# II. Beadandó

## Habzda Fruzsina (XNUHTE)

Tartalomjegyzék

[Feladat 3](#_Toc529561315)

[Tervezés 3](#_Toc529561318)

[Specifikáció 3](#_Toc529561319)

[Osztály diagram 4](#_Toc529561320)

[Részletezett UML diagrammok 4](#_Toc529561321)

[Megvalósítás 6](#_Toc529561322)

[Főprogram 6](#_Toc529561323)

[GameLogic 6](#_Toc529561324)

[GameBoard 11](#_Toc529561325)

[FieldButtonActionListener 12](#_Toc529561326)

[Tesztelés 13](#_Toc529561327)

[Elvégzett tesztesetek 13](#_Toc529561328)

# Feladat

## *4. feladat: Kiszúrós amőba*

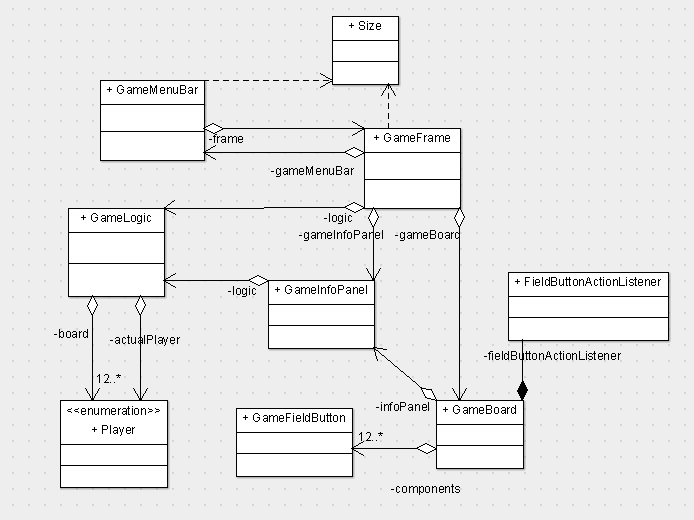
## *Készítsünk programot, amellyel a közismert amőba játék következő változatát játszhatjuk. Adott egy n × n-es tábla, amelyen a két játékos felváltva X, illetve O jeleket helyez el. Csak olyan mezőre tehetünk jelet, amely még üres. A játék akkor ér véget, ha betelik a tábla (döntetlen), vagy valamelyik játékos kirak 5 egymással szomszédos jelet vízszintesen, függőlegesen vagy átlósan. A program minden lépésnél jelezze, hogy melyik játékos következik, és a tábla egy üres mezőjét kijelölve helyezhessük el a megfelelő jelet. A kiszúrás a játékban az, hogy ha egy játékos eléri a 3 egymással szomszédos jelet, akkor a program automatikusan törli egy jelét egy véletlenszerűen kiválasztott pozícióról (nem biztos, hogy a hármasból), ha pedig 4 egymással szomszédos jelet ér el, akkor pedig kettőt. A program biztosítson lehetőséget új játék kezdésére a táblaméret megadásával (6×6, 10×10, 14×14), és ismerje fel, ha vége a játéknak. Ekkor jelenítse meg, hogy melyik játékos győzött (ha nem lett döntetlen), majd kezdjen automatikusan új játékot.*

# Tervezés

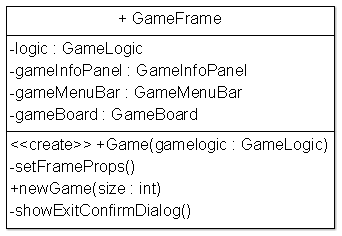
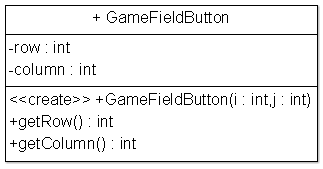
## Specifikáció

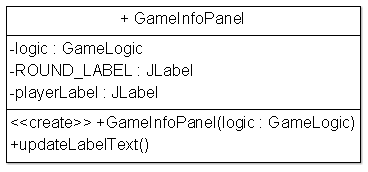
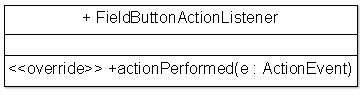
A game mappán belül két package lett létrehozva. A gui, mely tartalmazza a GameFrame.java, GameBoard.java, GameFieldButton.java, GameInfoPanel.java, GameMenuBar.java állományokat, és a logic mely tartalmazza a GameLogic.java, Player.java, Size.java állományokat. A GUI alapja a GameFrame, amit JFrame-ből származtattam. Tartalmaz egy menüt (GameMenuBar, JMenuBar-ból), egy információs panelt (GameInfoPanel, JPanel-ből), és a játék táblát (GameBoard). A menüből új játék indítható, választható táblamérettel. Az információs panel kiírja a soron lévő játékos (logic.actualPlayer) jelét. (Mindig X kezdi a játékot, a játék 6\*6-os táblával indul.) A játék táblán gombok (GameFieldButton) vannak elhelyezve, melyek először üres mezőket reprezentálnak, majd ha a soron következő játékos rákattint, annak a jelét jelenítik meg. Két féle játékossal lehet játszani (Player.X, Player.Y). A táblaméret konstans változóit a Size nevű osztályban tárolom (6 féle van: SMALL, MEDUIM, LARGE). A játékhoz szükséges számításokat a GameLogic osztály végzi. Mindegyik osztályból egyetlen objektum van létrehozva.

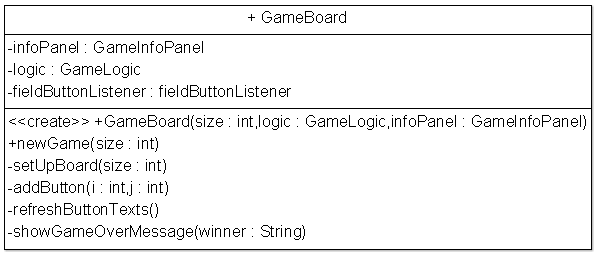
## Osztály diagram

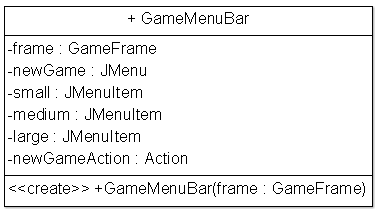
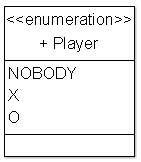
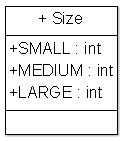


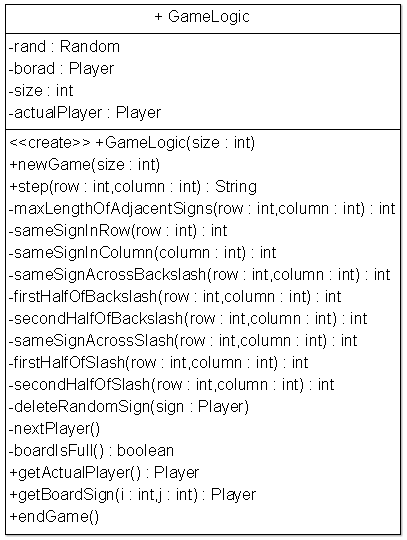
### Részletezett UML diagrammok





# Megvalósítás

### Főprogram

A főprogram (Boot) létrehoz egy új GameFrame-et, és láthatóvá teszi.

java.awt.EventQueue.*invokeLater*(new Runnable() {  
 public void run() {  
 new GameFrame(new GameLogic(Size.*SMALL*)).setVisible(true);  
 }  
});

### GameLogic

A játék belső számításait végző osztály a GameLogic.

#### newGame:

public void newGame(int size){  
 actualPlayer = Player.*X*;  
 this.size = size;  
 board = new Player[size][size];  
 for (int row = 0; row < size; ++row) {  
 for (int column = 0; column < size; ++column) {  
 board[row][column] = Player.*NOBODY*;  
 }  
 }  
}

#### Step:

Ez a metódus végzi a „lépést”. Először is megvizsgálja, hogy egyáltalán rakhat-e a játékos az adott „mezőre”. Ha igen, rak. Ebben az esetben megnézni, hogy kell e törölni, illetve ha igen, hányat, és töröl. Ha sikeresen elvégezte a lépést, vált a következő játékosra. Ha kijött az 5 ugyanolyan egymás mellett, az adott játékost visszaadja, és nem lép a következőre. Ha a tábla betelt „döntetlent” ad vissza.

public String step(int row, int column) {  
 if (board[row][column] == Player.*NOBODY*){  
 board[row][column] = actualPlayer;  
 if (maxLengthOfAdjacentSigns(row, column) >= 5) {  
 return String.*valueOf*(actualPlayer);  
 } else if (maxLengthOfAdjacentSigns(row, column) == 4) {  
 deleteRandomSign(actualPlayer);  
 deleteRandomSign(actualPlayer);  
 nextPlayer();  
 return "";  
 } else if (maxLengthOfAdjacentSigns(row, column) == 3) {  
 deleteRandomSign(actualPlayer);  
 nextPlayer();  
 return "";  
 } else if (boardIsFull()) {  
 return " - . A meccs döntetlen.";  
 }  
 nextPlayer();  
 }  
 return "";  
}

#### maxLengthOfAdjacentSigns:

Kiválasztja a maximumot a sameSignInColumn, sameSignInRow, sameSignAcrossBackslash, sameSignAcrossSlash metódusok visszatérési értékei közül.

private int maxLengthOfAdjacentSigns(int row, int column){  
 int max = sameSignInRow(row);  
 if (max < sameSignInColumn(column)){  
 max = sameSignInColumn(column);  
 }  
 if (max < sameSignAcrossBackslash(row, column)){  
 max = sameSignAcrossBackslash(row, column);  
 }  
 if (max < sameSignAcrossSlash(row, column)){  
 max = sameSignAcrossSlash(row, column);  
 }  
 return max;  
}

#### sameSignInColumn:

Végig iterál azon az oszlopon, mely meg lett adva paraméterként, és visszaadja a leghosszabb olyan összefüggő oszlop hosszát, mely az aktuális játékos jeleit tartalmazza.

private int sameSignInColumn(final int column){  
 int max = 0;  
 for (int i=0; i<size; i++){  
 int sum = 0;  
 while (i<size && board[i][column]==actualPlayer){  
 sum++;  
 i++;  
 }  
 if (max<sum){  
 max = sum;  
 }  
 }  
 return max;  
}

#### sameSignInRow:

Végig iterál azon a soron, mely meg lett adva paraméterként, és visszaadja a leghosszabb olyan összefüggő sor hosszát, mely az aktuális játékos jeleit tartalmazza.

private int sameSignInRow(final int row){  
 int max = 0;  
 for (int i=0; i<size; i++){  
 int sum = 0;  
 while (i<size && board[row][i]==actualPlayer){  
 sum++;  
 i++;  
 }  
 if (max<sum){  
 max = sum;  
 }  
 }  
 return max;  
}

#### sameSignAcrossBackslash:

Összeadja a fordított perjel irányban (bal felső -> jobb alsó irány) megszámolt elemek első és második szakaszát, és hozzáadja a középső elemet. Összegezve tehát a fordított perjel irányú azonos jelek számával tér vissza.

private int sameSignAcrossBackslash(int row, int column){  
 return (1 + firstHalfOfBackslash(row,column) + secondHalfOfBackslash(row,column));  
}

#### firstHalfOfBackslash:

A megadott elem pozíciójától (paraméterben megadott sor, oszlop) jobbra lefelé számolja meg az azonos jeleket.

private int firstHalfOfBackslash(int row, int column){  
 int sum = 0;  
 int i=row+1;  
 int j=column+1;  
 while((i < size && j < size) && (i >= 0 && j >= 0) && board[i][j]==actualPlayer)  
 {  
 i++;  
 j++;  
 sum++;  
 }  
 return sum;  
}

#### secondHalfOfBackslash:

A megadott elem pozíciójától (paraméterben megadott sor, oszlop) balra felfelé számolja meg az azonos jeleket.

private int secondHalfOfBackslash(int row, int column){  
 int sum = 0;  
 int i=row-1;  
 int j=column-1;  
 while((i >= 0 && j >= 0) && (i < size && j < size) && board[i][j]==actualPlayer)  
 {  
 i--;  
 j--;  
 sum++;  
 }  
 return sum;  
}

#### sameSignAcrossSlash:

Összeadja a perjel irányban (jobb felső -> bal alsó irány) megszámolt elemek első és második szakaszát, és hozzáadja a középső elemet. Összegezve tehát a perjel irányú azonos jelek számával tér vissza.

private int sameSignAcrossSlash(int row, int column){  
 return (1 + firstHalfOfSlash(row,column) + secondHalfOfSlash(row,column));  
}

#### firstHalfOfSlash:

A megadott elem pozíciójától (paraméterben megadott sor, oszlop) balra lefelé számolja meg az azonos jeleket.

private int firstHalfOfSlash(int row, int column){  
 int sum = 0;  
 int i=row+1;  
 int j=column-1;  
 while((i >= 0 && j < size) && (i < size && j >= 0) && board[i][j]==actualPlayer)  
 {  
 i++;  
 j--;  
 sum++;  
 }  
 return sum;  
}

#### secondHalfOfSlash:

A megadott elem pozíciójától (paraméterben megadott sor, oszlop) jobbra felfelé számolja meg az azonos jeleket.

private int secondHalfOfSlash(int row, int column){  
 int sum = 0;  
 int i=row-1;  
 int j=column+1;  
 while((i >= 0 && j < size) && (i < size && j >= 0) && board[i][j]==actualPlayer)  
 {  
 i--;  
 j++;  
 sum++;  
 }  
 return sum;  
}

#### deleteRandomSign:

private void deleteRandomSign(Player sign){  
 int randomRow = rand.nextInt(size);  
 int randomColumn = rand.nextInt(size);  
 while(board[randomRow][randomColumn] != sign){  
 randomRow = rand.nextInt(size);  
 randomColumn = rand.nextInt(size);  
 }  
 board[randomRow][randomColumn] = Player.*NOBODY*;  
}

#### boardIsFull:

Végignézi a tábla elemeit, és ha nem talál NOBODY játékost (tehát olyan helyet, ahova lehetne rakni), igaz értékkel tér vissza (ami azt jelenti, hogy a tábla megtelt).

private boolean boardIsFull(){  
 for(int row=0; row<size; row++){  
 for(int column=0; column<size; column++){  
 if(board[row][column]==Player.*NOBODY*){  
 return false;  
 }  
 }  
 }  
 return true;  
}

### GameBoard

A GameBoard végzi a játéktábla megjelenítését. JPanel-ből lett származtatva.

#### newGame:

public void newGame(int size){  
 removeAll();  
 setUpBoard(size);  
 infoPanel.updateLabelText();  
}

#### setUpBoard:

Elkészíti a táblát.

private void setUpBoard(int size) {  
 setLayout(new GridLayout(size, size));  
 for (int i = 0; i < size; ++i) {  
 for (int j = 0; j < size; ++j) {  
 addButton(i, j);  
 }  
 }  
}

#### addButton:

Létrehozza a gombokat, pozícióval, mérettel, és hozzáadja a táblához.

private void addButton(int i, int j){  
 final GameFieldButton button = new GameFieldButton(i, j);  
 button.addActionListener(fieldButtonActionListener);  
 button.setPreferredSize(new Dimension(45, 45));  
 add(button);  
}

#### refreshButtonTexts:

A táblán elhelyezett gombok feliratát változtatja meg a GameLogic által kezelt board nevű tömb alapján.

private void refreshButtonTexts(){  
 for(Component component : getComponents()){  
 GameFieldButton button = (GameFieldButton) component;  
 int row = button.getRow();  
 int column = button.getColumn();  
 if (logic.getBoardSign(row, column) != Player.*NOBODY*){  
 button.setText(String.*valueOf*(logic.getBoardSign(row, column)));  
 } else {  
 button.setText("");  
 }  
 }  
}

#### showGameOverMesssage:

private void showGameOverMessage(String winner) {  
 JOptionPane.*showMessageDialog*(this, "A nyertes: " + winner);  
 logic.endGame();  
 refreshButtonTexts();  
}

### FieldButtonActionListener

#### actionPerformed:

Elvégzi a lépést az adott gombon („mezőn”), frissíti az információ panelt, és ha a visszakapott érték egy játékos vagy döntetlen, megjeleníti a játék vége üzenetet.

@Override  
public void actionPerformed(ActionEvent e) {  
 GameFieldButton button = (GameFieldButton) e.getSource();  
 String winner = logic.step(button.getRow(), button.getColumn());  
 refreshButtonTexts();  
 infoPanel.updateLabelText();  
 if (winner != ""){  
 showGameOverMessage(winner);  
 }  
}

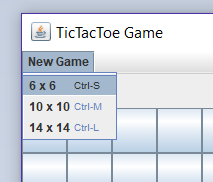
# Tesztelés

## Elvégzett tesztesetek

* Menügombok működése
  + új játék indításánál jó méret töltődik be, lenullázódik a tábla (logic.*newGame()*, gameBoard.*newGame()*) (1.-5.)
* step() metódus tesztelése
  + akkor változik meg a pálya, ha a játékos nem lefoglalt mezőre rak, különben nem (6.)
  + ha 5 ugyanolyan keletkezik egy sorban, oszlopban, átlósan, akkor a jelenlegi játékos nyer (9.-10.)
  + ha 4 ugyanolyan keletkezik egy sorban, oszlopban, átlósan, akkor 2 jele a pályáról törlődik (*deleteRandomSign()*) (8.)
  + ha 3 ugyanolyan keletkezik egy sorban, oszlopban, átlósan, akkor 1 jele a pályáról törlődik (*deleteRandomSign()*) (7.)
  + ha betelt a pálya (nincs lehetőség sehova se rakni, *boardIsFull()*), befejeződik a játék
  + akkor változik az aktuális játékos, ha az előző tudott rakni (*nextPlayer()*) (6.-8.)
  + jól határozza meg az oszlopokban, sorokban, és átlósan az egymás melletti ugyanolyan jelek számát - *maxLengthOfAdjacentSigns()* rekurzívan (*sameSignInRow() 9., sameSignInColumn() 8., sameSignAcrossSlash() 10. , sameSignAcrossBackslash() 7.*)
* FieldButtonActionListener :: actionListener() tesztelése
  + helyesen változnak meg a gombok feliratai, ha törlés volt eltűnik, ha lépés volt, „lerak” (*refreshButtonTexts()*) (6.-8.)
  + megváltozik az aktuálisan soron következő szerint az infoPanel (*updateLabel()*) (6.-10.)
  + *showGameOverMessage()* (9.-10.)
* kilépés megerősítés ablak működése (11.)

*(Megjegyzés: a kék számok a teszt sorszámára utalnak.)*

A menü:



##### Játék megnyitása

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=6  actualPlayer=X  board=  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  } |  |

##### Új játék kezdése (6\*6)

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=6  actualPlayer=X  board=  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  } | C:\Users\Fruzsina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\6x6.png |

##### Új játék indítása (10\*10)

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=10  actualPlayer=X  board=  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  } |  |

##### Új játék kezdése (14\*14)

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=14  actualPlayer=X  board=  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  - - - - - - - - - - - - - -  } |  |

##### Új játék kezdése nem üres tábla esetén (10\*10)

|  |  |  |
| --- | --- | --- |
|  | Logic | GUI |
| Előtte | GameLogic{  size=6  actualPlayer=O  board=  - - - - - X  - X O - - -  - - - X - -  - - - - X -  - O - O - -  - - - - - -  } |  |
| Utána | GameLogic{  size=6  actualPlayer=X  board=  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  - - - - - -  } | C:\Users\Fruzsina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\6x6.png |

##### Ha nem „rossz” helyre rak az aktuális játékos, megint ő következik

|  |  |  |
| --- | --- | --- |
|  | Logic | GUI |
| Előtte | GameLogic{  size=6  actualPlayer=O  board=  - - - - - X  - X O - - -  - - - X - -  - - - - X -  - O - O - -  - - - - - -  } | C:\Users\Fruzsina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\újjátékelőtt.png |
| Utána | GameLogic{  size=10  actualPlayer=X  board=  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  } | C:\Users\Fruzsina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\10x10.png |

##### 3 ugyanolyan jel egymás mellett

|  |  |  |
| --- | --- | --- |
|  | Logic | GUI |
| Előtte | GameLogic{  size=6  actualPlayer=X  board=  - - - O - -  - O - X - X  - - - - - -  - X - - O -  - - - - - -  - - X - O -  } |  |
| Utána | GameLogic{  size=6  actualPlayer=O  board=  - - - O - -  - O - X - -  - - X - - -  - X - - O -  - - - - - -  - - X - O -  } |  |

##### 4 ugyanolyan jel egymás mellett

|  |  |  |
| --- | --- | --- |
|  | Logic | GUI |
| Előtte | GameLogic{  size=10  actualPlayer=O  board=  - - X - - - - - - -  - - - - - X - - X -  O - - - - - - - - -  O - - - - - - - X -  - - - - X - - - - -  O - - - - - - - - -  - - - - - - - - - -  - - - - O - - - - -  - - - - - - - - - -  - - - - - - - - - -  } |  |
| Utána | GameLogic{  size=10  actualPlayer=X  board=  - - X - - - - - - -  - - - - - X - - X -  O - - - - - - - - -  O - - - - - - - X -  - - - - X - - - - -  O - - - - - - - - -  - - - - - - - - - -  - - - - O - - - - -  - - - - - - - - - -  - - - - - - - - - -  } |  |

(Megjegyzés: 2-t törölt, ez esetben az egyik ilyen elem, az lett, ami most lett lerakva, szóval a véletlenszerűen kiválasztott elem, lehet az imént lerakott is.)

##### X nyert, üzenet

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=6  actualPlayer=X  board=  X X X X X -  - - - - - O  - O - - - -  - - O - O -  - O - - - -  - - - X - -  } |  |

##### O nyert, üzenet

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=6  actualPlayer=O  board=  - X - - O -  - - - O X -  - - O - - -  - O - - X -  O - - - - -  - - X - - -  } |  |

##### Játék vége után (OK gomb lenyomása után) új játék kezdése

|  |  |  |
| --- | --- | --- |
|  | Logic | GUI |
| Előtte | GameLogic{  size=10  actualPlayer=X  board=  - - - - - - - - - -  - O X - O O O - - -  O - X - X O X O - -  - O X O O X X - - -  - - - X O X O X - -  - - X O X - O O X -  - O O O X O X X - -  - - X - X - O - - -  - X O - O - - - - -  - - - - - - - - - -  } |  |
| Utána | GameLogic{  size=10  actualPlayer=X  board=  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  - - - - - - - - - -  } |  |

(Megjegyzés: Az automatikus újrakezdés mérettartó.)

##### kilépés

|  |  |
| --- | --- |
| Logic | GUI |
| GameLogic{  size=6  actualPlayer=O  board=  O O X X - O  - - - O X -  - X - - - -  O O - - X X  O - O X - X  X - - O - O  } | C:\Users\Fruzsina\AppData\Local\Microsoft\Windows\INetCache\Content.Word\bezárás.png |