

# Aknakereső játék – specifikáció

## A PROGRAM CÉLJA

---

Maga a program célja a már világhírű és közismert Aknakereső (Minesweeper) játék egyszemélyes verziójának elkészítése Java nyelven, továbbá a szabályrendszerének betartása, és dicsőséglista vezetése az addig eltárolt eredményekről.

## A PROGRAM HASZNÁLATA

---

A program elindítása után a felhasználó választhat az előre elkészített nehézségi fokozatok közül egy szám beírásával, vagy egyénileg is megszabhatja azt a billentyűzet segítségével. A nehézség függ a tábla méretétől és a mezők alatt rejtett akna számától is. A következő táblaméretek, aknamennyiség beállítására van lehetőség:

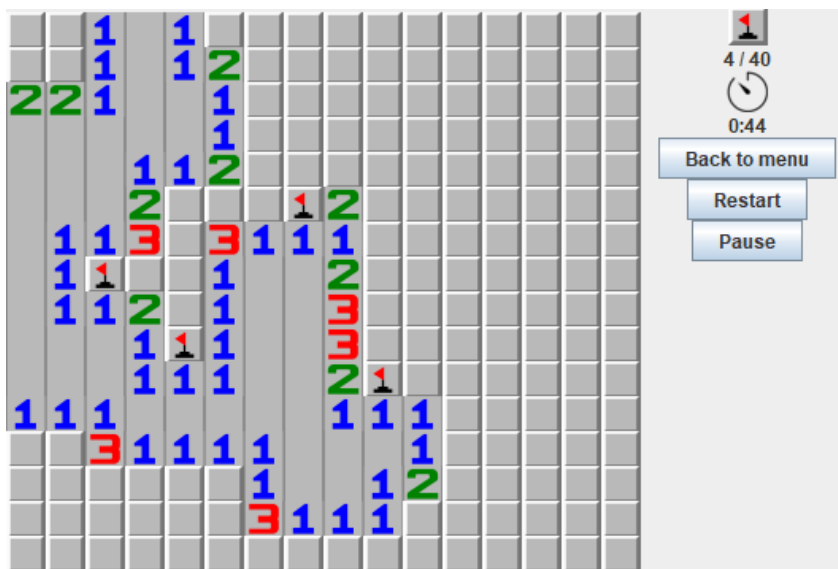
- (1) Kezdő: 8x8 mező 10 aknával
- (2) Haladó: 16x16 mező 40 aknával
- (3) Mester: 30x16 mező 99 aknával
- (4) Egyéni

Az „Egyéni” beállítás kiválasztása esetén a program megkérdezi, hogy hány sorból és oszlopból álljon a tábla, továbbá az akna összes mezőtől vett százalékát, ahol a minimum 1% és a maximum 96%.







## JÁTÉKSZABÁLYOK, A JÁTÉK MENETE, USER MANUAL

---

A pálya kinézete



A játéknézetben az egyforma, négyzetes mezőkre osztott téglalap alakú pályával a szembesül a felhasználó. A mezőknek (vagy celláknak) különböző állapotaik is vannak:

- **fedett** (alapállás): 
- **szomszédos aknával rendelkező aknamentes feltárt**, melyet cellán belüli szám jelöl (ez lehet egytől nyolcig): 
- **aknamentes feltárt**: 
- **zászlós** (játékos szerint akna van alatta): 
- **robbanó aknás feltárt**: 
- **korábban zászlóval jelzett aknás feltárt**: 

A jobb egérgombbal zászlót tehetünk egy adott mezőre, amely csak segítséget nyújt a játékmenet során a felhasználónak az aknák észben tartásához, ugyanis zászlók lehelyezése egyáltalán nem szükséges feltétele a győzelemnek. A játék során maximum annyi zászlót helyezhetünk le, amennyi akna van, továbbá letételük után fel is szedhetjük őket.

A bal egérgomb segítségével rákattinthatunk egyes cellákra, majd megtudhatjuk, hogy mit rejtenek feltárt állapotukban. Ha egy aknamentes mezőt találtunk, azonban a közvetlen környezetében található akna, akkor a mezőn belül megjelenik egy szám (minimum: 1, maximum 8), amely a cella melletti aknák számát jelzi.

Másik lehetőség, ha olyan aknamentes mezőre bukkanunk, amelynek a környezete sem tartalmaz aknát, ekkor az adott mezővel határos cellák mindegyike feltárul, továbbá az így feltáruló aknamentes „sziget” határos cellák is feltárulnak, ha nincs alattuk akna.

Ha a játékos mindkét egérgombbal kattint egyszerre egy olyan mezőre, amin egy szám áll, és az adott mező közvetlen környezetében pontosan a számmal megegyező mennyiségű zászló van lerakva, akkor az adott mező körüli aknamentes mezők felderítődnek. A zászlók helyzete itt nagyon fontos ugyanis, ha a felhasználó rosszul döntött és a zászlók nem a tényleges aknákon helyezkednek el a számmal jelölt mező körül, akkor felderítődik az aknás mező, tehát a felhasználó veszített.

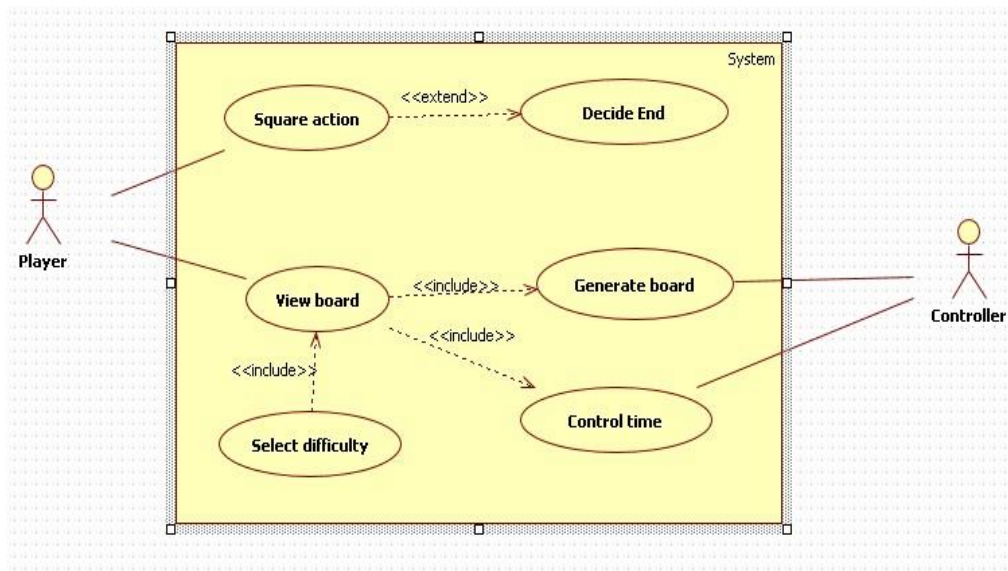
A játék során a felhasználó nyomon követheti a kezdéstől mért időt másodpercben, a zászlók számát, továbbá a maradék aknák számát, amely az eredeti mennyiségből letett zászlónként eggyel csökken. A cél egyértelműen az összes aknamentes mező megtalálása a legrövidebb idő alatt.

Nyerni csak úgy lehet, ha logikai úton (és persze kellő szerencsével) megtaláljuk az összes aknamentes cellát, anélkül, hogy rákattintanánk egy aknára ugyanis, ha ez bekövetkezne, akkor az adott mező „felrobban”, feltárul a többi aknát fedő mező, és vége a játékmenetnek, miután a felhasználó kiléphet, megtekintheti a dicsőséglistát az eddigi legjobb eredményekkel, vagy új játékmenetet indíthat. Abban az esetben, ha a felhasználó megtalálta az összes aknamentes mezőt, akkor is véget ér a játék, a felhasználó szembesül az eltelt idővel, és begépelheti a becenevét, amely később megjelenhet a dicsőséglistán.

A dicsőséglista ablakban a legjobb idők jelennek meg, adott nehézségi fokozatonként csoportosítva, amit a felhasználó szintén megválaszthat-

## A JÁTÉKMENET USE-CASE LEÍRÁSA

### USE-CASE DIAGRAM



### USE-CASE LEÍRÁSOK

Cím	Square action
Leírás	A játékos a tábla egy cellájával lép interakcióba.
Aktorok	Player
Főforgatókönyv	1. A játékos bal, jobb vagy egyszerre mindkét egérgombbal kattint egy cellára.
Alternatív forgatókönyv	1.A.1. A játékos egy olyan mezőre kattint bal gombbal, amelynek csak közvetlen környezetében van akna, akkor felderítődik a mező és a benne lévő szám megmutatja, mennyi akna van a közelben.
Alternatív forgatókönyv	1.A.1.A.1. Ha a játékos az összes aknamentes mezőt megtalálta nyert, és begépeli a nevét a dicsőséglistába.
Alternatív forgatókönyv	1.B.1. A játékos egy olyan mezőre kattint bal gombbal, amelynek nincs a közvetlen környezetében akna, akkor az adott mezővel határos cellák mindegyike feltárul, továbbá az így feltáruló aknamentes „szigettel” határos cellák is feltárulnak, ha nincs alattuk akna.
Alternatív forgatókönyv	1.B.1.A.1. Ha a játékos az összes aknamentes mezőt megtalálta nyert, és begépeli a nevét a dicsőséglistába.
Alternatív forgatókönyv	1.C.1. A játékos veszít, ha olyan mezőre kattint, amin akna van.
Alternatív forgatókönyv	1.D.1. A játékos jobb egérgombbal kattint egy cellára, amivel letesz egy zászlót.
Alternatív forgatókönyv	1.D.1.A.1. A játékos egy olyan mezőre nyom jobb gombbal, amin már van zászló, akkor felveszi az adott zászlót.
Alternatív forgatókönyv	1.D.1.B.1. Ha játékos jobb egérgombbal kattint egy cellára, azonban már korábban letett annyi zászlót, ahány akna van, akkor nem történik semmi.

<b>Alternatív forgatókönyv</b>	<b>1.E.1.</b> Ha a játékos mindkét egérgombbal kattint egyszerre egy olyan mezőre, amin egy szám áll, és az adott mező közvetlen környezetében pontosan a számmal megegyező mennyiségű zászló van lerakva, akkor az adott mező körüli aknamentes mezők felderítődnek.
<b>Alternatív forgatókönyv</b>	<b>1.E.1.A.1.</b> Ha a játékos az összes aknamentes mezőt megtalálta nyert, és begépeli a nevét a dicsőséglistába.
<b>Alternatív forgatókönyv</b>	<b>1.E.1.B.1.</b> Ha a játékos nem az aknákra helyezte a zászlókat a számmal jelölt mező körül, akkor felderítődik az aknás mező, és a felhasználó vesz.

<b>Cím</b>	<b>Decide end</b>
<b>Leírás</b>	A játékos eldönti, hogy mit szeretne tenni, ha vége a játéknak.
<b>Aktorok</b>	Player
<b>Főforgatókönyv</b>	<b>1.</b> A játékos új játékot kezd, megtekinti a dicsőséglistát, vagy kilép.
<b>Alternatív forgatókönyv</b>	<b>1.A.1.</b> A játékos kategorizálhatja a dicsőséglistát nehézség szerint.

<b>Cím</b>	<b>View board</b>
<b>Leírás</b>	A játékos megtekinti a táblát.
<b>Aktorok</b>	Player
<b>Főforgatókönyv</b>	<b>1.</b> A rendszer kirajzolja a tábla aktuális állapotát. <b>2.</b> A játékos megtekinti a tábla aktuális állapotát.

<b>Cím</b>	<b>Select difficulty</b>
<b>Leírás</b>	A játékos kiválasztja a nehézséget.
<b>Aktorok</b>	Player
<b>Főforgatókönyv</b>	<b>1.</b> A játékos megválasztja a tábla nagyságát és aknák számát.
<b>Alternatív forgatókönyv</b>	<b>1.A.1.</b> A játékos az előre megadott nehézségi szintek közül választ.
<b>Alternatív forgatókönyv</b>	<b>1.B.1.</b> A játékos az egyéni nehézségi szintet választja és megadja, hogy mekkora legyen a tábla mérete és az aknák száma.

<b>Cím</b>	<b>Generate board</b>
<b>Leírás</b>	Inicializálja a táblának az állapotát.
<b>Aktorok</b>	Controller
<b>Főforgatókönyv</b>	<b>1.</b> A Controller véletlenszerűen elhelyezi a felhasználó által kiválasztott számú aknát a táblán, és az összes cellát fedettre állítja.

Cím	Control time
Leírás	Az idő változik a játék során.
Aktorok	Controller
Főforgatókönyv	1. A rendszer, a Controller másodpercenként növeli nulláról az eltelt idő értékét eggyel.
Alternatív forgatókönyv	1.A.1. Ha a játékos megtalálja az összes aknamentes mezőt a Controller megállítja az időszámlálót.
Alternatív forgatókönyv	1.B.1. Ha a játékos bal egérgommbal aknára kattint, a Controller megállítja az időszámlálót.

## A PROGRAM VÁZLATOS ISMERTETÉSE, A PROGRAM BE ÉS KIMENETEI

---

Maga az aknakereső játék egy Swing alapú GUI-val rendelkezik. A program indítása után a felhasználó a tényleges játékmenet előtt kiválaszthatja a táblaméretet és az aknaszámot a GUI és az előre elkészített menün keresztül. Az egyéni nehézség kiválasztása esetén a program bekéri az akna számát és a tábla méretét. A tábla állapota cellánként nyílvántartott a Java Collection API (Java gyűjtemény keretrendszer) segítségével.

A játék során a program először véletlenszerűen kisorsolja, hogy a kiválasztott számú akna hova kerüljön a táblán. Ezután a folyamat után a játékos interakcióba léphet a táblával: mezőket deríthet fel, zászlókat tehet le vagy vehet fel. A játékmenet közben a felhasználó visszaléphet a menübe, megállíthatja a játékot, vagy újra is kezdheti azt, a korábbi játékbeállításokkal. Attól függetlenül, hogy győzelemmel vagy vereséggel végződött a játékmenet a felhasználó egy “végső” képernyővel szembesül, ahol új játékot indíthat, kiléphet, meglekintheti a dicsőséglistát, vagy akár meg is adhatja a nevét, hogy felkerüljön a szóban forgó listára, ha sikerült nyernie.

A dicsőséglista fájlban eltárolása és visszatöltése szerializáció, vagyis a sorosítás segítségével valósul meg. A program a legjobb eredményekhez tartozó objektumokat konvertálja át bináris formába, mely bináris adat persze tárolható, és visszaolvasható fájlból, ha a lista megtekintésre kerül egy játék után.

A grafikus felületnek köszönhetően egészen kevés be és kimenettel rendelkezik a program. Bemeneti fájl a szerializált dicsőséglista („besttimes.dat”), illetve a különböző képfájlok, melyekkel a játékmenet során találkozhatunk és beolvasódnak a program indulása során. Felhasználó által megadott bemenet felléphet akkor, amikor a felhasználó egyéni beállításokat szeretne, és megadja a tábla paramétereit, illetve amikor megnyeri a játékot és begépelheti a nevét a dicsőséglistába.

Kimeneti fájl pedig egyedül a bináris formájú dicsőséglista fájl, melyre a következő indításkor feltétlenül szükség van.

# AZ OSZTÁLYOK ÉS METÓDUSAIK ISMERTETÉSE

---

Az osztályok és metódusaik ismertetése megtalálható a **forráskódban JavaDocs formában**, továbbá a tömörített fájlban csatolt DoxyGen és Graphviz által generált programozói dokumentációban is, mely megnyitható a .zipben csatolt index.html fájl megnyitásával. A generált dokumentáció ezen felül tartalmazza **minden függvénynek a hívási gráfját**, illetve a program **osztálydiagramját**, de az alábbiakban is ismertetésre kerülnek az osztályok a rövid leírásukkal:

## **minesweeper.higscore.HighscoreUI.BackButtonListener**

A visszagomb listenerje, amely elmenti az adatokat, ha visszalépünk a táblázatból, majd eltünteti az ablakot.

## **minesweeper.gameboard.Board**

Maga a tényleges aknakereső játéktérnek a kezelője. Tartalmazza a pálya paramétereit (oszlopok, sorok száma, aknák száma, vége van-e a játéknak, lerakott zászlók), továbbá a cellákat. Lekezel, ha egy cellával valamilyen interakció történik. Az osztály a teljes program gerince: az egész menürendszer-e köré épül fel. A játékmenet fő algoritmusait is magába foglalja: egy cella és szomszédainak rekurzív megjelenítése vagy egy cella szomszédainak meghatározása, majd a rajtuk végzett műveletek. Egy listában tárolja az összes cellát, hogy azok könnyen kezelhetők legyen. Lehetővé teszi, hogy egy cellát koordinátái, vagy akár a listában szereplő indexe alapján meghatározzunk, vagy műveletet végezzünk rajta.

## **minesweeper.gameboard.BoardTest**

A Board osztály tesztelésére szolgál. Főbb algoritmusokat tesztel, melyek elengedhetek egy játékfolyamat lebonyolításában.

## **minesweeper.gameboard.Cell**

Egy mezőt jelképez a táblán. Ebből az osztályból származó példányokat tartalmaz egy **Board** osztály. Maga az osztály lényegében szinte csak egy "adattároló", ami tartalmaz egy táblaegységhez tartozó paramétereket: van-e akna a cellán, hány akna van körülötte, van-e zászló rajta, hol helyezkedik el a táblán, van-e rajta zászló, felderítette-e már a felhasználó. A legfontosabb metódusa a setIsMine, mely során, ha egy cellát aknával látunk el, akkor a cella megnöveli a szomszédjainak az adjacentMines változóját egyel, hiszen a szomszédok mostmár eggyel több aknával határosak. Egy ilyen **Cell** osztály használata nagyban megkönnyíti a játékmenet során a teljes tábla kezelését, illetve algoritmusok futtatását.

## **minesweeper.gameboard.CellTest**

A Cell osztály egyetlen egy fontos metódusának tesztelésére szolgál.

## **minesweeper.gui.GameGUI.ClickListener**

Az infopanelen szereplő gombokhoz tartozó listener. A megfelelő gombokhoz rendeli a megfelelő meghívandó metódusokat, pl: a pause gomb megállítja az időt, a restart gomb újraindítja a játékmenetet, a best times gomb megnyitja a dicsőséglistát.

## **minesweeper.gui.MenuGUI.ClickListener**

A menügombok ActionListener-je. A megfelelő gombok megnyomásuk után létrehoznak az adott nehézségi fokozatnak megfelelő táblát, melyet átad a newGameStarts függvénynek.

## **minesweeper.gui.CustomDiffGUI.ClickListener**

Az ablak 2 JButton-jének listenerje, ahol meghatározzuk, hogy mit csináljanak a gombok, ha a felhasználó rájuk kattint.

## **minesweeper.highscore.HighscoreUI.ComboBoxListener**

A combobox listenerje, ami az aktuálisan kijelölt elem szerint meghatározza, milyen nehézségi fokozat rekordjait kell mutatni.

## **minesweeper.gui.WinScreen.confirmButtonListener**

A megerősítő gomb listenerje.

## **minesweeper.gui.CustomDiffGUI**

Az osztály annak az ablak megjelenítésére szolgál, amivel akkor szembesül a felhasználó, ha egyéni paraméterekkel rendelkező táblán szeretne játszani. A felhasználó megadhatja a JSpinner segítségével az oszlop, sor, illetve aknaszámot is. Az aknák százalékos aránya minimum 1% maximum 96% lehet. Ennek tekintetében minden tábla érvényesnek tekinthető, melyen legalább 1 akna van, vagy akár a nagy aknaszám miatt szinte megnyerhetetlen tábla is. Az ablakból elindítható a megadott tábla, vagy akár vissza is léphetünk belőle a főmenübe.

## **minesweeper.gui.GameGUI**

Az osztály a tényleges játékmenet közbeni ablak megjelenítésére, felhasználói inputok feldolgozására szolgál. Az ablakban a felhasználó szembesül a táblát reprezentáló JButton-ök kétdimenziós táblázatával, melyeken keresztül interakcióba léphet a játékos a táblával. A táblán kívül a felhasználó lát egy információs panelt is, ahol különböző játékmeneten felüli funkciókat ér el. Meg lehet állítani a játék közbeni időszámlálót, visszalépni a menübe, vagy újratekinteni az aktuális játékmenetet, illetve a játék végeztével a dicsőséglista is megtekinthető.

## **minesweeper.gui.GameUITest**

A GameGui osztály tesztelésére szolgál.

## **minesweeper.highscore.Highscore**

Az osztály egy nyert játékot, vagyis egy rekordot jelképez. Ezeket a rekordokat (legjobb eredményekhez tartozó), objektumokat, sorosítjuk, vagyis konvertáljuk át bináris formába, melyeket később visszaolvasunk. Implementálja a Comparable interfészt is, hogy össze tudjunk hasonlítani két rekordot, idő szerint, hogy sorbarendezzük őket. Ezekkel a tervezői döntésekkel kényelmessé válik a különböző rekordok vizsgálata, szervezése.

## **minesweeper.highscore.HighscoreData**

Olyan adatmodellt jelképez, mely tartalmaz egy rekordokból álló listát. Hat oszloppal rendelkezik, hiszen az összes rekordról szeretnénk tudni, hogy milyen paraméterekkel ment végbe a játék. Különböző adatmodellek létrehozásával, és a paraméterekre vonatkozó feltételekkel vagy kikötésekkel szortírozni tudjuk a rekordokat nehézségi fokozat szerint. A JTable és JScrollPane használata során nagyon kényelmesessé válik a különböző nehézségi fokozatok rekordjainak listázása, hiszen minden nehézséghez külön adatmodell tartozik, melyben vannak a nehézséghez tartozó rekordok. Ezt az adatmodellt az összes adat és az egyéni rekordok kezelésére szolgál, hiszen egy pályához tartalmazó minden paramétert tartalmaz.

## **minesweeper.highscore.HighscoreDataTest**

A HighScoreData osztály tesztelésére szolgáló osztály.

## **minesweeper.highscore.HighscoreUI**

A dicsőséglista megjelenítésére, azon adatainak menedzselésére szolgál. Minden indításkor visszaolvassa fájlból az eddigi rekordokat, szortírozza azokat különböző adatmodellekbe, majd a felhasználó kérésére esetlegesen megjeleníti azokat. Kényelmessé teszik a különböző nehézségekkel és táblákkal való munkát az, hogy a nehézségeknek külön adatmodelljük van és rendszerezve vannak idő szerint is. Bezáráskor, vagy a visszalépéskor természetesen elmentjük az adatokat, hogy azok később is visszaolvashatók legyenek. Az osztály csak akkor látható, vagy jeleníthető meg, a tényleges játékmenet már végetért.

## **minesweeper.gui.MenuGUI**

A főmenü megjelenítésre szolgáló osztály. Itt választhatja ki a nehézséget a felhasználó, és a program indításakor ezzel az ablakkal szembesül először. Az előre megadott beállítások kiválasztása után elindul a tényleges játékmenet, azonban ha egyéni beállításokat szeretne a felhasználó egy másik ablakkal szembesül. Ez a main osztály, itt található a program belépési program is.

## **minesweeper.gui.MenuGUITest**

A MenuGUI osztály tesztelésére szolgál.

## **minesweeper.gui.Move**

Egy egészen keresztüli user inputut tud lekezelni, majd megjátszani a lépést a tényleges táblán, és meghívni a játék UI frissítési metódusát. Egy lépés során leellenőrzi, hogy vége van-e már a játéknak, illetve meg van-e állítva a játékmenet, mert ilyenkor nem reagálunk a kiadott lépésre.

## **minesweeper.highscore.SetDifficultyData**

Az adatmodell egy előre megadott nehézségi fokozat rekordjainak kezelésére szolgál. A HighScoreData osztállyal ellentétben, ez csak 3 oszloppal rendelkezik, hiszen egy előre megadott fokozatnál a pálya paraméterei adottak, ezért azok megjelenítése teljesen irreleváns, mert minden rekordnál ugyan azok lennének. Természetesen ez preferencia, hiszen ha a HighScoreData osztály használnánk a fentebb említett fokozatokra, akkor minden pályaparamétert látnánk, pontosabban minden sorban ugyan azt. A modellnek köszönhetően könnyen megjeleníthető a táblázat, továbbá kezelhető az esemény, amikor a felhasználó különböző nehézségeket választ a HighScoreUI JComboBox-ában. Ilyen típusú adatmodellekbe rendezi a HighScoreUI a rekordokat, melyeket az összes adatból szortíroz ki.

## **minesweeper.gui.TimeCounter**

Az osztály lehetővé teszi az idő múlásának nyilvántartását, továbbá a megállítást, ha a felhasználó éppen azt szeretné. A tényleges játékmenet közben használjuk, ahol fontos az idő megjelenítése is, ezért használja az osztály a **GameGUI** attribútumot is.

## **minesweeper.gui.WinScreen**

Az osztály azt a képernyőt kezeli, mellyel a felhasználó csak akkor szembesül, ha helyesen felderíti az összes aknamentes mezőt a pályán, ezáltal megnyerve a játékot. Az ablakban megtalálható a kijátszott pálya paraméterei (oszlopok,sorok,aknák száma), továbbá a pálya teljesítéséhez szükséges idő. A játékos begépelheti a nevét az idő mellé, majd az eredménye elmentődik a dicsőséglistába a HighScoreUI osztály segítségével