Németh Máté (VE1TSQ)

Programtervezési minta – Connect Four

File-ok főbb funkció

app.java:

Ez a program belépési pontja. Kezeli a játékosok nevét, a tábla és a játékállapot kezelését, valamint a felhasználóval való interakciókat. Támogatja a játékmentést és betöltést, valamint a játékosok eredményeinek lekérdezését az adatbázisból.

board.java:

A játéktábla adatait és logikáját kezeli. Tartalmazza a győzelem ellenőrzésére szolgáló funkciókat, a táblára helyezett elemek kezelését, valamint a tábla állapotának megjelenítését és betöltését.

DatabaseHelper.java:

Az adatbázis-kezelést végzi. Kapcsolódik egy SQLite adatbázishoz, és támogatja a játékosok beszúrását, győzelmeik frissítését, valamint az aktuális állapotuk lekérdezését.

GameState.java:

A játék állapotát tárolja, beleértve a tábla aktuális állapotát, a soron következő játékost, és az eddig megtett lépéseket.

Programtervezési minta

Model-View-Controller (MVC), elkülöníti az adatkezelést, a logikát és a megjelenítést.

Model:

Az adatok és a játéklogika kezelésére:

Board: A játék állapotát kezeli (pl. táblaelemek, győzelem ellenőrzése).

GameState: Tárolja a játék állapotát, így lehetővé teszi a mentést/betöltést.

DatabaseHelper: Adatbázis-műveletek, például játékosok és győzelmek kezelése.

View:

A felhasználói interakciók kezelésére:

A jelenlegi kód esetében a konzol alapú interakció tölti be ezt a szerepet. Ezt a konzolos kódrészletet elkülönítheted egy dedikált View osztályba a tisztább szerkezet érdekében.

Controller:

A logika és az adatkezelés vezérlésére:

app: A program fő vezérlője. Koordinálja a játékosok közötti váltásokat, kezeli a játékmentést és -betöltést, és irányítja az adatbázis-műveleteket.

Egyéb tervezési minták

Factory Method (Gyártó metódus):

Lehetővé teszi, hogy az objektumok létrehozását alosztályokra bízzuk, ahelyett, hogy közvetlenül az osztályban végeznénk el. Ez a minta egy absztrakt osztályban deklarál egy metódust az objektumok létrehozására, amelyet a konkrét alosztályok implementálnak. Az így létrehozott megoldás rugalmasságot biztosít, mivel új típusú objektumok könnyen hozzáadhatók a rendszerhez a meglévő kód módosítása nélkül. Tipikusan akkor használják, ha az osztályok pontos típusa előre nem ismert, vagy ha a példányosítás bonyolult logikát igényel. Ez elősegíti a nyitott/zárt elv alkalmazását az objektumorientált programozásban.

(Létrehozási m.)

Bridge (híd):

Az absztrakciót elválasztja annak implementációjától, lehetővé téve, hogy mindkettőt függetlenül lehessen módosítani. A minta két hierarchiát hoz létre: az egyik az absztrakciót, a másik az implementációt képviseli. Az absztrakció egy interfészen keresztül delegálja a műveleteket az implementációnak, így a különböző implementációs változatok könnyen cserélhetők anélkül, hogy az absztrakció logikáját meg kellene változtatni. Ezt a mintát akkor használják, ha egy osztályt több dimenzióban is ki kell terjeszteni, és el akarjuk kerülni a komplex öröklési hierarchiákat. (Szerkezeti m.)

Command (parancs):

A műveleteket objektumokként kezeli, így lehetővé teszi a műveletek paraméterezését, sorosítását, naplózását vagy visszavonását. A minta elválasztja a kérést (amit végre kell hajtani) a végrehajtótól azáltal, hogy egy parancsosztályban tárolja a műveletet és annak összes szükséges információját. Ez a minta három fő elemet foglal magában: a *Command*-ot (a parancsot reprezentáló osztály), az *Invoker*-t (amely kezdeményezi a parancsot), és a *Receiver*-t (a művelet végrehajtója). A Command mintát olyan helyzetekben használják, ahol rugalmas vezérlést és bővíthetőséget kell biztosítani a műveletek végrehajtása felett.

(viselkedési m.)