**ProgTech // 2. Beadandó // Dokumentáció**

**Tatai Áron Péter // G07ZOE**

**9. feladat**

Készítsünk programot, amellyel egy Rubik táblát lehet kirakni. A Rubik tábla lényegében a Rubik-kocka két dimenziós változata. A játékban egy 𝑛×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.

Közös követelmények:

• A megvalósításnak felhasználóbarátnak és könnyen kezelhetőnek kell lennie. Törekedni kell az objektumorientált megoldásra, de nem kötelező a többrétegű architektúra alkalmazása

• A megjelenítéshez lehet vezérlőket használni, vagy elemi grafikát. Egyes feladatoknál különböző méretű játéktábla létrehozását kell megvalósítani, ekkor ügyelni kell arra, hogy az ablakméret mindig alkalmazkodjon a játéktábla méretéhez.

• A dokumentációnak tartalmaznia kell a feladat elemzését,

a program szerkezetének leírását (UML osztálydiagrammal),

valamint az esemény-eseménykezelő párosításokat

és a tevékenység rövid leírását.

# Osztálydiagram

Graphical user interface

Description automatically generated

# Feladat Elemzése.

egy n×n-es játéktáblához egy (n+1)\*(n+1)-es táblát generálunk, a szélső helyeken gombok kapnak helyet, amikkel az éppen aktuális sor/oszlopot fel vagy le lehet mozgatni. Erre készült a Button.Type enum, ami azonosítja őket. a COLOR a rubik tábla elemeit jelenti, az EMPTY pedig a tábla széleit.

Játék menet:

* Játéktér lerenderlődik, a model::shuffle() megkeveri a mezőket
* Amíg ( nincs az összes szín sorba vagy oszlopba rendezve. Ezt a model::isFinished metódusa ellenörzi )
  + Várunk a felhasználó kattintására. Egy gomb kattintás megmondja, hogy melyik sort/oszlop ID-t kell mozgatni +-1 -el. Ezt a model::shift metódusa kezeli le.
  + Minden lépés után a model::steps nő eggyel.
* Ha nyert a user, akkor kiírjuk, hogy hány lépésben nyert, és lehetőséget adunk neki új játék kezdésésre.
* A játék alatt a user bármikor kiléphet, vagy a tábla méret átállítás után új játékot indíthat.

# Esemény-Eseménykezelő párosítások

Új játék kezdése Action 🡪 Menüben beállított mező méret lekérése, majd új model példányosítása, majd a grid újrarenderelése és a színek kirakása

Ablak Bezárása Action 🡪 Kilépés előtt ’OK’ és ’Nem’gomb megjeleníése.

Minden egyes gomb mouseClicked listener 🡪

* Csak akkor kattintható a gomb (nem disabled) ha ez egy iránygomb.
* Lekérjük a koordinátáit, majd a modelben és a view-ban is frissítjuk az új pozíciót.
* Ha így ki lett rakva a kocka, akkor egy Confirm dialógiust mutatunk, ahol új játékot lehet kezdeni