Programozás alapjai 3. – Házi Feladat dokumentáció

Szabó Egon Róbert

DEQGWW

1. Game of Life

A Game of Life, magyarul életjáték egy sejtautomata, amit John Horton Conway matematikus talált ki. A klasszikus életjátékot általában egy négyzetrácson valósítják meg, de ebben a programban ehelyett egy hexagonális háló a játék színtere. Minden cellában egy sejt élhet, és ezeket a cellákat 6 szomszédos cella veszi körül. Az eredeti szabály szerint, ha egy sejtnek két vagy három szomszédja van, akkor túléli a következő generációig, ha kettőnél kevesebb, vagy pedig háromnál több szomszédja van, akkor pedig elpusztul. Emellett minden üres cellában új sejt születik, melynek szomszédságában pontosan 3 élő sejt van. A program arra is lehetőséget nyújt, hogy ezt a szabályt szabadon megváltoztathassuk.

A játék menete tehát a következő: A játékos megadhat egy szabályt ami a sejtek születésére és elpusztulására vonatkozik, és egy alakzatot, amely a szimuláció kezdőpontjaként szolgál, majd a többi lépést a számítógép végzi. Ezután a játékos hátra dőlhet és megfigyelheti a játéktáblán kirajzolódó érdekes alakzatokat. Emiatt ezt tulajdonképpen egy nulla személyes játéknak is lehet nevezni.

2. Osztályok leírása

2.1 Cell

A sejteket megvalósító osztály.

Attribútumok

-co: Coord	A sejt koordinátái.

Metódusok

+Cell(c: Coord)	Konstruktor a koordináta alapján.
+Cell(x: int, y: int, z: int)	Konstruktor a koordináták értékei alapján.
+getCoord(): Coord	Visszatér a koordinátával.
+equals(o: Object): boolean	Egyenlőség függvény fölüldefiniálása. Akkor egyenlő két sejt, ha a koordinátájuk megegyezik.
+hashCode(): int	Saját hash függvény a cellák hashmapben való tárolásához.

2.2 Coord

Mivel hatszögekből álló rácson valósítjuk meg a játékot, ezért 3 koordinátára van szükségünk. Megvalósítja a Serializable interfészt.

Attribútumok

-posx: int	Az x koordináta értéke.
-posY: int	Az y koordináta értéke.
-posZ: int	A z koordináta értéke.

Metódusok

+Coord(x: int, y: int, z: int)	Konstruktor, paraméterei a koordináták értékei.
+getPosX(): int	Az x koordináta értékével tér vissza.
+getPosY(): int	Az y koordináta értékével tér vissza.
+getPosZ(): int	A z koordináta értékével tér vissza.
+setPosX(x: int)	Beállítja az x koordináta értékét.
+setPosY(y: int)	Beállítja az y koordináta értékét.
+setPosZ(z: int)	Beállítja a z koordináta értékét.
+areNeighbours(c: Coord): boolean	Ha a koordináta szomszédos a paraméterként kapott koordinátával, akkor igazzal tér vissza, egyébként hamissal.
+getNeighbours(): Coord[]	Visszatér a koordinátával szomszédos koordinátákkal.
+equals(o: Object): boolean	Egyenlőség függvény fölüldefiniálása. Akkor egyenlő két koordináta, ha posX, posY és posZ értéke megegyezik.
+hashCode(): int	Saját hash függvény a koordináták hashmapben való tárolásához.

2.3 EvolutionRule

A szabály, amely meghatározza, hogy egy cella mikor születik meg és mikor éli túl a következő generációig.

Attribútumok

-rules: boolean[][]	A szabályokat tároló kétdimenziós tömb.

Metódusok

+EvolutionRule(born: String, survive: String)	Konstruktor, paraméterei két String, amelyek a születésre és a túlélésre vonatkozó szabályokat tartalmazzák.
+canBeBorn(neighbours: int): boolean	Igazzal tér vissza, ha a paraméterként kapott szám kielégíti a születésre vonatkozó feltételt.
+canStayAlive(neighbours: int): boolean	Igazzal tér vissza, ha a paraméterként kapott szám kielégíti a túlélésre vonatkozó feltételt.

2.4 FileHandler

A fájlkezeléssel foglalkozó osztály.

Attribútumok

-

Metódusok

-FileHandler()	Privát konstruktor, hogy ne legyen példányosítható.
+saveGrid(filename: String, delim: String, g: Grid)	Lementi a paraméterként kapott fájlba a játékmezőn található élő cellák koordinátáit szerializálás segítségével.
+loadGrid(filename: String, delim: String, g: Grid)	Beolvassa a paraméterként kapott fájlból az élő cellák koordinátáit, és hozzáadja a játéktérhez.
+getSaveNames(): ArrayList<String>	Visszatér a program főkönyvtárában található mentésfájlok neveivel.

2.5 GameOfLife

A main metódust tartalmazó osztály. Nem szerepel az osztálydiagramon.

Attribútumok

-

Metódusok

+main(args: String[])	A program main metódusa.

2.6 Grid

A játékteret megvalósító osztály. HashMapek helyett ConcurrentHashMap-eket használ a szálbiztosság érdekében.

Attribútumok

-aliveCells: ConcurrentHashMap<Cell, Integer>	Az életben lévő cellákat (kulcs) tárolja és ezen cellák élő szomszédjainak számát (érték).
-neighbours: ConcurrentHashMap<Coord, Integer>	Az életben lévő cellákkal szomszédos mezők koordinátáit (kulcs) tárolja és ezen mezők élő szomszédjainak számát (érték).
-startState: ConcurrentHashMap<Cell, Integer>	A szimuláció elindítása előtt életben lévő cellákat (kulcs) tárolja és ezen cellák élő szomszédjainak számát (érték).
-rule: EvolutionRule	Az evolúciós szabályt tárolja, ami alapján a szimuláció fut.

Metódusok

+Grid(r: EvolutionRule)	Konstruktor, paramétere az evolúciós szabály.
+getRule(): EvolutionRule	Visszatér az evolúciós szabállyal.
+setRule(r: EvolutionRule)	Beállítja az evolúciós szabály értékét.
+updateNeighbours()	Végigmegy az élő cellákon és beállítja élő szomszédjaik számát, valamint frissíti a tömböt, amiben a szomszédokat tároljuk.
+getAliveNeighbours(c: Coord): int	Visszatér a paraméterként kapott koordinátához tartozó mező élő szomszédjainak számával.
+nextGeneration()	Megvizsgálja, hogy mely cellák maradnak életben, és melyek születnek meg, majd frissíti az aliveCells ConcurrentHashMapet.
+getAliveCells(): ConcurrentHashMap<Cell, Integer>	Visszatér az élő cellákkal.
+getStartState(): ConcurrentHashMap<Cell, Integer>	Visszatér a kezdőállapotban található cellákkal.
+getNeighbours(): ConcurrentHashMap<Coord, Integer>	Visszatér a neighbours ConcurrentHashMappel.
+setStartState()	A jelenlegi élő cellákat beállítja a kezdőállapotban található cellákra.

2.7 GridGraphics

A játéktér kirajzolásáért, a grafikáért felelős osztály. A JPanel osztályból örököltetjük.

Attribútumok

-WIDTH: int	A játéktér szélessége.
-HEIGHT: int	A játéktér magassága.
-grid: Grid	A kirajzolandó Gridet tartalmazó mező.
-zoom: double	A nagyítás értéke.
-screenOffsetX: int	Az eltolás értéke x tengelyen.
-screenOffsetY: int	Az eltolás értéke y tengelyen.

Metódusok

+GridGraphics(g: Grid)	Konstruktor, paramétere a kirajzolandó Grid.
+getGrid(): Grid	Visszatér a Grid mezővel.
+paintComponent(g: Graphics)	A JComponentből örökölt kirajzoló függvény felülírása.
+drawHexGridLoop(g: Graphics, origin: Point, size: int, radius: int, filledhex: ConcurrentHashMap<Cell, Integer>)	A hexagon rács kirajzolásáért felelős függvény. A paraméterként kapott élő cellákon is végigmegy, a helyükre kitöltött hatszöget rak.
-drawHex(g: Graphics, x: int, y: int, r: int, filled: boolean)	Hatszögeket kirajzoló függvény, a drawHexGridLoop() hívja meg.

+stepSimulation()	Lépteti a szimulációt, azaz meghívja az updateNeighbours() és a nextGeneration() függvényeket.
+getZoom(): double	Visszatér a nagyítás értékével.
+setZoom(z: double)	Beállítja a zoom értékét.
+getWidth(): int	Visszatér a játéktér szélességével.
+getHeight(): int	Visszatér a játéktér magasságával.
+getScreenOffsetX(): int	Visszatér az x tengelyen való eltolással.
+getScreenOffsetY(): int	Visszatér az y tengelyen való eltolással.
+setScreenOffsetX(i: int)	Beállítja az x tengelyen történő eltolás értékét.
+setScreenOffsetY(i: int)	Beállítja az y tengelyen történő eltolás értékét.

2.8 HexTile

A hatszög mezőkért felelős osztály. A Polygon osztályból örököltetjük.

Attribútumok

-SIDES: int	Konstans, értéke 6.
-rotation: int	A hatszög mező elforgatásának mértéke fokban. Értéke 90.
-points: Point[]	A hatszög csúcsai.
-center: Point	A hatszög középpontja.
-radius: int	A hatszög köré írható kör sugara.

Metódusok

+HexTile(x: int, y: int, radius: int)	Konstruktor, paraméterei a középpont x és y koordinátái és a sugár értéke.
+getPoints(): Point[]	Visszatér a hatszög csúcsaival.
+getRadius(): int	Visszatér a sugár értékével.
+setRadius(radius: int)	Beállítja a sugár értékét.
+setCenter(x: int, y: int)	Beállítja a hatszög középpontját.
#findPoint(angle: double): Point	A paraméteként kapott szög alapján megkeresi a hatszög egy pontját.
#updatePoints()	A hatszög összes pontját megkeresi, majd ezeket belerakja a points tömbbe.
+draw(g: Graphics2D, x: int, y: int, lineThickness: int, colorValue: int, filled: bollean)	A hatszög saját kirajzoló függvénye, ezt hívja meg a GridGraphics osztály drawHex függvénye.

2.9 MenuFrame

A menüért és az ablak megjelenítéséért felelős osztály. Megvalósítja az ActionListener interfészt és belső osztályokat is tartalmaz (lásd később). A JFrame-ből örököltetjük.

Attribútumok

-timer: Timer	Az időzítő, amely irányítja a szimulációt.
-gameSpeed: int	A játék sebessége.
-timerIsRunning: boolean	Értéke igaz, ha fut az időzítő.
-startb: JButton	A start gomb, elindítja a szimulációt.
-stopb: JButton	A stop gomb, megállítja a szimulációt.
-nextb: JButton	A next gomb, eggyel lépteti a szimulációt.
-resetb: JButton	A reset gomb, visszaállítja a kezdőállapotot.
-clearb: JButton	A clear gomb, kitörli az összes élő cellát.
-saveb: JButton	A save gomb, lementi a játék állását
-loadb: JButton	A load gomb, betölt egy korábbi játékállást.
-applyb: JButton	Az apply gomb, megváltoztatja az evolúciós szabályt.
-speeds: JSlider	Játéksebességet állító slider.
-rulef: JTextField	Ide írhatunk be új evolúciós szabályt.
-loadbox: JComboBox	A mentések neveit tartalmazó JComboBox.
-savef: JTextField	A mentés nevét állítja be.
-saveNames: ArrayList<String>	Az összes mentés nevét tartalmazó lista.

Metódusok

+initComponents()	Inicializáló függvény, létrehozza és beállítja a menü komponenseit.
+MenuFrame(g: GridGraphics)	Konstruktor, paramétere a GridGraphics osztály egy példánya.
+actionPerformed(e: ActionEvent)	Az interface függvénye. A különböző gombokra valósítja meg a megnyomásukkor végbemenő eseményeket.
+timerStart()	Elindítja az időzítőt a gameSpeed jelenlegi értéke alapján.
+timerStop()	Megállítja az időzítőt.
+setGameSpeed(s: int)	Beállítja a játék sebességét.

2.10 ML

Belső osztály, a MenuFrame osztály tartalmazza. A MouseInputAdatper osztályból örököl. Feladata az egér által irányított funkciók megvalósítása.

Attribútumok

-prevPos: Point	A nézet mozgatásánál tárolja az előző mintavételezés pozícióját.
-drag: boolean	Ha épp mozgatjuk a nézetet, akkor értéke igaz, egyébként hamis.
-draw: boolean	Ha épp rajzolunk a játéktéren, akkor értéke igaz, egyébként hamis.
-lastCoordX: int	A cella x koordinátája, amibe utoljára kattintottunk.
-lastCoordY: int	A cella y koordinátája, amibe utoljára kattintottunk.

Metódusok

+mouseDragged(e: MouseEvent)	Az egér húzásakor történő eseményeket állítja be. Jelen esetben ez a rajzolás és a nézet mozgatása.
+mousePressed(e: MouseEvent)	Az egérgomb lenyomására történő esemény. Bal egérgombra a draw mezőt állítja igazra, a jobb egérgomb esetén a drag mezőt.
+mouseClicked(e: MouseEvent)	Az egérgomb lenyomására, majd felengedésére végbemenő esemény. A cellák elhelyezését valósítja meg.
+mouseReleased(e: MouseEvent)	Az egér felengedésekor végbemenő esemény. Hamisra állítja a draw és drag mezők értékét.
+mouseWheelMoved(e: MouseWheelEvent)	Az egérgörgőre végbemenő esemény. A nagyítás értékét állítja be.

2.11 SliderListener

Belső osztály, a MenuFrame osztály tartalmazza. A ChangeListener interfészt valósítja meg. Feladata a sebességet állító slider eseményeinek beállítása.

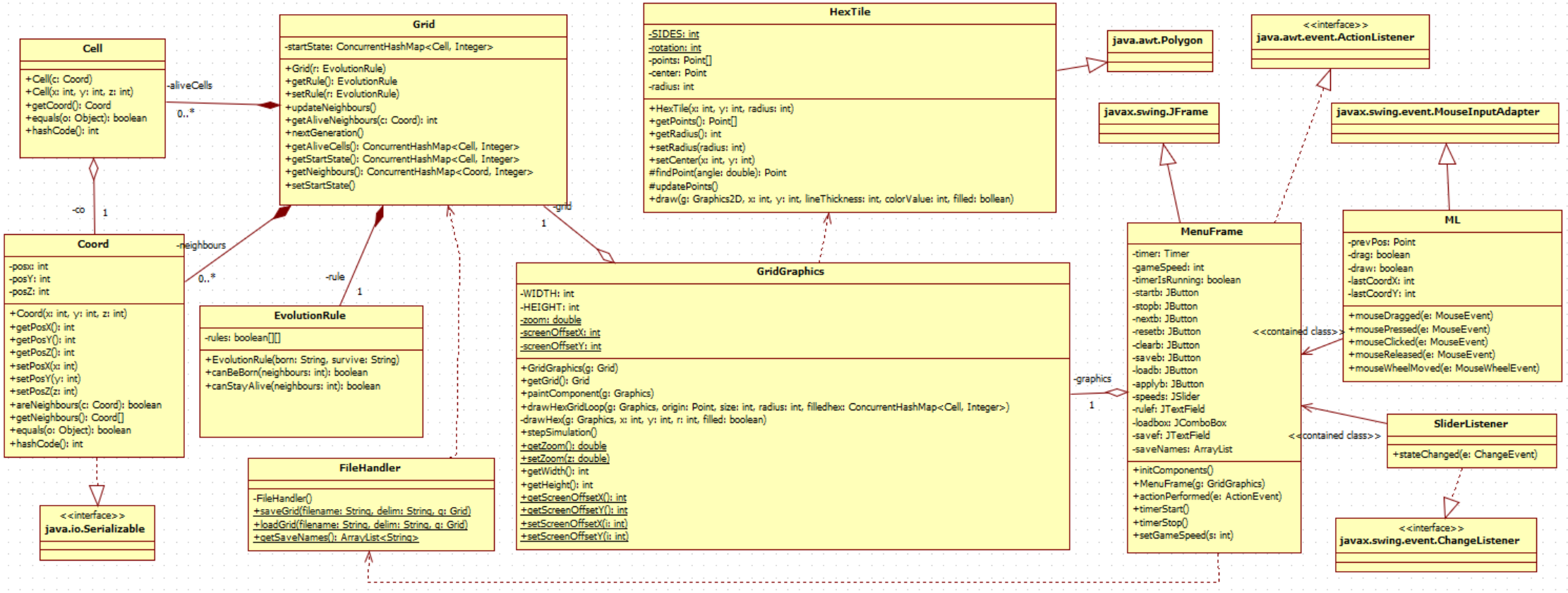
Attribútumok

-

Metódusok

+stateChanged(e: ChangeEvent)	Akkor bekövetkező esemény, amikor a slider állapota megváltozik. Ilyenkor átállítja a játék sebességét.

3. Osztálydiagram



4. Tesztelés

A program tesztelése a JUnit4 segítségével történik. A tesztek 5 különböző osztály, összesen 11 metódusát teszteli.

4.1 CoordTest

A Coord osztály equals, areNeighbours és getNeighbours nevű függvényeit teszteli összesen 4 tesztesetben.

4.2 EvolutionRuleTest

Az EvolutionRule osztályon hajt végre egy paraméteres tesztet. A canBeBorn és a canStayAlive metódusokat teszteli, 4 tesztesetben futásonként.

4.3 FileHandlerTest

A FileHandler osztályt teszteli. A setup metódusban a „test.txt” nevű fájlba ment ki egy állapotot, majd ennek beolvasását teszteli a FileHandler osztály loadGrid metódusának segítségével.

4.4 GridTest

A Grid osztály updateNeighbours, nextGeneration és setStartState metódusait teszteli 3 tesztesetben.

4.5 HexTileTest

A HexTile osztály findPoint és updatePoints metódusait teszteli 2 tesztesetben.

5. Felhasználói kézikönyv

A programban a Game of Life nevű közismert játékot próbálhatja ki a felhasználó egy hatszögekből álló pályán. A játék irányítása az egérrel és a játéktér alatt elhelyezkedő menügombokkal történik. Ezek a gombok a következők (sorrendjük megegyezik a programban található sorrenddel):

- Start: Ha a szimuláció nem fut, akkor elindítja.
- Stop: Ha a szimuláció fut, akkor leállítja. Ilyenkor a pályán a sejtek állapota megmarad.
- Next: Eggyel lépteti a szimulációt. Csak akkor használható, ha a szimuláció egyébként nem fut.
- Reset: Leállítja a szimulációt és a pálya állapotát visszaállítja abba a helyzetbe, ami a szimuláció elindítása előtt állt fenn.
- Clear: Kitörli az összes élő sejtet a játéktérről, de nem befolyásolja a szimuláció futását.

A játék indításakor a játéktér kezdőállapota üres. A felhasználó úgy helyezhet el új cellákat, ha az egérmutatót a megfelelő üres cellára viszi és megnyomja a bal egérgombot. Ha élő sejtre kattint, akkor abból a cellából kiveszi az élő sejtet. A bal egérgomb nyomva tartásával „rajzolni” is lehet a pályára, ilyenkor minden élő sejt meghal, valamint minden üres cellába élő sejtet helyez, amelyek fölött elhúzza az egeret.

A nézet mozgatása lehetőséget nyújt arra, hogy a felhasználó szabadon megtekinthesse a játéktér bármely részét. Ezt úgy teheti meg, ha nyomva tartja a jobb egérgombot és közben elmozdítja az egeret az egyik irányba. A nézet eltolása az egérmutató mozgásával megegyező irányban és mértékben történik.

A nézet változtatásához tartozik még a nagyítás mértékének beállítása. Ha az egér görgőjét a felhasználó saját maga felé görgeti, akkor távolabbról, ha az ellenkező irányba, akkor közelebbről láthatja a pályát. A nagyításnak vannak korlátjai mindkét irányban, ha ezt eléri, akkor a görgő ugyanolyan irányú mozgatása nem csinál semmit.

A program lehetőséget nyújt arra is, hogy a játéktér állapotát bármikor elmentsük, valamint korábbi játékállásokat töltsünk be. A következő két gomb erre vonatkozik:

- Save: A gombtól balra található szövegmezőbe gépelve adhatja meg a felhasználó a mentés nevét. Ezután a Save gombra kattintva elmentheti a játéktér jelenlegi állapotát a megadott névvel. Üres neveket a program nem fogad el, az elutasított nevet felugró ablakkal jelzi. Ha már létező mentés nevét adjuk meg, akkor a régi mentés felülíródik.
- Load: A gombtól balra található lenyitható mezőben jelennek meg a mentések nevei. Ezek közül egyet kiválasztva, majd a Load gombra kattintva a felhasználó betöltheti a megfelelő mentést. A mentéseket tartalmazó doboz minden új mentés hozzáadása után automatikusan frissül.

A szimuláció sebessége egy közepes alapértékre van beállítva a program indulásakor, ennek megváltoztatására a „Speed” felirattal ellátott csúszka ad lehetőséget. A csúszkán a gombra kattintva és balra húzva csökkenthetjük, jobbra húzva pedig növelhetjük a szimuláció sebességét.

Az evolúciós szabály két részből áll. Az első rész azt mondja meg, hány elő szomszéd esetén születik meg egy olyan mező, amelyik jelenleg nem él, a másik része pedig azt, hogy hány élő szomszéd esetén éli túl egy mező a következő generációig. Az alapértelmezett szabály Born/Survive: 3/23, ami azt jelenti, hogy egy cella 3 szomszéd esetén születik meg, és 2 vagy 3 élő szomszéd esetén maradhat életben, más esetekben elpusztul. A felhasználónak lehetősége van változtatni ezen a szabályon. Ezt a „Change rule" felirattal ellátott szövegmezőbe az új szabályt beírva, majd az Apply gombra nyomva teheti meg. Fontos megkötések, hogy a szabálynak mindig tartalmaznia kell egy '/' karaktert, és a '/' jel egyik oldala sem lehet teljesen üres. Ilyenkor a program felugró ablakkal jelzi a hibás szabályt. Ha olyan szabályt akar megadni a felhasználó, hogy például soha ne élje túl egy cella, akkor azt szóközzel jelezheti a szabály beírásakor a '/' megfelelő oldalán (vagy akármilyen más karakterrel, ami nem szám).

A programból a jobb felső sarokban található „X" gomb segítségével léphetünk ki.

6. Eltérések a specifikációtól

- A State enumeráció teljesen megszűnt, mert nem volt lényegi funkciója a programban.
- A Cell osztály nem tartalmazza az evolúciós szabályt, mert nincs szüksége a szabály ismeretére.
- A Grid osztályban HashSet helyett ConcurrentHashMap használata, mert szükség van a szálbiztosságra és a szomszédjaikat is egyszerűbb így tárolni.

7. Felhasznált segédanyagok, források

Hatszögkoordináták fajtái, általános megértésük:
https://www.redblobgames.com/grids/hexagons/

Egy ötlet a hatszögrács megvalósítására, megjelenítésére java-ban:

https://stackoverflow.com/questions/20734438/algorithm-to-generate-a-hexagonal-grid-with-coordinate-system