NK: C9TRD0
László Botond

Második beadanó

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, felhasználói eseteit (UML felhasználói esetek diagrammal), a program szerkezetének leírását (UML osztálydiagrammal), valamint az esemény-eseménykezelő párosításokat és a tevékenység rövid leírását.

Feladat : Lovagi torna

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani. Adott egy n×n mezőből álló tábla, amelynek a négy sarkába 2-2 fehér, illetve fekete ló figurát helyezünk el (az azonos színűek ellentétes sarokban kezdenek). A játékosok felváltva lépnek, a figurák L alakban tudnak mozogni a játéktáblán. Kezdetben a teljes játéktábla szürke színű, de minden egyes lépés után az adott mező felveszi a rá lépő figura színét (bármilyen színű volt előtte). A játék célja, hogy valamely játékosnak függőlegesen, vízszintesen, vagy átlósan egymás mellett 4 ugyanolyan színű mezője legyen. A játéknak akkor van vége, ha minden mező kapott valamilyen színt. A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (4×4, 6×6, 8×8), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd automatikusan kezdjen új játékot.

Megoldás

JoustGame osztály:

Első lépésben létrehozunk egy 8xn8 -as gombokból(ez akár azonnal is modosítható a felhasználó által) álló ablakot init() metódussal, mely a játék újraindítása szempontjából fontos, a négy sarokba lehelyezünk a játékosokat(startNewGame() metódus).

Minden ilyen gombhoz hozzáadunk egy ButtonHandler() -t , ez a ButtonHandler a felelős azért, hogy megvizsgálja melyik színű játékos következik illetve meghívódik menne még pár fügvény , isCorrectPosition() megvizsgálja helyes e a pozíció ahove le szeretnénk tenni a lovat, checkingWinners() , a ButtonHandler() végén mindig megvizsgálja van e nyertes, ha van felugró ablak jelzi és kiírja ki nyert, az ok gomb megnyomása után a játék ujraindul 8x8 -as méretben.

A játék végén töröljük a frame-ről a gombokat, majd az init() metódussal újra „rajzoljuk”.

Paths osztály :

Ez az osztály adja vissza a JoustGame osztályában a ButtonHandler()-nek a megfelelő adattagokat, úgy mint a nyertes színe, helyes e pozíció ahova lépni szeretnénk.

Az osztály tartalmaz egy lista típusú relativePaths adattagot, amely segítségével eldöntjük L alakú az új koordináta .

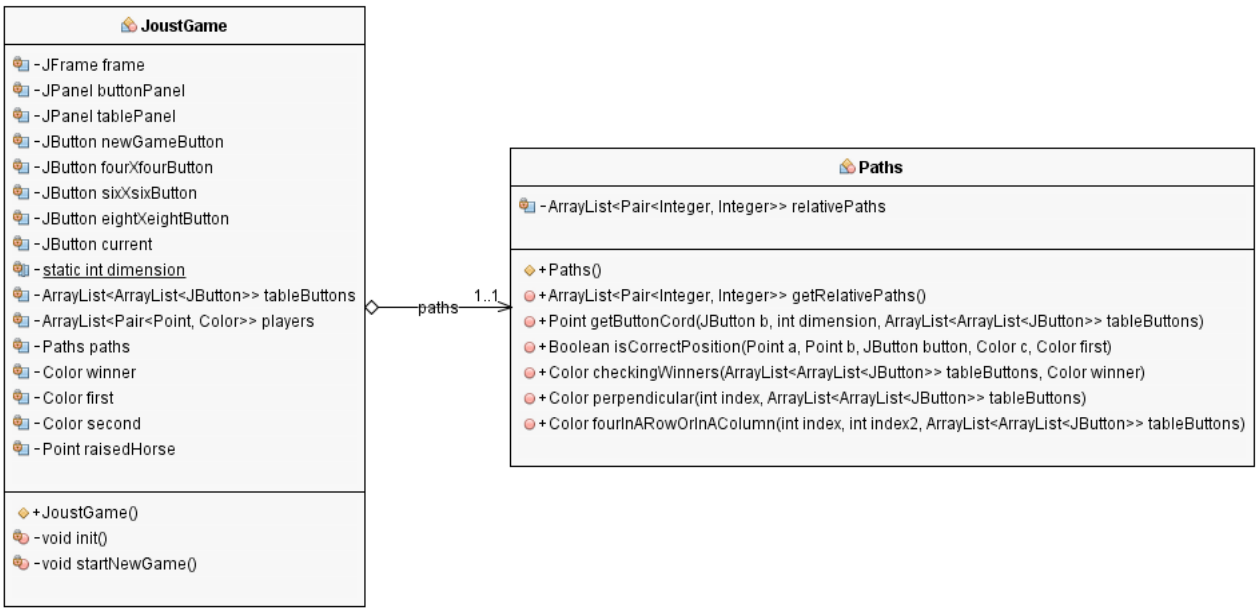
Különböző nyerési esetek:

-Sorban helyezkedik el négy egyforma szín, ezt a fourInARowOrInAColumn() ellenörzi a Paths osztáyon belül

-Oszloposan helyezkedik el egymás alatt négy szín, melyet szintén a fourInARowOrInAColumn() ellenöriz

Kettő kettő egymás alatt, ezt a perpendicular() metódussal ellenörzi, mindegyikük egy szín típussal tér vissza

UML



Tesztek :

Játékban nyer :
8x8
Fekete nyer , sorosan, oszloposan,egymás alatt
Fehér nyer , sorosan, oszloposan,egymás alatt
6x6
Fekete nyer , sorosan, oszloposan,egymás alatt
Fehér nyer , sorosan, oszloposan,egymás alatt
4x4
Fekete nyer , sorosan, oszloposan,egymás alatt
Fehér nyer , sorosan, oszloposan,egymás alatt

Megjegyzés : 4x4-es esetben egyáltalán nem biztos hogy lehet nyerni sorosan vagy oszloposan, a teszt elvégzésekor engedélyeztem a bábuknak a lépést sorrendre, L alaktól függetlenül

Újraindul a játék

8x8
6x6
4x4