Connect-4 Játék Tervezési Minták

1. Bevezetés

A Connect-4 játék egy stratégiai társasjáték, amely két játékos részvételével zajlik. A cél az, hogy először hozzon létre egy négyes sorozatot vízszintesen, függőlegesen vagy átlósan. A projekt során különböző tervezési mintákat alkalmaztunk a kód strukturálására és optimalizálására. Ebben a dolgozatban bemutatjuk ezeket a mintákat, azok alkalmazásának előnyeit, valamint részletes kódrészleteket is közlünk.

2. Tervezési Minták

2.1 Singleton Minta

A Singleton minta biztosítja, hogy egy osztálynak csak egy példánya legyen, és globális hozzáférést biztosít ehhez az egyetlen példányhoz. A DatabaseManager osztályban alkalmaztuk ezt a mintát, hogy az adatbáziskapcsolatból csak egy példány létezzen, amelyet a teljes alkalmazás használ.

Előnyök:

- **Erőforrás kezelés**: Egyetlen adatbáziskapcsolat létrehozásával és újrafelhasználásával minimalizáljuk az erőforrások használatát.
- **Karbantarthatóság**: Könnyebb karbantartani egyetlen példányt, mint több különböző példányt.

Kódrészlet:

```
public class DatabaseManager {
    private static DatabaseManager instance;
    private Connection connection;

    private DatabaseManager() {
        // Adatbázis kapcsolat létrehozása
    }

    public static DatabaseManager getInstance() {
        if (instance == null) {
            instance = new DatabaseManager();
        }
        return instance;
        }
    }
```

2.2 Factory Minta

A Factory minta egy olyan tervezési minta, amely lehetővé teszi, hogy az objektumokat egységes interfészen keresztül hozzuk létre, anélkül, hogy meg kellene adnunk a konkrét osztályokat. Ezt a mintát alkalmaztuk a Player objektumok létrehozására, hogy különböző típusú játékosokat hozhassunk létre.

Előnyök:

- **Rugalmasság**: Könnyen bővíthető különböző típusú játékosokkal anélkül, hogy módosítani kellene a kód többi részét.
- **Egységes interfész**: Az objektumok létrehozása egységes interfészen keresztül történik, ami egyszerűsíti a kódot.

Kódrészlet:

```
public class PlayerFactory {
    public static Player createPlayer(String type, String name) {
        if (type.equals("human")) {
            return new HumanPlayer(name);
        } else if (type.equals("bot")) {
            return new BotPlayer(name);
        }
        return null;
    }
}
```

2.3 Strategy Minta

A Strategy minta lehetővé teszi, hogy különböző algoritmusokat, stratégiákat definiáljunk és alkalmazzunk futásidőben. Ezt a mintát használtuk a különböző játékos stratégiák (emberi játékos vs. bot) kezelésére.

Előnyök:

- **Rugalmas és bővíthető**: Új stratégiák könnyen hozzáadhatók anélkül, hogy módosítani kellene a meglévő kódot.
- **Egyszerű tesztelhetőség**: A különböző stratégiák különálló egységként kezelhetők, ami egyszerűsíti a tesztelést.

Kódrészlet:

```
public interface PlayerStrategy {
```

```
    void makeMove(GameBoard board);
}

public class HumanStrategy implements PlayerStrategy {
    public void makeMove(GameBoard board) {
        // Emberi játékos lépés logikája
    }
}

public class BotStrategy implements PlayerStrategy {
    public void makeMove(GameBoard board) {
        // Bot lépés logikája
    }
}
```

3. Részletes Kód Példák

Ebben a részben részletesen bemutatjuk a fenti tervezési minták alkalmazását a projektben.

3.1 Singleton Minta

A Singleton minta alkalmazásával biztosítjuk, hogy az DatabaseManager osztályból csak egy példány létezzen, és ez az egyetlen példány kezelje az adatbáziskapcsolatot.

```
public class DatabaseManager {
    private static DatabaseManager instance;
    private Connection connection;

    private DatabaseManager() {
        // Adatbázis kapcsolat létrehozása
    }

    public static DatabaseManager getInstance() {
        if (instance == null) {
            instance = new DatabaseManager();
        }
        return instance;
    }

}
```

3.2 Factory Minta

A Factory minta segítségével különböző típusú játékosokat hozunk létre anélkül, hogy meg kellene adnunk a konkrét osztályokat.

```
public class PlayerFactory {
    public static Player createPlayer(String type, String name) {
        if (type.equals("human")) {
            return new HumanPlayer(name);
        } else if (type.equals("bot")) {
            return new BotPlayer(name);
        }
        return null;
    }
}
```

3.3 Strategy Minta

A Strategy minta lehetővé teszi különböző játékos stratégiák definiálását és alkalmazását futásidőben.

```
public interface PlayerStrategy {
    void makeMove(GameBoard board);
}


public class HumanStrategy implements PlayerStrategy {
    public void makeMove(GameBoard board) {
        // Emberi játékos lépés logikája
    }
}


public class BotStrategy implements PlayerStrategy {
    public void makeMove(GameBoard board) {
        // Bot lépés logikája
    }
}
```

4. Következtetés

A tervezési minták alkalmazása jelentősen javította a Connect-4 játék projekt kódjának olvashatóságát, karbantarthatóságát és bővíthetőségét. A Singleton minta biztosította az erőforrások hatékony kezelését, a Factory minta rugalmasságot nyújtott az objektumok létrehozásában, míg a Strategy minta lehetővé tette a különböző játékos stratégiák könnyű kezelését. Ezek a minták hozzájárultak a projekt stabilitásához és skálázhatóságához, és lehetővé tették a kód egyszerűbb tesztelését és karbantartását.