoba.model` csomag és a `hu.unipg.amoba.game` csomag alkotja.
 - **Osztályok:** Board, Game, Player, Cell, Move, Position.
 - **Szerep:** Itt történik a játékszabályok ellenőrzése (pl. `board.isWinningMove`), a tábla állapotának tárolása és az AI döntéshozatal. Ezek az osztályok nem tartalmazzak `System.out.println` hívásokat (kivéve debug), és nem kezelnek billentyűzet bemenetet.
- **View (Megjelenítés):**
 - A projekt jellegéből adódóan (Parancssoros alkalmazás) a View réteg a `System.out` konzolra írásokban valósul meg.
 - **Implementáció:** A `Main.java` osztályban található kiíró utasítások (pl. `System.out.println(board)`) felelősek a tábla kirajzolásáért. A `Board.toString()` metódus előkészíti a szöveges reprezentációt a View számára.
- **Controller (Vezérlő):**
 - **Osztály:** `hu.unipg.amoba.Main`.

- **Szerep:** A Main osztály main metódusa és a Scanner használata valósítja meg a vezérlést. Itt történik a parancsok (pl. "a5", "save", "quit") beolvasása, értelmezése, és a megfelelő Model metódusok (game.playHumanMove, BoardIO.save) meghívása.

2. Data Access Object (DAO) Minta

A DAO minta célja, hogy elválassza az alkalmazás logikáját az adatbázis-elérési technológiától. Ez lehetővé teszi, hogy az üzleti logika ne függjön az SQL parancsoktól.

Általános leírás

A DAO egy interfészt vagy osztályt biztosít, amely absztrakt műveleteket (mentés, lekérdezés) kínál, elrejtve a háttérben zajló adatbázis-kapcsolat kezelést és SQL lekérdezéseket.

Implementáció a projektben

- **Osztály:** hu.unipg.amoba.io.HighscoreDao
- **Működés:**
 - Ez az osztály felelős a H2 adatbázissal való kommunikációért.
 - A Main osztálynak nem kell tudnia arról, hogy SQL INSERT vagy MERGE parancsok futnak a háttérben. A Main csak meghívja a recordWin(name) metódust.
 - A HighscoreDao kezeli a Connection, Statement és ResultSet objektumokat, valamint a kivételek (SQLException) naplózását, így a főprogram kódja tiszta marad.

3. Value Object (VO) / Értékobjektum Minta

A Value Object olyan objektum, amelynek az azonosságát nem egy egyedi azonosító (ID), hanem az attribútumainak értéke határozza meg. Ezek az objektumok jellemzően immutábilisak (megváltoztathatatlanok).

Általános leírás

A VO-k egyszerű adatstruktúrák, amelyek logikailag összetartozó adatokat (pl. egy koordinátapár) fognak össze. Fontos jellemzőjük, hogy felülírják az equals() és hashCode() metódusokat, hogy az érték szerinti összehasonlítás működjön.

Implementáció a projektben

- **Osztályok:** hu.unipg.amoba.model.Position, hu.unipg.amoba.model.Move
- **Megvalósítás:**

- A projekt Java record típusokat használ (public record Position(int row, int col)).
- A record-ok automatikusan immutábilisak (a mezők final-ok), és a Java automatikusan generálja számukra a helyes equals(), hashCode() és toString() metódusokat.
- Például két Position(1, 2) objektum logikailag egyenlőnek számít a programban, ami elengedhetetlen a győzelmi feltételek ellenőrzésénél.

4. Static Factory Method (Statikus Gyártó Metódus)

Bár egyszerű formában, de a projekt használja a statikus gyártó metódusokat az objektumok létrehozásának olvashatóbbá tételére.

Implementáció a projektben

- **Osztály:** hu.unipg.amoba.model.Position
- **Metódus:** public static Position of(int row, int col)
- **Előnye:** A kód olvashatóbbá válik. Ahelyett, hogy new Position(r, c)-t írnánk mindenhol, a Position.of(r, c) kifejezőbb, és lehetőséget ad későbbi optimalizációra (pl. gyakori pozíciók gyorsítótárazása) anélkül, hogy a hívó kódot módosítani kellene.

Összegzés

A projekt sikeresen alkalmazza a modern Java fejlesztés alapvető mintáit. Az **MVC** biztosítja a kód modularitását, a **DAO** leválasztja az adatkezelést, a **Value Object**-ek pedig biztonságossá és egyszerűvé teszik az adatmozgatást a rétegek között.