

CONNECT FOUR PROGRAMTERV

A **Connect Four játék programtervének felépítése** különböző osztályokba szervezett logikai elemeket tartalmaz, amelyek mindegyike a játék egy adott részét valósítja meg. Ezek az osztályok segítenek a játék teljes folyamatának áttekinthető és hatékony megvalósításában, biztosítva a sikeres futást és a könnyű bővíthetőséget.

FELÉPÍTŐ OSZTÁLYOK:

1. BOARD OSZTÁLY

- Az osztály feladata a játék fő terének, vagyis a táblának a felépítése.
- Tartalmazza a tábla adatstruktúráját, amely a korongok elhelyezésére szolgál.
- Beépítettem a tábla állapotának ellenőrzésére szolgáló logikát, például hogy egy adott oszlop megtelt-e.
- Tartalmazza azokat a funkciókat is, amelyekkel a győzelem feltételeit ellenőrizzük vízszintes, függőleges és átlós irányokban.

2. DISC OSZTÁLY

- Enum osztály, amely a Connect Four játékban használt korongok tulajdonságait definiálja.
- A Disc enum a játékban résztvevő kétféle korongot reprezentálja:
 - Sárga korong (YELLOW), amely a játék egyik játékosához tartozik.
 - Piros korong (RED), amely a másik játékoshoz tartozik.

Mindegyik koronghoz tartozik egy szimbólum (symbol), amely karakterként jeleníti meg a korongot: Y a sárga korong szimbóluma, R a piros korong szimbóluma.

```
3 public enum Disc { 80 usages  Dorina8212
4     YELLOW( symbol: "Y"), 32 usages
5     RED( symbol: "R"); 29 usages
6
7     private final String symbol; 2 usages
8
9     Disc(String symbol) { 4 usages  Dorina8212
10         this.symbol = symbol;
11     }
12
13     @Override  Dorina8212
14     public String toString() {
15         return symbol;
16     }
17 }
```

3. GAMESTATE OSZTÁLY

- A GameState osztály a játék teljes logikáját kezeli, beleértve a lépések végrehajtását, az állapot ellenőrzését, a játékosok váltását, valamint a mentést és betöltést.
 - Ez az osztály biztosítja, hogy a játék gördülékenyen működjön, és könnyen bővíthető legyen, például fejlettebb AI-val vagy új funkciókkal.
-

4. POSITION OSZTÁLY

- Egyszerűen és hatékonyan kezeli a tábla pozícióit, kulcsfontosságú a játék logikájában.
 - A getterek biztosítják az attribútumok lekérdezését.
 - Az equals és hashCode metódusok megkönnyítik a pozíciók összehasonlítását és hash-alapú gyűjteményekben való használatát.
-

5. MAIN OSZTÁLY:

Ez a Main osztály a Connect Four játék futtatására és irányítására szolgál, beleértve a játékosok közötti interakciókat, a játék állapotának kezelését, valamint a statisztikák és mentett játékállapotok kezelését. A következőket végzi:

1. FŐ FUNKCIÓK

- **Új játék indítása vagy betöltése:** A program indításakor a felhasználónak lehetősége van új játékot kezdeni vagy egy mentett játékot betölteni.
 - **Játéklogika futtatása:** A játék körökre osztott logikát követ, ahol minden körben az aktuális játékos lép, majd a program ellenőrzi a nyerő állapotot.
 - **Játékosstatisztikák kezelése:** A győztes játékos eredményeit menti egy fájlba, és lehetőséget nyújt a győzelmi statisztikák megtekintésére.
 - **Mentés és betöltés:** A játékállapot menthető és betölthető, hogy a játék később folytatható legyen.
-

2. RÉSZLETES MAGYARÁZAT

Main metódus

- **Játék indítása:**
 - A program megkérdezi, hogy a felhasználó szeretne-e betölteni egy mentett játékot.
 - Ha van mentés, a felhasználó kiválaszthatja, melyiket tölti be.
 - Ha nincs mentés, vagy a betöltés sikertelen, új játék kezdődik.
 - **Játékmenet:**
 - Egy végtelen ciklusban fut a játékmenet:
 1. **Tábla kirajzolása:** Megjeleníti a tábla aktuális állapotát.
 2. **Játékos lépése:**
 - Ha az aktuális játékos ember, meg kell adnia, melyik oszlopba szeretne lépni.
 - Ha az aktuális játékos AI, véletlenszerű oszlopot választ.
 3. **Korong ejtése:** Az osztály a megfelelő oszlopba ejti a korongot.
 4. **Győzelem ellenőrzése:** Ha az aktuális lépés nyerő lépés, a győztest kihirdeti, és a játék véget ér.
 - **Statisztikák és mentés:**
 - Ha van győztes, a játék megkérdezi, hogy menteni kell-e az állapotot.
 - Megjeleníthető a győzelmi statisztikák táblázata.
-

Főbb segédmetódusok

1. **startNewGame(Scanner scanner):**

- Bekéri a játékosok nevét és azt, hogy emberi játékosok-e.
- Inicializálja a táblát és létrehozza a GameState példányt.

```
private static GameState startNewGame(Scanner scanner) { 4 usages 1 Dorina8212
    System.out.print("Enter name for player 1 (Yellow): ");
    String player1 = scanner.nextLine();
    System.out.print("Is player 1 human? (yes/no): ");
    String isPlayer1Human = scanner.nextLine();
    boolean player1Human = isPlayer1Human.equalsIgnoreCase("yes");

    System.out.print("Enter name for player 2 (Red): ");
    String player2 = scanner.nextLine();
    System.out.print("Is player 2 human? (yes/no): ");
    String isPlayer2Human = scanner.nextLine();
    boolean player2Human = isPlayer2Human.equalsIgnoreCase("yes");

    Board board = new Board( rows: 6, cols: 7);
    return new GameState(board, player1, player2, player1Human, player2Human);
}
```

2. **displayBoard(Disc[][] grid):**

- Kirajzolja a táblát a konzolra.
- Üres pozíció: -, sárga korong: Y, piros korong: R.

3. **updatePlayerStats(String playerName):**

- Növeli a megadott játékos győzelmi számát.
- A statisztikákat egy fájlban (player_stats.txt) tárolja.

4. **loadPlayerStats():**

- Beolvassa a statisztikákat a fájlból és egy térképre (Map) tölti.

5. **savePlayerStats(Map<String, Integer> stats):**

- Elmenti a statisztikákat egy fájlba, hogy a győzelmek megmaradjanak a játékok között.

6. **displayHighScoreTable():**

- Megjeleníti a statisztikai fájl alapján a játékosok győzelmi eredményeit.

3. PÉLDÁK ÉS MŰKÖDÉS

Új játék indítása

- A felhasználó megadja a játékosok nevét és típusát (ember/AI).
- A játék új táblával indul, a sárga játékos kezd.

Mentett játék betöltése

- A program megkeresi a `game_state_` kezdetű fájlokat a mappában.
- A felhasználó kiválaszthatja, melyik állapotot szeretné betölteni.

Győztes kihirdetése

- Amikor egy játékos győz, a program frissíti a statisztikát.
- A győzelmi táblázat megtekinthető a játék végén.

ÖSSZEFOGLALÓ A UNIT TESZTEKRŐL

Az alábbi tesztosztályok az **classes.connectfour** nevű Connect Four játékhoz tartozó fő komponensek funkcionalitásának ellenőrzését végzik. Minden teszt célja az egyes osztályok helyes működésének validálása különféle forgatókönyvekben.

1. BOARDTEST

Az osztály a játéktábla funkcionalitását ellenőrzi, beleértve a korongok ejtését, az oszlop telítettségének kezelését és a nyelési feltételek ellenőrzését.

- **testDropDisc:** Ellenőrzi, hogy egy korong ejtése a megfelelő sorba és oszlopba kerül.

```
@Test
void testDropDisc() {
    Board board = new Board( rows: 6, cols: 7);
    Position pos = board.dropDisc( col: 3, Disc.YELLOW);

    assertEquals( expected: 5, pos.getRow());
    assertEquals( expected: 3, pos.getCol());
    assertEquals(Disc.YELLOW, board.getDisc(pos));
}
```

- **testColumnFull:** Teszteli, hogy egy teljesen tele oszlop nem fogad több korongot, és kivételt dob.
- **testHorizontalWin:** Ellenőrzi a vízszintes nyelési feltételt.
- **testVerticalWin:** Ellenőrzi a függőleges nyelési feltételt.
- **testDiagonalWinPositiveSlope:** Pozitív irányú átlós nyelési feltételt tesztel.
- **testDiagonalWinNegativeSlope:** Negatív irányú átlós nyelési feltételt tesztel.
- **testNoWin:** Bizonyos állásokban ellenőrzi, hogy nincs győztes.

2. DiscTest

Az osztály a **Disc** enum működését teszteli, amely a sárga (YELLOW) és piros (RED) korongokat reprezentálja.

- **testYellowDiscToString** és **testRedDiscToString**: Ellenőrzi a megfelelő szöveges reprezentációt ("Y" és "R").
- **testEnumValues**: Biztosítja, hogy az enum értékei helyesen definiáltak.
- **testEnumValueOf**: Ellenőrzi, hogy az Enum.valueOf() metódus a megfelelő enum értéket adja vissza.

3. GameStateTest

Ez az osztály a játék állapotkezelését és az AI működését teszteli.

- **testInitialState**: Ellenőrzi az inicializált játék állapotát, például az első játékos nevét és korongját.
- **testSwitchPlayer**: Validálja, hogy a játékosváltás helyesen működik.
- **testDropDisc**: Ellenőrzi, hogy a GameState helyesen kezeli a korong ejtését.
- **testIsWinningMove**: Vizsgálja a nyerési feltétel helyességét.
- **testGetAIMove**: Ellenőrzi, hogy az AI nem választ telített oszlopot.
- **testAIPlaysCorrectly**: Validálja, hogy az AI helyes oszlopba ejti a korongot.

4. PositionTest

Az osztály a **Position** osztályt teszteli, amely egy cella helyét (sor és oszlop) reprezentálja.

- **testGetRowAndCol**: Ellenőrzi, hogy a helyes sor és oszlop visszatéríthető.
- **testEquals**: Teszteli, hogy két azonos pozíció egyenlő.
- **testNotEqualsWithNullAndDifferentClass**: Bizonyosodik arról, hogy egy pozíció nem egyenlő null-lal vagy más típusú objektummal.
- **testHashCode**: Ellenőrzi, hogy az azonos pozíciók azonos hash-kódot generálnak.

A **POM FÁJL** a Connect Four Java projektet irányítja, és konfigurálja a projekt építési, tesztelési és kódellenőrzési folyamatait. A Maven pluginok segítségével biztosítja a kód lefordítását, a tesztek futtatását, a kódlefedettség mérését és a kódstílus érvényesítését.

A **CHECKSTYLE** konfigurációs fájl egy egyszerű szabálykészletet tartalmaz a Java kód formázási és dokumentációs követelményeinek ellenőrzésére. A Checkstyle egy eszköz, amely segít biztosítani a kód konzisztenciáját és minőségét az előre meghatározott szabályok alapján.