Mintákra és komponensekre alapozott szoftvertervezés: gyakorlati projekt

Készítette: Nagy Boróka

Specifikáció:

Az általam megvalósított desktop alkalmazás fő célja a kínai nyelv elsajátításának elősegítése, illetve a már elsajátított tudás begyakorlása különböző játékok segítségével. A felhasználó 3 különböző játék segítségével mérheti fel és teheti próbára tudását, melyek kitérnek a kínai szavak, jelentésük és kínai karakterrel való megfeleltetésükön át a mondatokon belüli szavak helyes sorrendjére is.

Az alkalmazás megnyitása után a felhasználó egy kínai falut ábrázoló képpel találkozik, melynek közepén 3 gomb helyezkedik el egymás alatt. Ezen gombok mindegyike egy bizonyos játékot ábrázoló és megvalósító JPanelt jelenít meg.

A 3 játék rövid összefoglalója:

- **Találtassuk a kínai karaktert az angol megfelelőjével:** ebben a játékban egy 4x4-es GridLayouton belül különböző angol szavak, illetve kínai karakterek találhatók, ezeket kell párba állítani. Ha egy gombra rákattint a felhasználó, annak színe sárga lesz, és amint egy második gombra is rákattint, a két gomb ha pár, akkor mindkettő zöldre vált és többet nem lesznek kattinthatóak, ha pedig nem párok, akkor 2 másodpercig piros színűek lesznek, majd visszaváltanak az eredeti állapotukra. Ha a felhasználó minden párt megtalál, akkor ennek megfelelő üzenetet kap és a játék folytatódik egy következő körben, ahol ismét 8 random karakter-jelentés lesz megjelenítve. Innen a felhasználónak lehetősége bármikor új játékot kezdeni (a pontszáma nem veszik el, de új pálya generálódik), illetve vissza tud menni a főoldalra az erre kijelölt gomb segítségével.
- **Tegyük egy mondat szavait megfelelő sorrendbe:** ebben a játékban adott egy mondat angolul, illetve pinyin (kínai karakterek romanizált formája) segítségével, illetve a kínai karakterek, melyekre szükség van a mondatban, gombokon elhelyezve, randomizált sorrendben. A felhasználónak jó sorrendbe kell tennie a karaktereket, hogy ugyanazt jelentsék, mint az adott mondatok. Ha egy gombra rákattint, akkor a gomb törlődik a többi gomb mellől és a rajta levő karakter hozzá lesz fűzve a Your sentence résznél található dinamikusan változó JLabel szövegéhez. Lehetőség van Undo és Reset műveletekre, az Undoval az előző műveletet lehet visszafordítani, a Reset-tel pedig a Your sentence mondat törlődik és az összes gomb megjelenik a panel tetején. Itt is egy Score label segítségével láthatjuk, hogy hány mondatot tudtuk sikeresen sorba tenni. A panel alján egy New Game és Back to Main gombok is találhatók.

- **Memóriajáték képekkel:** A harmadik és egyben utolsó játék egy klasszikus memóriajáték, melynél képeket kell társítani a nekik megfelelő karakterekkel, mindezt úgy, hogy emlékezni kell a párok helyére is. A felhasználó minden találgatás után pop-up üzenet formájában tudja meg, hogy sikeresen tippelt-e: ha igen, akkor a két gomb setEnabled(false) állapotba kerül, ha pedig helytelenül, akkor mindkettő visszakerül az eredeti formájára. Itt is van New Game és Back to Main gomb.

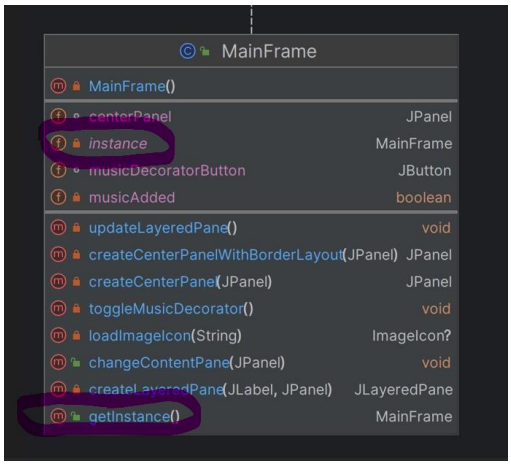
Az alkalmazás utolsó fő funkcionalitása a háttérzene beállításának opciója, amikor is egy gomb segítségével a felhasználó ki- és/vagy bekapcsolhatja a háttérzenét tetszése szerint.

Az alkalmazásom teljes UML diagramja:

(1. Ábra: Teljes UML diagram)

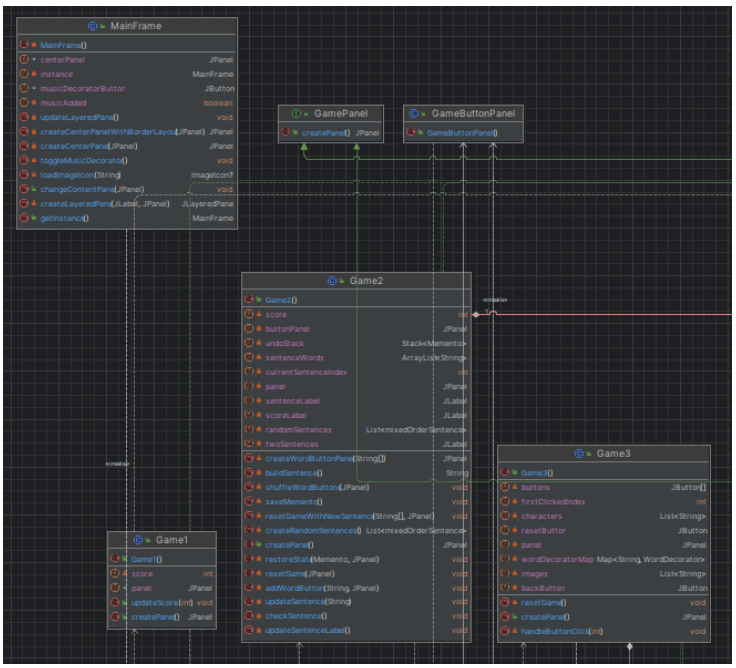
Az alkalmazás implementálásánál több design patternt is használtam, ezek a következők:

1. **Singleton:**
   A lehetővé teszi azt, hogy egy osztályból csak egyetlen példány jöjjön létre egy JVM-en belül. Mivel az applikációm egyetlen egy JFrame keretein belül van megjelenítve és ezen lesznek megjelenítve a különböző játékoknak megfelelő JPanelek, ezért a MainFrame osztálynál használtam a Singleton mintát, hogy biztosra le legyen kezelve az, hogy egyetlen játékkörnyezet lesz csak létrehozva. Az erre vonatkozó UML részlet:



(2. Ábra: Singleton)

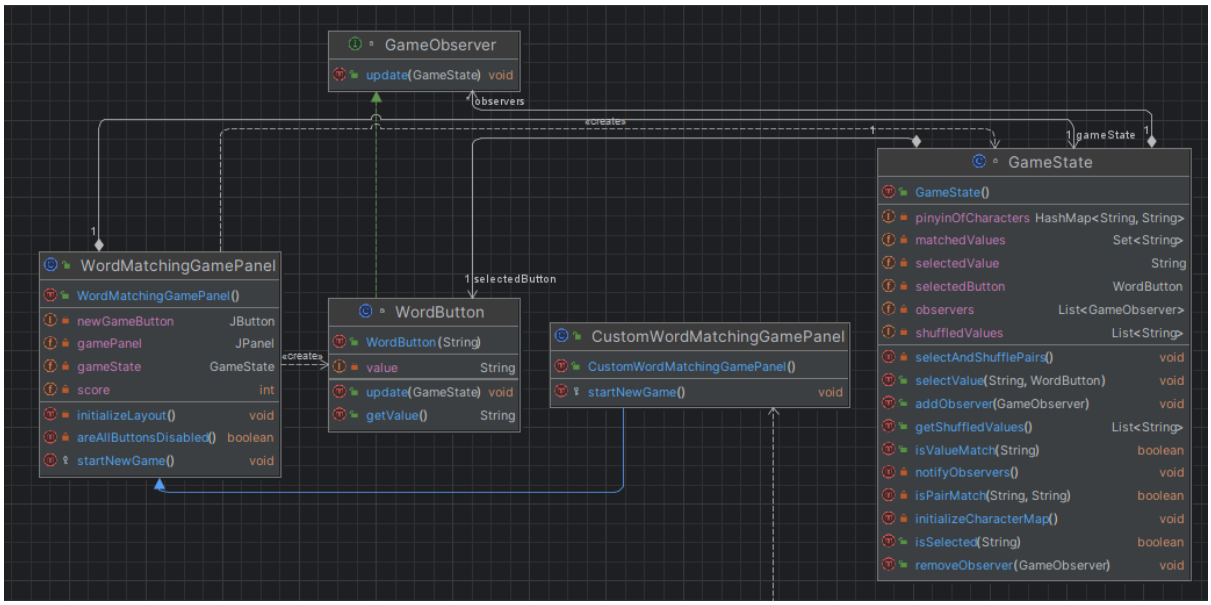
2. **Factory Pattern:**
   Mivel a az alkalmazásban 3 különböző játék van, melyek mind egy-egy különböző elemekkel és funkcionalitással rendelkező JPanelt hoznak létre, így ésszerűnek tűnt a Factory Pattern használata. Létrehoztam egy GamePanel nevű interfacet, melybe egy createPanel() nevű metódus került. Mivel mind a 3 játék ezt az interfacet implementálja, a játékpanelek létrehozása és visszatérítése egységesen a felülírt metódussal valósul meg.



(3. Ábra: Factory Pattern)

3. Observer:

Az Observer tervezési minta használata ebben az esetben azért hasznos, mert lehetővé teszi a szópár kiválasztó játék (Game1) állapotának és változásainak figyelését és értesítését az érdeklődő objektumoknak (többi gomb). Ebben az esetben a GameObserver interface az értesítendő objektumok közös interfészét definiálja, és a GameState osztály azon objektumokat tartalmazza, amelyek figyelik a játék állapotának változásait.



(4. Ábra: Observer)

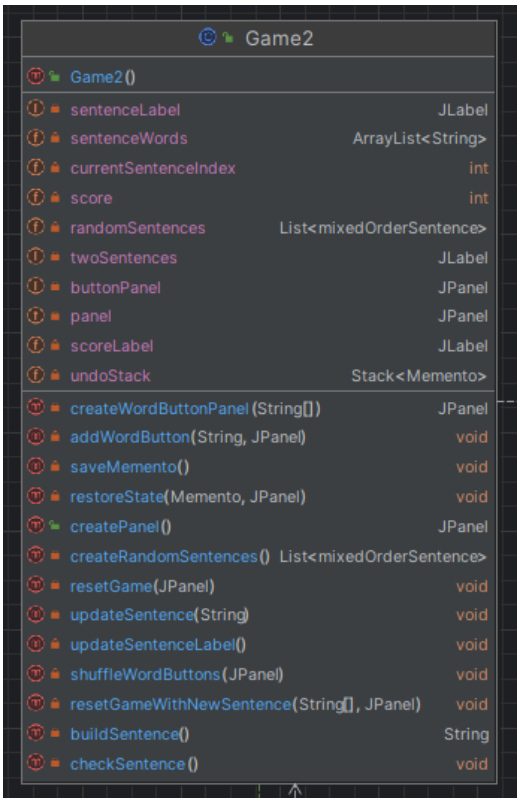
4. Model-View-Controller (MVC)

A következő minta nem tartozik a Gang Of Four tervezési mintái közé, ezért ez egy különlegesebb mintának számít. Az MVC minta elkülöníti az alkalmazás három fő komponensét: Model (modell), View (nézet) és Controller (vezérlő). A Model felelős az alkalmazás állapotának és logikájának tárolásáért. A View felelős az adatok megjelenítéséért és a felhasználói interfészért. A Controller felelős a felhasználói interakciók kezeléséért, és irányítja a Model és a View közötti kapcsolatot. Az én alkalmazásom esetén a Model a GameState, ebben az osztályban tárolódnak az alkalmazás állapotai, logikája, például a kiválasztott értékek és a párok státusza. A View a WordMatchingGamePanel, CustomWordMatchingGamePanel és WordButton osztályokban van, ezek felelősek a játék megjelenítéséért, az UI kezeléséért és a felhasználói interakciókért. A Controller szerepet jelen esetben explicit nem tölti be semmi, de az applikációm esetében a WordButton látja el a Controller feladatait, mivel itt vannak a gombnyomások kezelve és az események továbbítva. Ebben az esetben részlegesen lett implementálva az MVC minta, mert az eseménykezelés és a felhasználói interakciók közvetlenül kapcsolódnak a View és Model rétegekhez. Ehhez a mintához is az Observer mintánál feltűntetett diagramrészlet releváns.

5. Memento:

Memento minta egy olyan tervezési minta, amely lehetővé teszi egy objektum állapotának külső tárolását anélkül, hogy a külső objektum tudna a belső részletekről. Az én esetemben Memento belső osztály felelős a játék állapotának tárolásáért és visszaállításáért. A konstruktorban megkapja az aktuális játékállapotot, majd létrehoz egy másolatot belőle, a getState metódus visszaadja az állapot másolatát. A Game2 osztályban található undoStack változó egy Stack objektum, amely Memento

példányokat tartalmazza. Amikor az undo működés aktiválódik, a kód a jelenlegi játékállapotot elmenti egy Memento objektumban, majd azt a undoStack-en tárolja el. Amikor az undo folyamat végrehajtódik, a kód az undoStack-ből kiveszi az utolsó Memento objektumot, és visszaállítja vele a játék állapotát.



(5. Ábra: Memento)

6. Decorator:

Az utolsó általam alkalmazott design pattern a Decorator minta. A Decorator minta célja, hogy dinamikusan bővítse egy objektum funkcionalitását anélkül, hogy megváltoztatná annak alapstruktúráját. Lehetővé teszi a funkciók hozzáadását és eltávolítását az egyes objektumokhoz. Az én kódomban két helyen is megjelenik a Decorator pattern:

a. GameDecorator interfész:

-Definiál egy decoratePanel metódust. A metódus egy JPanel objektumot és egy fájl elérési útvonalát kapja paraméterként.

b. MusicDecorator osztály:

-Implementálja a GameDecorator interfészt.

-A háttérzenét kezeli a addBackgroundMusic, stopBackgroundMusic, restartBackgroundMusic, és stopMusic metódusokkal.

-A decoratePanel metódusban hozzáadja a háttérzenét a JPanel-hez.

c. ButtonDecorator interfész:

-A dekorátoroknak az JButton objektum dekorálására szolgáló decorate metódust kell implementálniuk. A throws IOException rész azt jelzi, hogy a dekorátoroknak kezelniük kell az

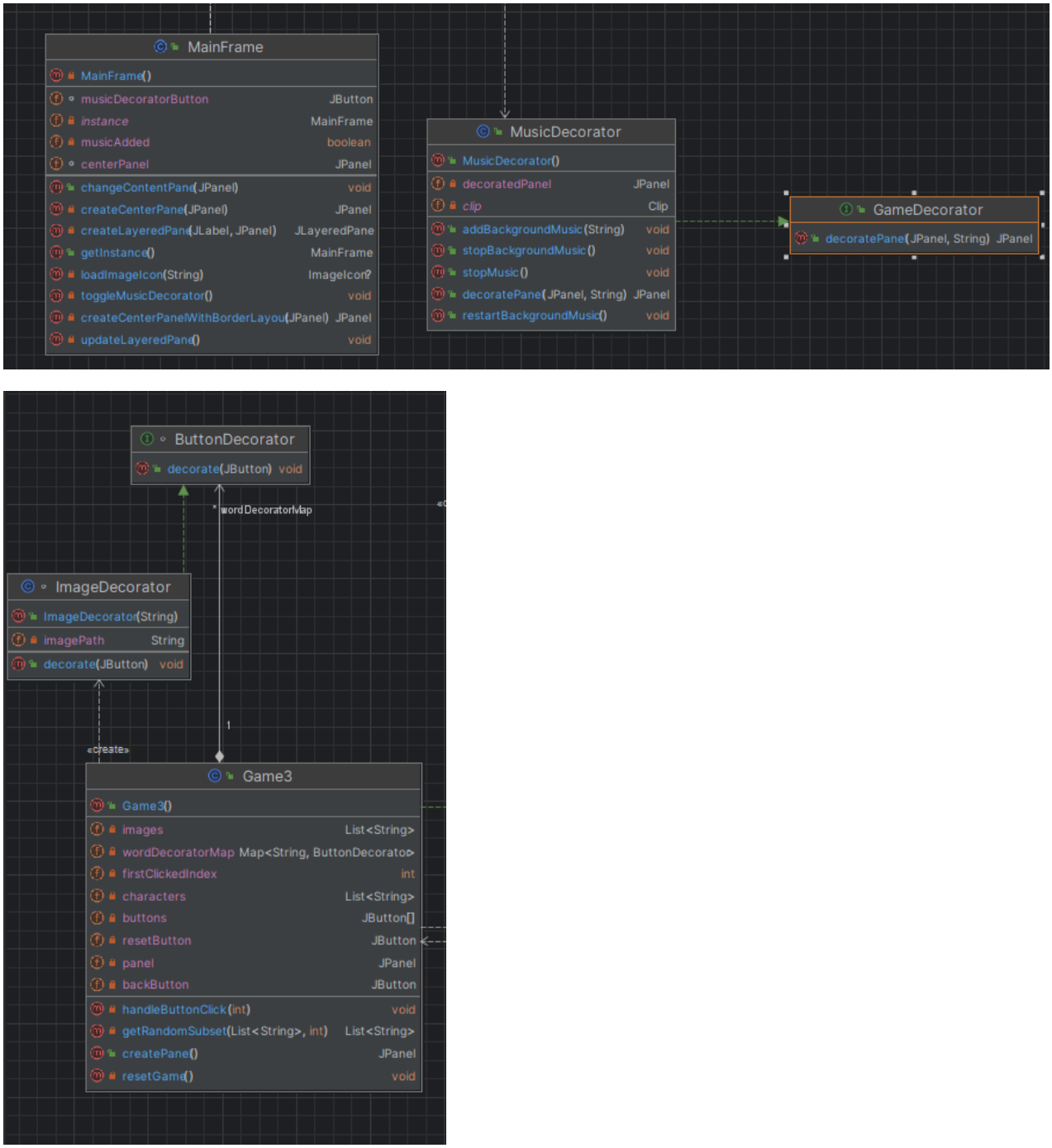
IOException kivételt, amely akkor dobódik, ha valami probléma merül fel az objektum dekorálása során, például egy fájl beolvasása közben.

d. ImageDecorator osztály:

-Implementálja a ButtonDecorator interfészt.

-A decorate metódusban egy JButton objektumot dekorál egy képpel.

-A kép a megadott elérési útvonalon található fájlból kerül beolvasásra.



(6. Ábra: Decorator)