Csák Attila Gergő

s7b01e

Feladat:

Készítsünk programot, amellyel a következő két személyes játékot lehet játszani.
Adott egy n × n mezőből álló tábla, ahol egy menekülő és egy támadó játékos helyezkedik el. Kezdetben a menekülő játékos figurája középen van, míg a támadó figurái a négy sarokban helyezkednek el. A játékosok felváltva lépnek. A figurák vízszintesen, illetve függőlegesen mozoghatnak 1-1 mezőt, de egymásra nem léphetnek.
A támadó játékos célja, hogy adott lépésszámon (4n) belül bekerítse a menekülő figurát, azaz a menekülő ne tudjon lépni.
A program biztosítson lehetőséget új játék kezdésére a táblaméret (3×3, 5×5, 7×7) és így a lépésszám (12, 20, 28) megadásával, folyamatosan jelenítse meg a lépések számát, és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, melyik játékos győzött, majd kezdjen automatikusan új játékot.

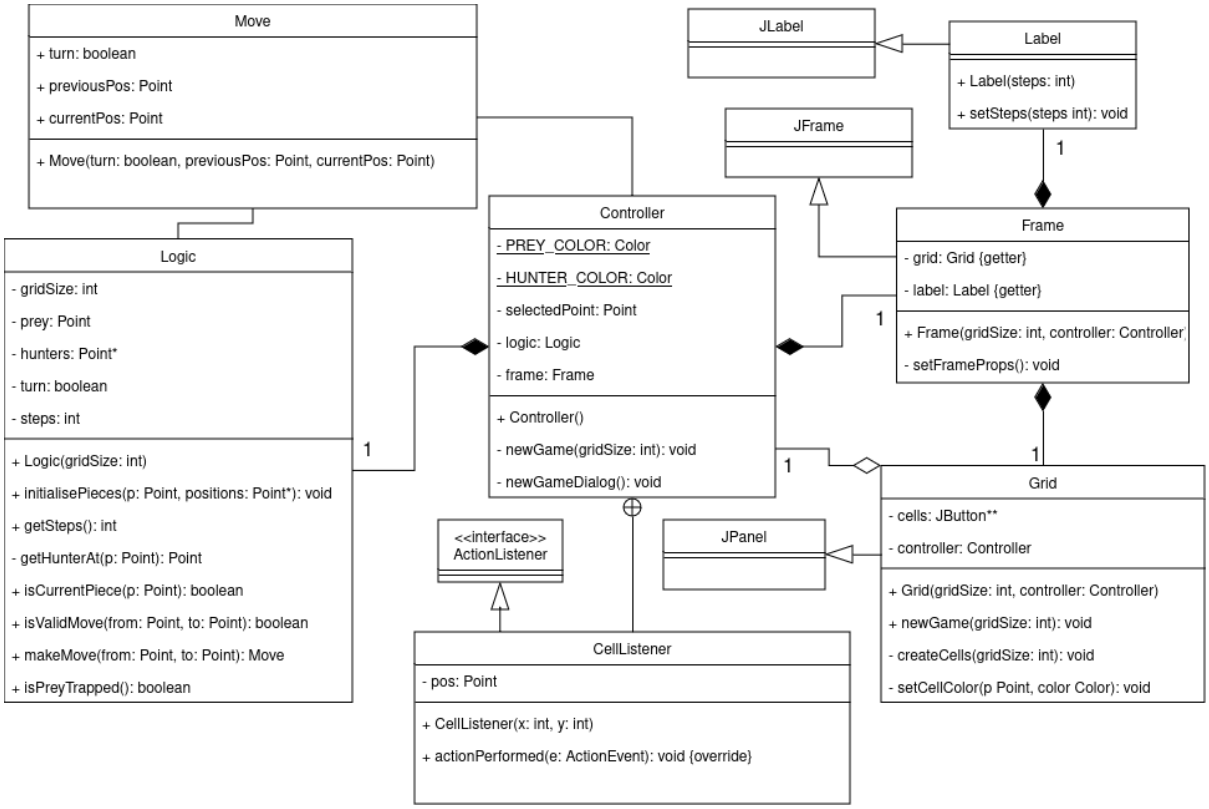
Megoldási Terv

A feladatban leírt játék során a játékosok a pálya celláira kattintva lépnek interakcióba a játékkal. Az első ösztönöm az volt, hogy minden cella a saját állapotát kezelje, és a menekülő figurát tartalmazó cella legyen felelős annak megállapításáért, hogy körbe van-e véve vagy sem. Azonban ezesetben minden cellának tisztában kéne lennie a többi cella állapotával is, nem csak akkor mikor azt akarjuk ellenőrizni hogy a játék véget ért-e, hanem minden egyes játékos lépésnél is. Ahhoz, hogy egy lépést végrehajtsunk a táblán, tudnunk kell, hogy melyik bábu akar lépni, és hova szeretne menni. Az egyik lehetséges megoldás az lett volna, hogy valamiféle interakciót hozunk létre a cellák között, ahol az egyik cella megkérdezheti a másikat, hogy elérhető-e a lépéshez, de ez továbbra is egy központi osztályt igényelt volna, ami nyomon követi, hogy melyik cella melyikkel kommunikáljon.

Végül a Model-View-Controller (MVC) mintát választottam, mivel ez lehetővé tette számomra, hogy a felhasználói felületet és a játékmenet logikáját elkülönítsem egy vezérlőosztály közbeiktatásával. Így a játék logikája nem kell hogy tudjon a GUI-ról, vagy interakcióba lépni azzal, és fordítva. A Controller kommunikálna mind a logikával, mind a grafikus felülettel, de csak közvetítőként funkcionálna a kettő között, a felhasználói felületet a játék logikájában lévő állapot alapján frissítve.

Megvalósítás

A tervek szerint létrehoztam egy Controller, egy Logic és egy Frame osztályt az MVC mintának megfelelően. A Logic osztály felelős a játék állapotának tárolásáért és az új állapot meghatározásáért, amikor a játékos lépést tesz. A Controller a felhasználói bemenetekre reagál, értesíti a Logic-ot a játékosok által végrehajtott akciókról, majd lekérdezi az új állapotot a Logic-tól, és frissíti a felhasználói felületet ennek megfelelően. A Frame és annak komponensei alkotják a játék azon részét, amely a játékosok számára látható.



Amint az osztály diagramon látható, csak a Logic osztály tárol ténylegesen bármilyen információt a játék állapotáról. A Frame, Grid és Label UI osztályok, amelyek mindegyike egy-egy Java Swing UI komponens osztályból származik, kizárólag a program felhasználói felületének elemeit tartalmazza. Azért választottam szét őket így, mert a Grid celláit és a Labelt el kell érni, mivel ezeknek a tartalmát vagy kinézetét frissíteni kell, míg a Frame a program fő felhasználói felület tárolója.

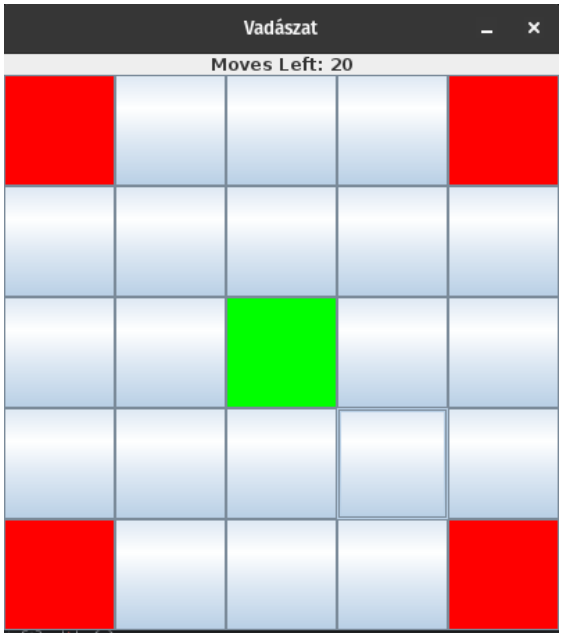
Érdekes lehet a CellListener, a Controller osztályon belüli beágyazott osztály. Ezt azért hoztam létre, hogy kezelje felhasználói kattintásokat; ehhez megvalósítja az ActionListener interfészt. Belső osztályként implementáltam, így használhatja a controller adattagjait és metódusait, miközben minden cellához külön példány jön létre. Minden példány tisztában van a saját cellájának pozíciójával. Ez az osztály tartalmazza a fő kódot, amely összeköti a felhasználót a játék logikájával. Kattintásokra az felülírt actionPerformed metódus fut le, ami továbbítja a játékosok akcióit a Controllernek a Logic példányának, majd frissíti a GUI-t a Logic példány új állapota szerint.

A Logic osztályon belül a makeMove a fő függvény, amely egy kiindulási pontot és egy érkező pontot kap meg paraméterűl, majd kiszámolja és beállítja az új állapotát és visszaad egy Move objektumot, amely tárolja a végrehajtott frissítés adatait. A Move osztályt csupán azért hoztam létre, hogy csoportosítsa azokat az adatokat, amelyekre a Controllernek szüksége van a felhasználói felület állapotának megfelelő frissítéséhez.

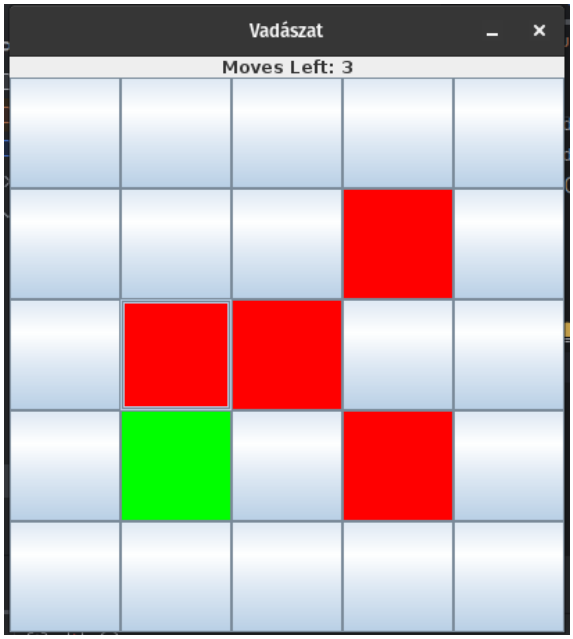
Amikor a program érzékeli a játék végét, egy dialógusablakban megjeleníti, hogy melyik játékos nyert. Ezután felkéri a felhasználót, hogy válasszon egy tábla méretet egy új játékhoz. Ilyenkor a játékosok kiléphetnek a programból a Cancel gomb megnyomásával, vagy kezdhetnek egy új játékot egy általuk választott méretű pályán.

Képek a futó programról:

Kezdeti állapot:



Lépések után:



Játék végét jelző dialógusablak:

Új pályaméret kiválasztása:

Tesztelési terv

A felhasználó limitált módon tud csak interaktálni a játékkal, ezért igazi “edge-case”-ekből nincsen sok, viszont érdemes ellenőrizni a játék megfelelő működését.
A következőket kellett tesztelni: hogy a játék felismeri-e és megfelelően reagál-e a játék végére, illetve hogy a program hogyan reagál különböző lépésekre. A felhasználók interakcióiból nem szabad hibáknak születniük; ha a játékos érvénytelen lépést próbál tenni, azt a játék egyszerűen figyelmen kívül hagyja. Különböző végjáték-konfigurációkat kell tesztelni (például amikor a menekülőt 4 vadász veszi körül, vagy amikor egy sarokba szorul, és csak 2 vadász blokkolja), valamint különböző kattintássorozatokat, amelyeket egy játékos végrehajthat (például egy játékos figurára kattintás majd ugyanarra a cellára újra kattintás, vagy egy figura mozgatásának megkísérlése, amikor nem az adott játékos következik).

User Stories

AS A	Játékos

I WANT TO	A figurámat mozgatni	
1	GIVEN	Én következek és a saját figurámra kattintottam
	WHEN	A választott figura melletti (nem átlós) üres mezőre kattintok
	THEN	A választott figura az új mezőre lép, és most a másik játékos jön.
2	GIVEN	Én következek és a saját figurámra kattintottam
	WHEN	Olyan mezőre kattintok, ami nem a választott figurával szomszédos
	THEN	A mezők nem változnak és továbbra is az én köröm van.
3	GIVEN	Én következek és a saját figurámra kattintottam
	WHEN	Olyan mezőre kattintok, amin már van egy figura
	THEN	A mezők nem változnak és továbbá az én köröm van.
4	GIVEN	Nem én következek és a saját figurámra kattintottam
	WHEN	A választott figura melletti (nem átlós) üres mezőre kattintok
	THEN	A mezők nem változnak és továbbá az én köröm van.
5	GIVEN	Én következek és a saját figurámra kattintottam
	WHEN	A választott figurára kattintok még egyszer
	THEN	A mezők nem változnak és továbbá az én köröm van.
AS A	Vadász játékos	
I WANT TO	Bekeríteni a menekülő figurát	
6	GIVEN	Az én köröm van
	WHEN	A lépésemmel be lesz kerítve a menekülő figura a figuráimmal mind a 4 oldalról
	THEN	Egy dialógusablak jelzi, hogy megnyertem a játékot
7	GIVEN	Az én köröm van
	WHEN	A lépésemmel a menekülő figura be lesz szorítva a pálya egyik sarkába 2 figurám által.
	THEN	Egy dialógusablak jelzi, hogy megnyertem a játékot
8	GIVEN	Az én köröm van, és csak 1 lépésem van hátra

	WHEN	A lépésem után a menekülő figura nem fog tudni lépni
	THEN	A lépés visszaszámláló lemegy 0-ra, de egy dialógusablakban ugyanúgy jelzi a játék, hogy nyertem.
9	GIVEN	Az én köröm van, és csak egy lépésem van hátra a lépés visszaszámláló szerint
	WHEN	Teszek egy olyan lépést, amitől a menekülő játékos utána tovább tudjon lépni
	THEN	Egy dialógusablak jelzi, hogy a menekülő játékos nyerte a játékot
AS A	Játékos	
I WANT TO	Játszani a játékkal	
10	GIVEN	-
	WHEN	Elindítom a programot vagy új játékot kezdek
	THEN	A lépés visszaszámláló helyes értékkel inicializálód.
11	GIVEN	A vadász játékos köre van
	WHEN	A vadász egy érvényes lépést tesz
	THEN	A lépés visszaszámláló csökken egyel.
12	GIVEN	A vadász játékos köre van
	WHEN	A vadász egy érvénytelen lépést próbál tenni
	THEN	A lépés visszaszámláló értéke nem változik
13	GIVEN	A menekülő játékos köre van
	WHEN	A vadász egy lépést tesz vagy próbál tenni
	THEN	A lépés visszaszámláló értéke nem változik
14	GIVEN	A játéknak vége és megjelent a játék újrakezdéséhez használható ablak
	WHEN	Becsukom az ablakot vagy a “Cancel” gombra nyomok
	THEN	Kilépek a programból
15	GIVEN	A játéknak vége és megjelent a játék újrakezdéséhez használható ablak
	WHEN	Kiválasztok egy pálya méretet és az “OK” gombra kattintok
	THEN	A játék újraindul egy új, helyes méretű pályával