Programozási Technológia

II. beadandó feladat

Perczel-Szabó Dániel
GQF4SF

1. Feladat

Készítsünk programot, amellyel egy Rubik órát lehet kirakni. A játékban 9 óra kap helyet, amely 1-12 közötti értéket mutatnak (kezdetben véletlenszerűen beállítva), és 3×3-as formában jelennek a játéktáblán. Az órák között az átlóknál 4 gomb helyezkedik el, amelyek a szomszédos 4 óra állását tudják eggyel növelni (tehát 4 óra van, amit csak egy gomb növel, 4, amit kettő, és 1, amit mind a négy gomb növel). A kezdő állásban az órák véletlenszerű időt mutatnak. A játék célja az, hogy a gombokkal történő állítással mind a 9 óra 12-t mutasson. A program biztosítson lehetőséget új játék kezdésére, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány lépéssel (állítással) győzött a játékos, majd kezdjen automatikusan új játékot.

2. Terv

2.1. Feladat elemzése

A feladat megvalósítása során az alábbiakra lesz szükség:

- 5x5 cellából álló játéktábla
- Cellák típusának megfelelő beállítása (3x3 szám, közöttük átlósan 4 gomb)
- Új játék gomb
- Gomb megnyomására a megfelelő cellák értékének változtatása
    - Cellák értékének ellenőrzése

2.2 Típusok

2.2.1 RubikOraModel

A `RubikOraModel` osztály a játék üzleti logikájáért felel. Adattagjai a privát `board` (`int[][]`), mely a játéktáblát, és a `steps` (`int`), ami a lépések számát reprezentálja. A konstruktor a paraméterében megadott n egész szám alapján hozza létre az n×n méretű táblát. Ebben az osztályban definiáljuk az `initializeBoard()` függvényt, amely a tábla elemeit feltölti 1 és 12 közötti random számokkal. Itt található az egész szám visszatérési értékű `getClockValue(int,int)` függvény, amely visszaadja a paraméterben megadott oszlopú és sorú cellában található értéket. Az `updateClock(int,int)` metódus a paraméterben megadott oszlopú és sorú, valamint a tőle jobbra, lefelé, és jobbra lefelé található cellák értékének 12-vel való osztási maradékát növeli eggyel. A logikai típusú `isGameWon()` függvény ellenőrzi, hogy a tábla minden eleme egyenlő-e 12-vel, és eszerinti értékkel tér vissza. Végül az egész szám típusú `getSteps()` metódus a lépések számával tér vissza.

2.2.2 RubikOraView

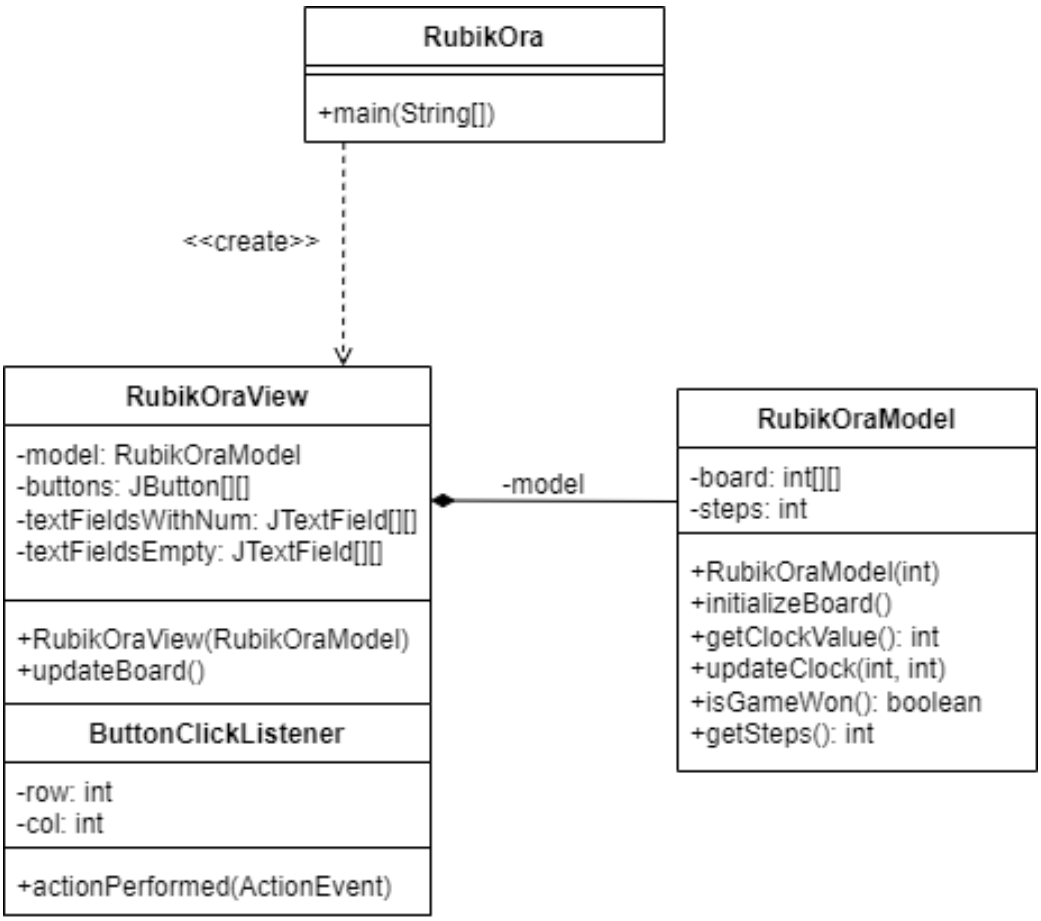
A `RubikOraView` osztály a játék kezelőfelületéért felelős. Privát adattagjai a `model` (`RubikOraModel`), a gombokat reprezentáló `buttons` (`JButton[][]`) kétdimenziós tömb, a számmezőket jelölő `textFieldsWithNum`(`JTextField[][]`) és az üres mezőkért felelős `textFieldsEmpty` (`JTextfield[][]`) kétdimenziós tömbök. A konstruktorban létrehozzuk a játékpanelt és feltöltjük 1-12 közötti random értéket tartalmazó szöveges mezőkkel, gombokkal, és közöttük üres mezőkkel. Itt kap helyet az új játékot létrehozó gomb is. Az `updateBoard()` függvény egy beágyazott ciklussal végigmegy a játéktáblán és beállítja az egyes mezőknek a `model` adattag `board`-jában lévő értékeket.

Az osztály konstruktorában létrehozott gombokhoz hozzárendeltük a `ButtonClickListener` eseménykezelő osztályt, amely implementálja az `ActionListener` interfészt. Adattagjai a privát `row` (`int`) és `col` (`int`), melyek a gomb pozícióját (sor és oszlop) reprezentálják. Az `actionPerform()` metódus meghívja a `model` `updateClock` függvényét, ezzel növelve a megfelelő számmezők értékét, majd frissíti a táblát az `updateBoard` függvénnyel. Megvizsgáljuk továbbá, hogy véget ért-e a játék, és ebben az esetben egy külön ablakban megjelenítjük a szükséges lépések számát.

2.2.3 RubikOra

Az osztály főfüggvényében példányosítunk egy `RubikOraModel` objektumot, majd ezt felhasználva egy `RubikOraView`-t.

2.3 Osztálydiagram



3. Tesztesetek

Leírás	**Elvárt viselkedés**
Játék elindítása	Megjelenik a felület a megfelelő tartalmú cellákkal
Gombra kattintás	Gomb körüli cellák értéke eggyel növekszik
Minden cella értéke 12	Megjelenik a felugró ablak a szükséges lépések számával
Megoldás után a párbeszédablak jóváhagyása	Új játék indul, új számokkal
Új játék gomb megnyomása	Új játék indul, új számokkal