**Prog3 HF - Minesweeper**

**Megoldási terv**

**A megoldásra 5 napot szánok:**

**1.** Tábla létrehozása, értékek generálása, kirajzolása

**2.** Interakciókezelés játék közben

**3.** Menürendszer, navigáció, pályaszerkesztő

**4.** Eredménytábla és kezelése, adat IO

**5.** Dokumentáció

**n: a tábla mérete (n\*n)**

**k: az elhelyezendő aknák száma**

A tábla eltárolása:

**Field in Minefield**

2D int(n\*n) tömb,

Ahol az egyes elemek -1 és 8 közötti értéket vehetnek fel.

-1: Akna

0: üres mező, inicializálási érték

1-8: a szomszédos mezőkön található aknák száma

**Field in DisplayField**

A megjelenítéshez egy másik, azonos paraméterekkel rendelkező tömb tartozik. Az itt előforduló elemek:

10: fedett mező, kezdéskor az összes mező ilyen

-1-8: balklikk esetén a táblát tároló tömbből átmásolásra kerül az érték.

9: a játékos által jobbklikkel megjelölt mező

**Tervezett osztályok:**

**Minefield**

Felelősségek: n,k paraméterekkel létrehozza az adatszerkezetet ami az aknamezőt tárolja, születésekor elhelyezi az aknákat, és kiszámolja a többi mező értékét.

Int: n,k

2D Int array: field

Minefield(n,k)

placeMines()

updateBoardValues()

getFieldValue(int x, int y)

**Display extends JPanel**

Int: n,k

2D Int array: displayField

2D JButton array: board

Display(n,k)

Minefield minefield

mark(int x, int y)

uncover(int x, int y)

**GameFrame extends JFrame**

**startScreen extends Jpanel**

**Konkrét játékmenet indulása**

A játékmenet indulásakor a méretet az n(int) változó beállításával szabjuk meg. A program létrehozza a field és displayField int(n\*n) tömböket. A displayField mező minden eleme 10-es értékkel inicializálódik. A placeMines()

**Unittesztek:**

1 GameInstance - markCell

2 GameInstance - uncover

3 GameInstance - getMinesMarked

4 Minefield - placeMines

5 Minefield – updateBoardValues

6 Minefield - checkIndex

7 GameInstance – toggleMarker

8 GameInstance - createBoard

9 ScoreData - sortScores

10