HÁZI FELADAT

The Game of Life, avagy az élet játéka

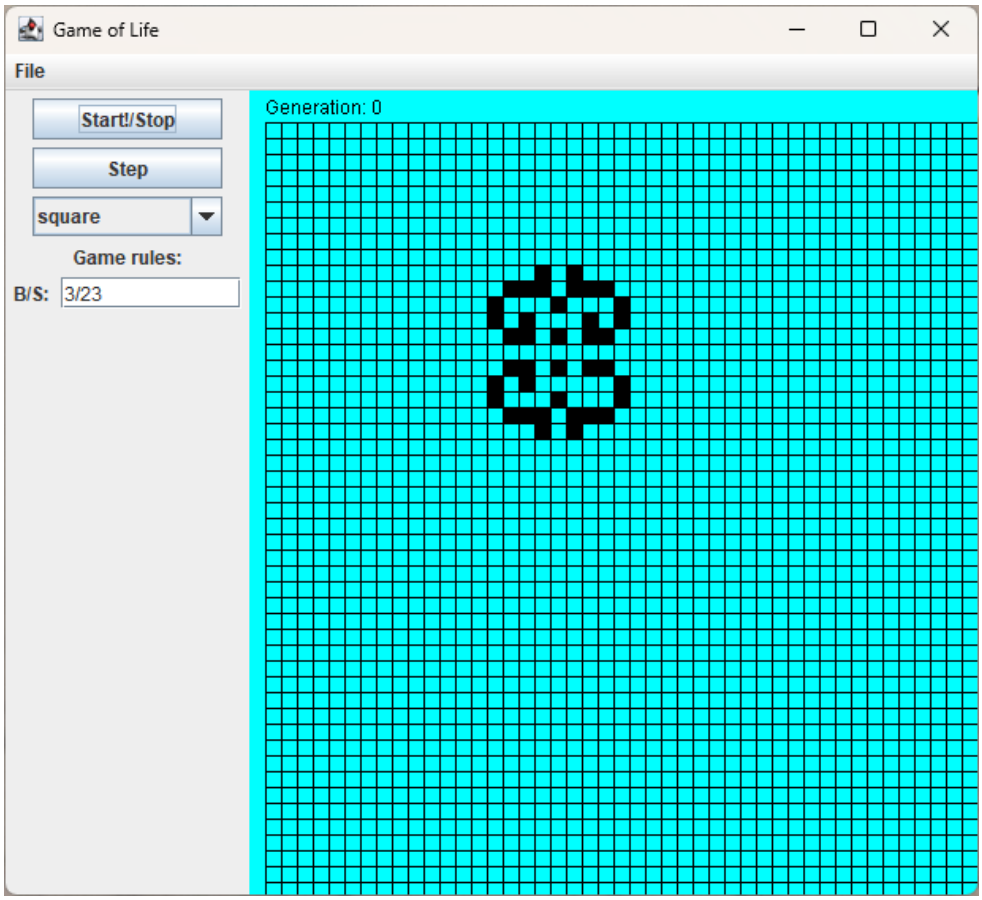
Felhasználói dokumentáció

Programozás alapjai 3.
Tóth Gábor
F041OM
2022.november 28.

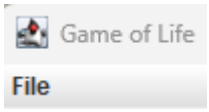
Az élet játéka John Horton Conway Cambridge-i Egyetem matematikusának a nevéhez fűződik. A játék egy sejtautómata, s ahogy a neve is utal rá, a sejtek születését, fejlődését majd halálát mutatja.

Az alapszabály az, hogy minden cella kettő állapotban lehet vagy élő vagy halott. Egy cella akkor születik meg, ha Moore környezetében pontosan 3 cella van életben, és akkor marad életben, ha 2 vagy 3 élő szomszédja van.

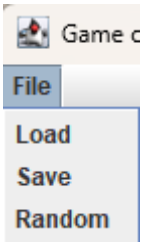
A program felülete:



A program fejlécén egy menü érhető el



A menüpontra kattintva, lenyílik egy fül, melyből 3 további fül érhető el.



- „Load” fülre kattintva a program betölti a mentett pályát
- „Save” füllel menthetjük a felületet.
- „Random” gombra kattintva pedig a program maga generálja a pályákat.

Felület bal oldalán a kezelőfelületben az alábbi gombokat érjük el:

- Start/Stop gombbal elindíthatjuk a játékot, vagy megállíthatjuk azt.
- Step gombbal a szimuláción csak a következő generációt jeleníti meg.
- Step gomb alatti legördülő menüsorból alakzatot kiválasztva, beállíthatjuk, hogy a celláink milyen alakúak legyenek.
- B/S (born/Survive) mezőben beírhatjuk az alkalmazni kívánt játékszabályt, a következő formátumban: Először azokat a számokat, ahány szomszéd esetén szeretnénk, ha új cella születne, majd per „/” jellel elválasztva, azokat a számokat, ahány szomszéd esetén szeretnénk, ha egy cella életben maradna. A változtatás a következő elindításnál lép majd életbe. A példában a Conway féle Életjátk szabályai szerepelnek, és ez a program alapbeállítása.

Nagyházi

Generated by Doxygen 1.9.5

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

File Index

3.1 File List

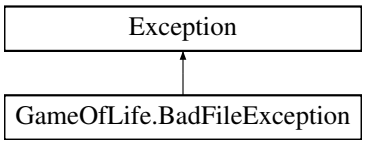
Here is a list of all documented files with brief descriptions:

Chapter 4

Class Documentation

4.1 GameOfLife.BadFileException Class Reference

Inheritance diagram for GameOfLife.BadFileException:



Public Member Functions

- BadFileException (String msg)

4.1.1 Detailed Description

Exception, hogyha a beolvasandó txt szintaktiai hibás. Különösbben fontos szerepe nincs.

Definition at line 6 of file BadFileException.java.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 BadFileException()

```
GameOfLife.BadFileException.BadFileException (
            String msg )
```

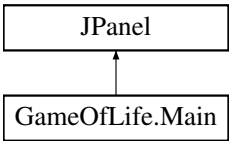
Definition at line 8 of file BadFileException.java.

The documentation for this class was generated from the following file:

- src/GameOfLife/BadFileException.java

4.2 GameOfLife.Main Class Reference

Inheritance diagram for GameOfLife.Main:



Static Public Member Functions

- static void main (String[ ] args)

4.2.1 Detailed Description

Definition at line 5 of file Main.java.

4.2.2 Member Function Documentation

4.2.2.1 main()

```
static void GameOfLife.Main.main (
            String[ ] args )  [static]
```

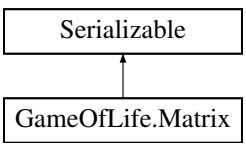
Definition at line 8 of file Main.java.

The documentation for this class was generated from the following file:

- src/GameOfLife/Main.java

4.3 GameOfLife.Matrix Class Reference

Inheritance diagram for GameOfLife.Matrix:



Public Member Functions

- Matrix (String file) throws IOException, BadFileException
- Matrix (int m, int n)
- int width ()
- int height ()
- int getValueAt (int rowNumber, int columnNumber)
- void setValueAt (int rowNumber, int columnNumber, int value)
- void randomFill ()
- boolean equals (Object o)

4.3.1 Detailed Description

Ez az osztály tárolja a pályát, és tölti be, txt file-ból.

Definition at line 9 of file Matrix.java.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 Matrix() [1/2]

```
GameOfLife.Matrix.Matrix (
            String file ) throws IOException, BadFileException
```

Az osztáy konstruktora. Egy megadott és megfelelően formázott szövegese file-ból betölti a pályát. A formázás↩ : Az értékeket `tab` választja el, és a sorok új sorba. Ahol nincs érték, ott nulla lesz, ez a java inicializációs tulajdonságaiból jön.

Parameters

file	Az a file neve, amit be szeretnénk olvasni, a projekt legfelső szintjén kell lennie.

Exceptions

IOException	Csak akkor dobja, ha valami belső függvény dobja, tehát mondjuk a file nem található, vagy írásvédett stb.
BadFileException	Akkor tobbja, ha a formázás nem helyes, és tudja beolvasni.

Definition at line 26 of file Matrix.java.

4.3.2.2 Matrix() [2/2]

```
GameOfLife.Matrix.Matrix (
            int m,
            int n )
```

Konstruktor. Egy üres mátrixot hoz létre, csak inicialiációs szempontból létezik, hogy utána ez értékeit módosíthassuk.

Parameters

m	szélesség
n	magasság

Definition at line 62 of file Matrix.java.

4.3.3 Member Function Documentation

4.3.3.1 equals()

```
boolean GameOfLife.Matrix.equals (
            Object o )
```

Egyenlőség vizsgáló. Magát a 2D-s tömböt deepEquals-al vizsgálja, így a mátrix minden értékének egyezése esetén ad vissza igaz értéket.

Parameters

o	A jobb oldali vizsgálandó Matrix

Returns

Igaz, ha egyenlőek a mátrixok értékei, különben hamis

Definition at line 124 of file Matrix.java.

4.3.3.2 getValueAt()

```
int GameOfLife.Matrix.getValueAt (
            int rowNumber,
            int columnNumber )
```

egy bizonyos sor és oszlop által meghatározott cella értékét adja vissza

Parameters

rowNumber	a sor száma
columnNumber	az oszlop száma.

Returns

a sor és oszlop álltal meghatározott érték

Definition at line 90 of file Matrix.java.

4.3.3.3 height()

```
int GameOfLife.Matrix.height ( )
```

A pálya magasságát (azaz az N-t) adja vissza.

Returns

a pálya magassága.

Definition at line 80 of file Matrix.java.

4.3.3.4 randomFill()

```
void GameOfLife.Matrix.randomFill ( )
```

Feltölti a matrix minden celláját random értékkel. a mérete ugyan akkora mint ami előzőleg volt. Megoldható, hogy nagyobb/kisebb legyen, de a programban minden plya 50x50-es, ezért ez is.

Definition at line 108 of file Matrix.java.

4.3.3.5 setValueAt()

```
void GameOfLife.Matrix.setValueAt (
            int rowNumber,
            int columnNumber,
            int value )
```

Sor ész oszlop szám álltat meghatározott cella értékét állytja be.

Parameters

rowNumber	a sor száma
columnNumber	az oszlop száma
value	a beállítandó érték.

Definition at line 100 of file Matrix.java.

4.3.3.6 width()

```
int GameOfLife.Matrix.width ( )
```

A pálya szélességét (azaz az M-t) adja vissza.

Returns

a pálya szélessége.

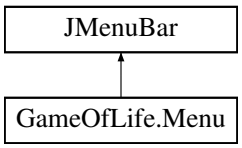
Definition at line 72 of file Matrix.java.

The documentation for this class was generated from the following file:

- src/GameOfLife/Matrix.java

4.4 GameOfLife.Menu Class Reference

Inheritance diagram for GameOfLife.Menu:



Classes

- class **LoadSaveFileListener**
- class **MenuItemListener**
- class **RandomActionListener**
- class **SaveMatrixActionListener**

4.4.1 Detailed Description

Menürendszer. Az ablak tetején megjelenő menurendszer. Egyelten nagy metódusból áll, ami létrehoz mindent. Egyébként még pár ActionListener-ből áll, a menük kezelésésre. Nagyrészt innen vettem át a kódot: tutorialspoint

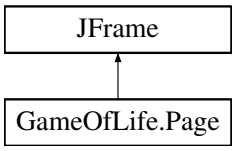
Definition at line 13 of file Menu.java.

The documentation for this class was generated from the following file:

- src/GameOfLife/Menu.java

4.5 GameOfLife.Page Class Reference

Inheritance diagram for GameOfLife.Page:



Classes

- class **startStopButtonActionListener**
- class **stepButtonActionListener**

Public Member Functions

- Page (String title)

Static Public Member Functions

- static void readBS ()
- static Table getEast ()

4.5.1 Detailed Description

Az egész ablakot megvalósító, tároló és kezelő osztály. A main csak meghívja ennek az egyetlen Konstruktorát.

Definition at line 13 of file Page.java.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Page()

```
GameOfLife.Page.Page (
            String title )
```

Konstruktor. Létrehozza az ablakot, és hozzáad mindent, ami kell. Alapvetően 3 egységre osztik.

1. Menu: menusáv. a Menu osztály kezeli.
2. East: Table osztály példánya, a pálya megjelenítését csinálja.
3. West: az irányítófelület, JPanel osztálya. Továbbá van benn egy windowActionListener, ami azt a célt szolgálja, hogy ha kilépünk a programból, akkor elmenti, a szimuláció aktuálsi szabályát.

Parameters

title	Az ablak címe.

Definition at line 34 of file Page.java.

4.5.3 Member Function Documentation

4.5.3.1 getEast()

```
static Table GameOfLife.Page.getEast ( )  [static]
```

Definition at line 188 of file Page.java.

4.5.3.2 readBS()

```
static void GameOfLife.Page.readBS ( )  [static]
```

A játékszabályokat olvassa be és értelmezi, majd tárolja el a megfeleő helyen (Table.rules), hogy szimuláció használni tudja

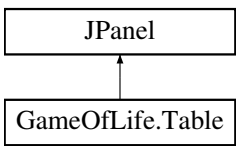
Definition at line 172 of file Page.java.

The documentation for this class was generated from the following file:

- src/GameOfLife/Page.java

4.6 GameOfLife.Table Class Reference

Inheritance diagram for GameOfLife.Table:



Classes

- enum **Shapes**

Public Member Functions

- Table (String file)
- void simulationStop ()
- void simulationStart ()

Static Public Member Functions

- static void loadGrid (String file) throws IOException, BadFileException
- static Color getBackgroundColor ()
- static void saveMatrix (String file) throws IOException
- static void loadMatrix (String file) throws IOException, ClassNotFoundException
- static void randomMatrix ()
- static HashMap< String, ArrayList< Integer > > getRules ()
- static void setRules (HashMap< String, ArrayList< Integer > > rules)

Protected Member Functions

- void paintComponent (Graphics g)

4.6.1 Detailed Description

A Tábla osztály. Feladat az ablak jobb oldalának kijelzése, és kezelése. A kijelzést a Graphics osztéllyan valósítja meg. Továbbá itt kapott helyet a szimuláció futtatása is.

Definition at line 14 of file Table.java.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 Table()

```
GameOfLife.Table.Table (
            String file )
```

Konstruktor. Betölti pályát.

Parameters

file	A file neve amit be kell töltenie. A projekt legfelső szintjén kell, hogy legyen.

Definition at line 46 of file Table.java.

4.6.3 Member Function Documentation

4.6.3.1 getBackgroundColor()

```
static Color GameOfLife.Table.getBackgroundColor ( )  [static]
```

Definition at line 209 of file Table.java.

4.6.3.2 getRules()

```
static HashMap< String, ArrayList< Integer > > GameOfLife.Table.getRules ( )  [static]
```

A játék szabályait adja, vissza, eredeti(HashMap) formában, fent meghatározott módon.

Returns

A szabályok.

Definition at line 250 of file Table.java.

4.6.3.3 loadGrid()

```
static void GameOfLife.Table.loadGrid (
            String file ) throws IOException, BadFileException  [static]
```

csak egy adapter metódus, hogy a Page-ből lehessem betölteni, meghatározott nevő file-t. Így ez a metódus, csak a Matrix mgfelelő konstruktorát hívja meg.

Parameters

file	A beolvasandó file neve. A projekt legfelső mappájában kell lennie.

Exceptions

IOException	Ha a filet, nem tudja, megnyitni
BadFileException	Ha a file formázása nem megfelelő, és nem tudja beolvasni

See also

Matrix

Definition at line 184 of file Table.java.

4.6.3.4 loadMatrix()

```
static void GameOfLife.Table.loadMatrix (
            String file ) throws IOException, ClassNotFoundException  [static]
```

A Matrix desorosítása. A MAtrix minden attribútumának beolvasása, a megadott file-ból.

Parameters

file	A file amiből olvasni szeretnénk.

Exceptions

IOException	Ha a file valamiért nem olvasható.
ClassNotFoundException	Ha az osztály nem található. Nem szabad, hogy keletkezzen.

Definition at line 232 of file Table.java.

4.6.3.5 paintComponent()

```
void GameOfLife.Table.paintComponent (
            Graphics g )  [protected]
```

A JPanel metósuda felüldefiniálva. Megnézni, hogy négy- vagy háromszöget kell rajtolnia, és meghívja a szükséges metódust.

Parameters

g	the `Graphics` object to protect

Definition at line 63 of file Table.java.

4.6.3.6 randomMatrix()

```
static void GameOfLife.Table.randomMatrix ( )  [static]
```

Kívülről elérhető metódus új random matrix generálására.

Definition at line 242 of file Table.java.

4.6.3.7 saveMatrix()

```
static void GameOfLife.Table.saveMatrix (
            String file ) throws IOException  [static]
```

A Matrix sorosítás. A Matrix, és annak minden attribútumának kiírása, java standard módon.

Parameters

file	A célfile, amibe írni szeretnénk.

Exceptions

IOException	Ha a file valamiért nem írható.

Definition at line 218 of file Table.java.

4.6.3.8 setRules()

```
static void GameOfLife.Table.setRules (
            HashMap< String, ArrayList< Integer > > rules )  [static]
```

A játékszabályok beállítása, fent meghatározz formátumban HashMap-ben.

Parameters

rules	

Definition at line 258 of file Table.java.

4.6.3.9 simulationStart()

```
void GameOfLife.Table.simulationStart ( )
```

Kivülről meghívható metódus a szimuláció megállítására.

Definition at line 198 of file Table.java.

4.6.3.10 simulationStop()

```
void GameOfLife.Table.simulationStop ( )
```

Kívülről meghívható metósud a szimuláció indítására.

Definition at line 191 of file Table.java.

The documentation for this class was generated from the following file:

- src/GameOfLife/Table.java

Chapter 5

File Documentation

5.1 BadFileException.java

```
00001 package GameOfLife;
00002
00006 public class BadFileException extends Exception{
00007     String msg;
00008     public BadFileException(String msg) {
00009         this.msg = msg;
00010     }
00011 }
```

5.2 Main.java

```
00001 package GameOfLife;
00002
00003 import javax.swing.*;
00004
00005 public class Main extends JPanel {
00006     static Page frame = new Page("Game of Life");
00007
00008     public static void main(String[] args) {
00009         frame.setVisible(true);
00010     }
00011
00012 }
```

5.3 Matrix.java

```
00001 package GameOfLife;
00002
00003 import java.io.*;
00004 import java.util.Arrays;
00005
00009 public class Matrix implements Serializable {
00010     //A pálya szélessége
00011     private int M = 50;
00012     //A pálya magassága
00013     private int N = 50;
00014     //A nátrix ami 1/0 értéket tárol attól függően, hogy a cella él-e vagy sem.
00015     private int[][] matrix = new int[M][N];
00016
00026     public Matrix(String file) throws IOException, BadFileException {
00027         int[][] safemx = matrix.clone();
00028         File grid = new File(file);
00029         FileReader fr = new FileReader(grid);
00030         BufferedReader br = new BufferedReader(fr);
00031         String[] read;
00032         int sor = 0;
00033         br.mark(200);
00034         int size = (br.readLine().length()+1)/2;
00035         N = M = size;
```

```
00036         matrix = new int[size][size];
00037         br.reset();
00038         while (true) {
00039             String line = br.readLine();
00040             if (line == null) break;
00041             read = line.split("\t");
00042             for (int i=0; i<read.length; i++){
00043                 try {
00044                     matrix[sor][i] = Integer.parseInt(read[i]);
00045                 } catch (NumberFormatException e){
00046                     matrix = safemx.clone();
00047                     N = safemx.length;
00048                     M = safemx[0].length;
00049                     throw new BadFileException("Hiba a szám olvasásánál @ " + sor + ":" + i);
00050                 }
00051             }
00052             sor++;
00053         }
00054         br.close();
00055     }
00056
00062     public Matrix(int m, int n) {
00063         M = m;
00064         N = n;
00065         matrix = new int[m][n];
00066     }
00067
00072     public int width(){
00073         return M;
00074     }
00075
00080     public int height(){
00081         return N;
00082     }
00083
00090     public int getValueAt(int rowNumber, int columnNumber){
00091         return matrix[rowNumber][columnNumber];
00092     }
00093
00100     public void setValueAt(int rowNumber, int columnNumber, int value){
00101         matrix[rowNumber][columnNumber] = value;
00102     }
00103
00108     public void randomFill(){
00109         matrix = new int[N][M];
00110         for (int i = 0; i < N; i++){
00111             for (int j = 0; j < M; j++){
00112                 matrix[i][j] = Math.round((float)Math.random());
00113             }
00114         }
00115     }
00116
00123     @Override
00124     public boolean equals(Object o) {
00125         if (this == o) return true;
00126         if (o == null || getClass() != o.getClass()) return false;
00127
00128         Matrix matrix1 = (Matrix) o;
00129
00130         if (M != matrix1.M) return false;
00131         if (N != matrix1.N) return false;
00132         return Arrays.deepEquals(matrix, matrix1.matrix);
00133     }
00134 }
```

5.4 Menu.java

```
00001 package GameOfLife;
00002
00003 import javax.swing.*;
00004 import java.awt.event.*;
00005 import java.io.IOException;
00006
00007
00013 public class Menu extends JMenuBar{
00014
00015     Menu() {
00016         showMenu();
00017     }
00018
00019     private void showMenu(){
00020         //create menus
00021         JMenu fileMenu = new JMenu("File");
```

```
00022         JMenu editMenu = new JMenu("Edit");
00023 
00024         JMenuItem openMenuItem = new JMenuItem("Load");
00025         openMenuItem.setActionCommand("Load");
00026 
00027         JMenuItem saveMenuItem = new JMenuItem("Save");
00028         saveMenuItem.setActionCommand("Save");
00029 
00030         JMenuItem randomMenuItem = new JMenuItem("Random");
00031         saveMenuItem.setActionCommand("Random");
00032 
00033         JMenuItem exitMenuItem = new JMenuItem("Exit");
00034         exitMenuItem.setActionCommand("Exit");
00035 
00036         JMenuItem cutMenuItem = new JMenuItem("Cut");
00037         cutMenuItem.setActionCommand("Cut");
00038 
00039         JMenuItem copyMenuItem = new JMenuItem("Copy");
00040         copyMenuItem.setActionCommand("Copy");
00041 
00042         JMenuItem pasteMenuItem = new JMenuItem("Paste");
00043         pasteMenuItem.setActionCommand("Paste");
00044 
00045         MenuItemListener menuItemListener = new MenuItemListener();
00046 
00047         openMenuItem.addActionListener(new LoadSaveFileListener());
00048         saveMenuItem.addActionListener(new SaveMatrixActionListener());
00049         randomMenuItem.addActionListener(new RandomActionListener());
00050         exitMenuItem.addActionListener(menuItemListener);
00051         cutMenuItem.addActionListener(menuItemListener);
00052         copyMenuItem.addActionListener(menuItemListener);
00053         pasteMenuItem.addActionListener(menuItemListener);
00054 
00055         final JRadioButtonMenuItem showLinksMenu = new JRadioButtonMenuItem(
00056                 "Show Links", true);
00057 
00058         fileMenu.add(openMenuItem);
00059         fileMenu.add(saveMenuItem);
00060         fileMenu.add(randomMenuItem);
00061         fileMenu.addSeparator();
00062         fileMenu.add(showLinksMenu);
00063         fileMenu.addSeparator();
00064         fileMenu.add(exitMenuItem);
00065 
00066         editMenu.add(cutMenuItem);
00067         editMenu.add(copyMenuItem);
00068         editMenu.add(pasteMenuItem);
00069 
00070         //add menu to menubar
00071         add(fileMenu);
00072         add(editMenu);
00073     }
00074 
00075     class MenuItemListener implements ActionListener {
00076         public void actionPerformed(ActionEvent e) {
00077             System.out.println(e.getActionCommand() + " JMenuItem clicked.");
00078         }
00079     }
00080 
00081     class LoadSaveFileListener implements ActionListener{
00082         @Override
00083         public void actionPerformed(ActionEvent e) {
00084             try {
00085                 Table.loadMatrix("map.mx");
00086             } catch (IOException | ClassNotFoundException ex) {
00087                 ex.printStackTrace();
00088             }
00089             Page.getEast().repaint();
00090         }
00091     }
00092     class SaveMatrixActionListener implements ActionListener {
00093         @Override
00094         public void actionPerformed(ActionEvent e) {
00095             try {
00096                 Table.saveMatrix("map.mx");
00097             } catch (IOException ex) {
00098                 ex.printStackTrace();
00099             }
00100         }
00101     }
00102 
00103     class RandomActionListener implements ActionListener {
00104         @Override
00105         public void actionPerformed(ActionEvent e) {
00106             Table.randomMatrix();
00107             Page.getEast().repaint();
00108         }
```

```
00109     }
00110
00111 }
```

5.5 Page.java

```
00001 package GameOfLife;
00002
00003 import javax.swing.*;
00004 import java.awt.*;
00005 import java.awt.event.*;
00006 import java.io.IOException;
00007 import java.util.ArrayList;
00008 import java.util.HashMap;
00009
00013 public class Page extends JFrame {
00014     // A szumláció éppen fut-e
00015     private static boolean running = false;
00016     //A pálya. Ami egy Table
00017     private static Table east = new Table("grid.txt");
00018     //A kezelő felület
00019     private static JPanel west = new JPanel();
00020     private static JComboBox comboBox;
00021     private JMenuBar menu;
00022     private static JTextField bsTextField;
00023
00034     public Page(String title){
00035         super(title);
00036         Dimension d1 = new  Dimension(120, 1);
00037
00038         JButton startStopButton = new JButton("Start!/Stop");
00039         startStopButton.addActionListener(new startStopButtonActionListener());
00040         startStopButton.setSize(d1);
00041         startStopButton.setMinimumSize(d1);
00042         startStopButton.setMaximumSize(d1);
00043
00044         JButton stepButton = new JButton("Step");
00045         stepButton.addActionListener(new stepButtonActionListener());
00046         stepButton.setSize(d1);
00047         stepButton.setMinimumSize(d1);
00048         stepButton.setMaximumSize(d1);
00049
00050         Object[] shapes = {"square", "triangle"};
00051         comboBox = new JComboBox<>(shapes);
00052         comboBox.setSize(d1);
00053         comboBox.setMinimumSize(d1);
00054         comboBox.setMaximumSize(d1);
00055
00056         JLabel ruleLabel = new JLabel("Game rules:");
00057         //b/s mező
00058         JPanel bs = new JPanel();
00059         JLabel bsLabel = new JLabel("B/S: ");
00060         bsTextField = new JTextField(10);
00061         bsTextField.setText("3/23");
00062         bs.add(bsLabel);
00063         bs.add(bsTextField);
00064
00065         east.setBackground(Table.getBackgroundColor());
00066
00067         west.setBounds(61, 11, 81, 140);
00068         west.setLayout(new BoxLayout(west, BoxLayout.Y_AXIS));
00069
00070         west.add(Box.createRigidArea(new Dimension(5, 5)));
00071         west.add(startStopButton);
00072         startStopButton.setAlignmentX(Component.CENTER_ALIGNMENT);
00073
00074         west.add(Box.createRigidArea(new Dimension(5, 5)));
00075         west.add(stepButton);
00076         stepButton.setAlignmentX(Component.CENTER_ALIGNMENT);
00077
00078         west.add(Box.createRigidArea(new Dimension(5, 5)));
00079         west.add(comboBox);
00080         comboBox.setAlignmentX(Component.CENTER_ALIGNMENT);
00081
00082         west.add(Box.createRigidArea(new Dimension(5, 5)));
00083         west.add(ruleLabel);
00084         ruleLabel.setAlignmentX(Component.CENTER_ALIGNMENT);
00085
00086         west.add(bs);
00087         bs.setAlignmentX(Component.CENTER_ALIGNMENT);
00088
00089         menu = new Menu();
00090         setJMenuBar(menu);
```

```
00091
00092         add(east, BorderLayout.CENTER);
00093         add(west, BorderLayout.WEST);
00094
00095         setVisible(true);
00096
00097         addWindowListener(new WindowAdapter() {
00098             @Override
00099             public void windowClosing(WindowEvent e) {
00100                 try {
00101                     Table.saveMatrix("map.mx");
00102                 } catch (IOException ex) {
00103                     ex.printStackTrace();
00104                 }
00105                 super.windowClosing(e);
00106             }
00107         });
00108
00109         Dimension d = new Dimension(630, 570);
00110         setMinimumSize(d);
00111         setSize(d);
00112         setDefaultCloseOperation(this.EXIT_ON_CLOSE);
00113         pack();
00114     }
00115
00119     static class startStopButtonActionListener implements ActionListener {
00120
00121         @Override
00122         public void actionPerformed(ActionEvent e) {
00123             readBS();
00124             String shape = (String) comboBox.getSelectedItem();
00125             if (shape.equals("square")){
00126                 east.shape = Table.Shapes.SQUARE;
00127             } else if(shape.equals("triangle")){
00128                 east.shape = Table.Shapes.TRIANGLE;
00129             } else{
00130                 System.out.println("Panic!");
00131             }
00132             if (running) {
00133                 running = false;
00134                 east.simulationStop();
00135             } else{
00136                 running = true;
00137                 east.simulationStart();
00138             }
00139         }
00140     }
00141
00145     static class stepButtonActionListener implements ActionListener{
00146         @Override
00147         public void actionPerformed(ActionEvent event){
00148             String shape = (String) comboBox.getSelectedItem();
00149             if (shape.equals("square")){
00150                 east.shape = Table.Shapes.SQUARE;
00151             } else if(shape.equals("triangle")){
00152                 east.shape = Table.Shapes.TRIANGLE;
00153             } else{
00154                 System.out.println("Panic!");
00155             }
00156             readBS();
00157             if(running){
00158                 running = false;
00159                 east.simulationStop();
00160                 east.step();
00161                 east.repaint();
00162             } else{
00163                 east.step();
00164                 east.repaint();
00165             }
00166         }
00167     }
00168
00172     public static void readBS(){
00173         String[] bsString = bsTextField.getText().split("/");
00174         HashMap<String, ArrayList<Integer» bs = new HashMap<>();
00175         bs.put("born", new ArrayList<>());
00176         String[] s = bsString[0].split("");
00177         for (int i = 0; i < s.length; i++) {
00178             bs.get("born").add(Integer.parseInt(s[i]));
00179         }
00180         bs.put("survive", new ArrayList<>());
00181         s = bsString[1].split("");
00182         for (int i = 0; i < s.length; i++) {
00183             bs.get("survive").add(Integer.parseInt(s[i]));
00184         }
00185         Table.setRules(bs);
00186     }
```

```
00187
00188     public static Table getEast() {
00189         return east;
00190     }
00191 }
```

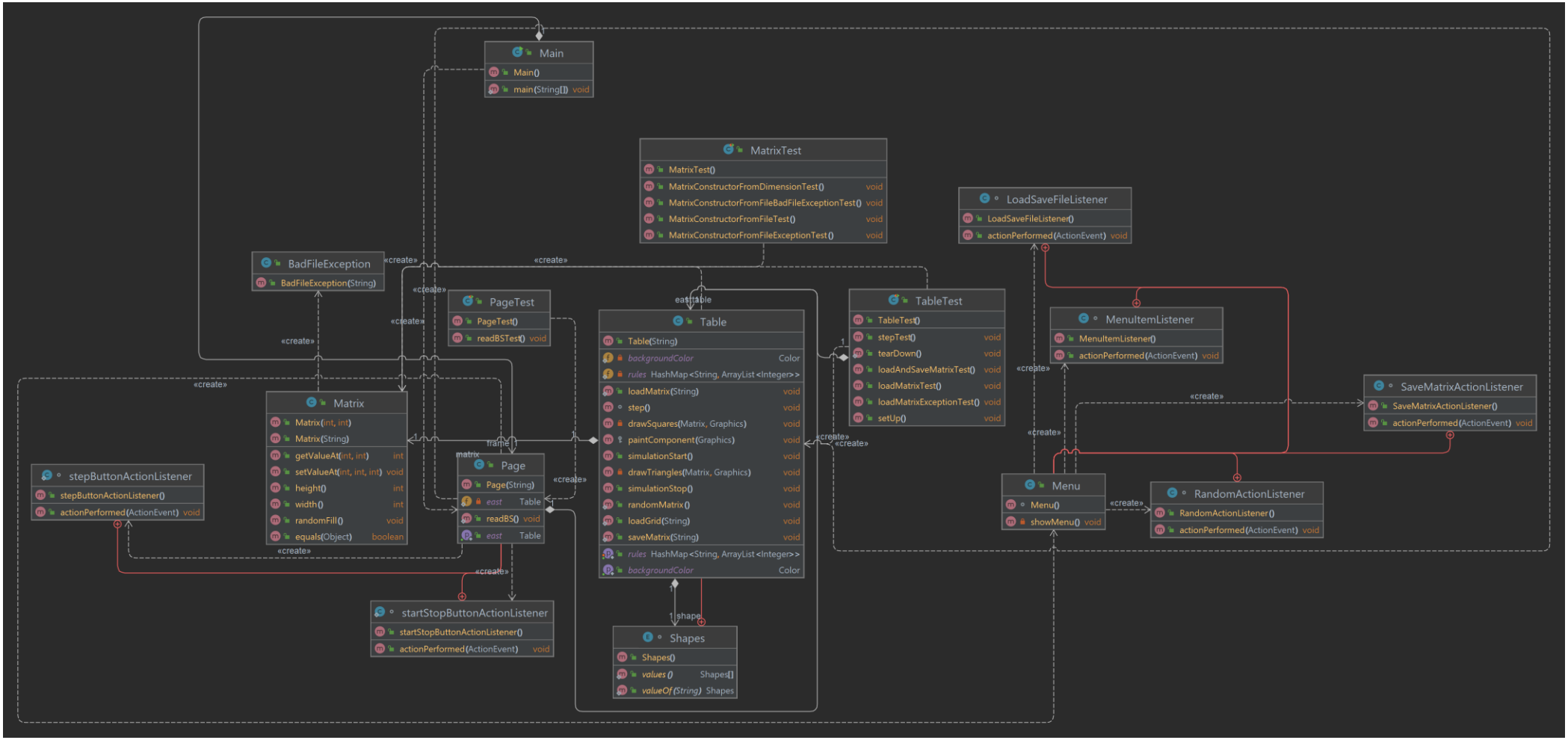
5.6 Table.java

```
00001 package GameOfLife;
00002
00003 import javax.swing.*;
00004 import java.awt.*;
00005 import java.io.*;
00006 import java.util.ArrayList;
00007 import java.util.HashMap;
00008
00014 public class Table extends JPanel {
00015     private int generation = 0;
00016     //A kockák és négyzetek mögötti szín.
00017     private static Color backgroundColor = Color.white;
00018     //A kockák és négyzetek színe, valamint a generáció szöveg színe.
00019     private static Color foregroundColor = Color.black;
00020     //Éppen milyen formákat akarunk kijelezni. Ez egy enum, amely jelenleg 2 értéket tud felvenni.
00021     Shapes shape = Shapes.SQUARE;
00022     //A pálya, a 20,20 az csak beállított érték, jelentése nincs, de kell, hogy legyen valami értéke a
      szélességnek/magasságnak.
00023     static Matrix matrix = new Matrix(20, 20);
00024     //A játékszabályt tároló. A HashMap-ben 2 kulcs van, a "born" és a "survive", mindegyikhez
      tartozik egy ArrayList
00025     // mint value amiben azok at értékek vannak, ahány szomszédja ha a cellának van, a cella
      megszóletik (bron) vagy túlél (survive).
00026     private static HashMap<String, ArrayList<Integer» rules = new HashMap<>();
00027     static {
00028         ArrayList<Integer> born = new ArrayList<>();
00029         born.add(3);
00030         rules.put("born", born);
00031         ArrayList<Integer> survive = new ArrayList<>();
00032         survive.add(2);
00033         survive.add(3);
00034         rules.put("survive", survive);
00035     }
00036     // A timer, ami 50 ms-ént lépteti a szimulációt, fur folyamatosan.
00037     private Timer timer = new Timer(50, e1 -> {
00038         step();
00039         this.repaint();
00040     });
00041
00046     public Table(String file) {
00047         try {
00048             loadGrid(file);
00049         } catch (IOException e) {
00050             System.out.println("Nem megfelelő file");
00051         } catch (BadFileException e){
00052             System.out.println("Nem sikerült olvasni a file-t a mx nem változott.");
00053         }
00054         backgroundColor = Color.CYAN;
00055     }
00056
00062     @Override
00063     protected void paintComponent(Graphics g) {
00064         g.setColor(foregroundColor);
00065         super.paintComponent(g);
00066         g.drawString("Generation: " + generation, 10, 15);
00067
00068         if(shape == Shapes.TRIANGLE){
00069             drawTriangles(matrix, g);
00070         } else {
00071             drawSquares(matrix, g);
00072         }
00073     }
00074
00081     private void drawSquares(Matrix matrix, Graphics g){
00082         for (int i =0; i<matrix.width(); i++){
00083             for (int j = 0; j < matrix.height(); j++) {
00084                 int y = i*10+20;
00085                 if (matrix.getValueAt(i, j) == 0) {
00086                     g.clearRect(j * 10 +10, y, 10, 10);
00087                     g.setColor(backgroundColor);
00088                     g.fillRect(j * 10+10, y, 10, 10);
00089                     g.setColor(foregroundColor);
00090                     g.drawRect(j * 10+10, y, 10, 10);
00091                 } else {
00092                     g.setColor(foregroundColor);
```

```
00093                     g.fillRect(j * 10+10, y, 10, 10);
00094                 }
00095             }
00096         }
00097     }
00098
00106     private  void drawTriangles(Matrix matrix, Graphics g) {
00107         int x = 10;
00108         for (int i = 0; i < matrix.width(); i++) {
00109             for (int j = 0; j < matrix.height(); j++) {
00110                 int y = i * 10 + 20;
00111                 int[] xpoints;
00112                 int[] ypoints;
00113                 if (j % 2 == 1) {
00114                     xpoints = new int[]{x + 5, x, x + 10};
00115                     ypoints = new int[]{y, y + 10, y + 10};
00116                 } else {
00117                     xpoints = new int[]{x, x + 10, x + 5};
00118                     ypoints = new int[]{y, y, y + 10};
00119                 }
00120                 Polygon triangle = new Polygon(xpoints, ypoints, 3);
00121                 if (matrix.getValueAt(i, j) == 0) {
00122                     g.setColor(foregroundColor);
00123                     g.drawPolygon(triangle);
00124                 } else {
00125                     g.setColor(foregroundColor);
00126                     g.fillPolygon(triangle);
00127                 }
00128                 x = x + 5;
00129             }
00130             x = 10;
00131         }
00132     }
00133
00139     void step(){
00140         int M = matrix.width();
00141         int N = matrix.height();
00142         Matrix future = new Matrix(M, N);
00143
00144         for (int l = 0; l < M; l++)
00145         {
00146             for (int m = 0; m < N; m++)
00147             {
00148                 // finding no Of Neighbours that are alive
00149                 int aliveNeighbours = 0;
00150                 for (int i = -1; i <= 1; i++)
00151                     for (int j = -1; j <= 1; j++)
00152                         if ((l+i>=0 && l+i<M) && (m+j>=0 && m+j<N))
00153                             aliveNeighbours += matrix.getValueAt(l+i, m+j);
00154
00155                 // The cell needs to be subtracted from
00156                 // its neighbours as it was counted before
00157                 aliveNeighbours -= matrix.getValueAt(l, m);
00158
00159                 //A cell dies, bc to lonely or overpopulated
00160                 if ((matrix.getValueAt(l, m) == 1) &&
    !(rules.get("survive").contains(aliveNeighbours)))
00161                     future.setValueAt(l, m, 0);
00162
00163                     // A new cell is born
00164                 else if ((matrix.getValueAt(l, m) == 0) &&
    (rules.get("born").contains(aliveNeighbours)))
00165                     future.setValueAt(l, m, 1);
00166
00167                     // Remains the same
00168                 else
00169                     future.setValueAt(l, m, matrix.getValueAt(l, m));
00170             }
00171         }
00172         matrix = future;
00173         generation++;
00174     }
00175
00184     public static void loadGrid(String file) throws IOException, BadFileException {
00185         matrix = new Matrix(file);
00186     }
00187
00191     public void simulationStop(){
00192         timer.stop();
00193     }
00194
00198     public void simulationStart(){
00199         timer.start();
00200     }
00201
00205     enum Shapes{
00206         SQUARE, TRIANGLE
```

```
00207     }
00208
00209     public static Color getBackgroundColor() {
00210         return backgroundColor;
00211     }
00212
00218     static public void saveMatrix(String file) throws IOException{
00219         FileOutputStream f = new FileOutputStream(file);
00220         ObjectOutputStream out = new ObjectOutputStream(f);
00221         out.writeObject(matrix);
00222         out.close();
00223
00224     }
00225
00232     static public void loadMatrix(String file) throws IOException, ClassNotFoundException{
00233         FileInputStream f = new FileInputStream(file);
00234         ObjectInputStream in = new ObjectInputStream(f);
00235         matrix = (Matrix) in.readObject();
00236         in.close();
00237     }
00238
00242     public static void randomMatrix(){
00243         matrix.randomFill();
00244     }
00245
00250     public static HashMap<String, ArrayList<Integer» getRules() {
00251         return rules;
00252     }
00253
00258     public static void setRules(HashMap<String, ArrayList<Integer» rules) {
00259         Table.rules = rules;
00260     }
00261 }
```

Index


Main
Main()
main(String[]) void
MatrixTest
MatrixTest()
MatrixConstructorFromDimensionTest() void
MatrixConstructorFromFileBadFileExceptionTest() void
MatrixConstructorFromFileTest() void
MatrixConstructorFromFileExceptionTest() void
LoadSaveFileListener
LoadSaveFileListener()
actionPerformed(ActionEvent) void
BadFileException
BadFileException(String)
PageTest
PageTest()
readBSTest() void
Table
Table(String)
backgroundColor Color
rules HashMap<String, ArrayList<Integer>>
loadMatrix(String) void
step() void
drawSquares(Matrix, Graphics) void
paintComponent(Graphics) void
simulationStart() void
drawTriangles(Matrix, Graphics) void
simulationStop() void
randomMatrix() void
loadGrid(String) void
saveMatrix(String) void
rules HashMap<String, ArrayList<Integer>>
backgroundColor Color
TableTest
TableTest()
stepTest() void
tearDown() void
loadAndSaveMatrixTest() void
loadMatrixTest() void
loadMatrixExceptionTest() void
setUp() void
MenuItemListener
MenuItemListener()
actionPerformed(ActionEvent) void
SaveMatrixActionListener
SaveMatrixActionListener()
actionPerformed(ActionEvent) void
Matrix
Matrix(int, int)
Matrix(String)
getValueAt(int, int) int
setValueAt(int, int, int) void
height() int
width() int
randomFill() void
equals(Object) boolean
Page
Page(String)
east Table
readBS() void
east Table
stepButtonActionListener
stepButtonActionListener()
actionPerformed(ActionEvent) void
Menu
Menu()
showMenu() void
RandomActionListener
RandomActionListener()
actionPerformed(ActionEvent) void
startStopButtonActionListener
startStopButtonActionListener()
actionPerformed(ActionEvent) void
Shapes
Shapes()
values() Shapes[]
valueOf(String) Shapes
«create»
eastTable
matrix
frame
1
shapes