Oláh Norbert
PST8RA

Programozás Technológia
2. Beadandó

Feladat:

Készítsünk programot, amellyel egy Rubik táblát lehet kirakni.
A Rubik tábla lényegében a Rubik-kocka két dimenziós változata. A játékban egy $n \times n$ mezőből álló táblán $n$ különböző színű mező lehet, mindegyik színből pontosan $n$ darab, kezdetben véletlenszerűen elhelyezve. A játék célja az egyes sorok, illetve oszlopok mozgatásával (ciklikus tologatásával, azaz ami a tábla egyik végén lecsúszik, az ellentétes
végén megjelenik) egyszínűvé alakítani vagy a sorokat, vagy az oszlopokat (azaz vízszintesen,
vagy függőlegesen csíkokat kialakítani).
A program biztosítson lehetőséget új játék kezdésére a táblaméret (és így a színek számának)
megadásával (2×2, 4×4, 6×6), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hány
lépéssel győzött a játékos, majd kezdjen automatikusan új játékot.

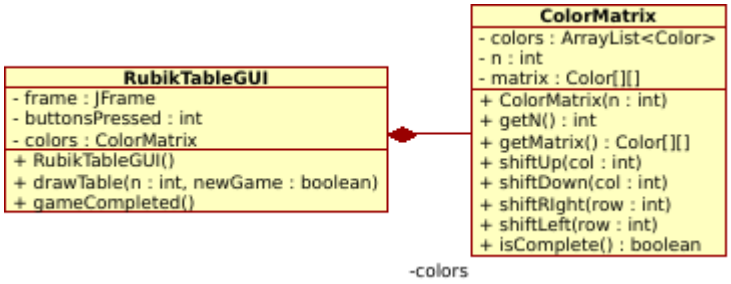
A megoldás menete:
A program belépési helyén létrehozunk egy új RubikTableGUI-t. A konstruktor létrehozza a JFrame-t, majd belerakja a menübárt, a menübárba pedig egy lenyíló listát, ami a táblákat tartalmazza, illetve egy kilépés gombot. Beállítja még a kilépési opciókat, kilépéskor a JFrame bezáródik és a program visszatér. Kilépni az x-re kattintva vagy a menübárban a kilépésre kattintva lehet.

A táblák (2x2, 4x4, 6x6) gombjaihoz illeszt a konstruktor 1-1 eseménykezelőt, amik kattintáskor legenerálják az adott táblát, vagy ha már van aktív játék, akkor újragenerálják. A táblák generálása úgy történik, hogy a drawTable metódus legenerálja az adott táblát, generál egy ColorMátrixot hozzá (ez lesz a kódbeli reprezentációja a táblának), majd az összes sor és oszlop két végére generál egy gombot. Ezekkel a gombokkal lehet a tábla elemeit csúsztatni négyféle irányban.

Ezekre a gombokhoz a generálás közben a metódus illeszt egy eseménykezelőt, ami ha a gomb lenyomódik, meghívja a ColorMátrix megfelelő csúsztató függvényét, majd az így keletkezett ColorMátrix alapján újrarajzolja a táblát.

Minden újrarajzolás után lefut a ColorMátrix isComplete metódusa, ami megvizsgálja, hogy a tábla ki lett-e rakva. Ha a metódus igazzal tér vissza, akkor lefut a gameCompleted metódus, ami közli a felhasználóval, hogy nyert, azt, hogy hány lépéssel, majd vár 4 másodpercet (ez alatt az idő alatt a játékos nem tud interaktálni semmivel), majd automatikusan indít egy új játékot.

A megoldáshoz használt osztályok osztálydiagramja a következő:




Események, eseménykezelők:

A menübárban lévő lenyíló lista elemei (táblakiválasztás):

A 3 gombhoz 3 ActionListener van illesztve, amik a gombok lenyomásakor a drawTable(n, newGame) függvényt hívják meg a megfelelő n paraméterrel (nxn-es tábla), és true-val.
A drawTable függvény legenerálja a táblát és a játékhoz szükséges gombokat.

A menübárban lévő kilépésgomb:

A kilépésgombhoz egy ActionListener van illesztve, ami a gomb lenyomásakor a System.exit(0) metódussal leállítja a programot.

A tábla elemeinek csúsztatásához használt gombok:

A tábla méretének függvényében n*4 gombhoz van illesztve n*4 ActionListener, amik a gombok megnyomásakor hozzáadnak a buttonsPressed változóhoz egyet, majd a gomb helyzetének függvényében meghívják a megfelelő táblát csúsztató metódust, majd meghívják a drawTable függvényt a megfelelő n paraméterrel és false-al.

Tesztelés
1: A program indulásakor egy ablak megnyílik
2: Játék menü lenyílik
3: Új játék menü lenyílik
4: 2x2-es tábla működik
5: 4x4-es tábla működik
6: 6x6-os tábla működik
7_1: Felfele csúsztatás működik
7_2: Lefele csúsztatás működik
7_3: Balra csúsztatás működik
7_4: Jobbra csúsztatás működik
8: Játék vége működik