Budapesti Műszaki és Gazdaságtudományi Egyetem
Méréstechnika és Információs Rendszerek Tanszék

Multiplayer Aknakereső

Házi feladat
2015/16. II. félév

Hódos Gergő, Korpa Antal, Tolnai Dániel Zoltán

IV. évf, MSc villamosmérnök szakos hallgatók
Beágyazott rendszerek szoftvertechnológiája

Konzulens:

Györke Péter
(Méréstechnika és Információs Rendszerek Tanszék)

1. Tartalomjegyzék

2. A játék menete

Az általunk választott házi feladat a jól ismert Aknakereső multiplayer megvalósítása Java alapokon. Célunk a hagyományos játékot modern, interaktív környezetbe ültetni.

2.1 Az eredeti játék rövid leírása

A játék egy négyszögekkel lefedett síkrészen játszódik, ahol némelyik négyszög aknát rejt. A játékos sorra nyitja fel a négyszögeket, miközben célja, hogy elkerülje az aknákat. Ebben segítségére van, hogy a már felnyitott mezők megmutatják, hogy hány darab velük szomszédos mezőn van akna.

Amennyiben a játékos felnyit egy aknát tartalmazó mezőt, a játék véget ér. Amikor az összes, aknát nem tartalmazó mező felfedésre kerül, a játékos győzött. Két különböző sikeres játszma közül a rövidebb idő alatt teljesített a jobb.

2.2 Multiplayer változat

A játék logikáján nem változtatunk, a cél ugyanúgy az aknák elkerülése és a többi mező felnyitása.

A játékot két különböző módban lehet játszani. Az egyik során a játékosok szimultán játszanak saját tábla-példányon, míg a másik során körökre osztva próbálnak megoldani egyetlen pályát. Mindkét játékmód két játékost feltételez.

Mindkét játékmód során végig tudják a játékosok, hogy még hány megjelöletlen akna van, illetve hogy mennyi idő telt el a játszma kezdete óta.

2.2.1 Szimultán játékmód

Ez a mód nagyban hasonlít az eredeti játékhoz: A két játékos ugyan azt a pályát kapja és egymás tábláit nem látják a játék során. Mindketten az eredeti szabályok szerint játszanak, egészen addig, amíg mind a ketten végeznek a játékkal. A program mind a két játékosnak pontszámot számít az alábbiak szerint:

$$pontszám = \frac{helyesen\ megjelölt\ aknák\ száma - helytelenül\ megjelölt\ aknák\ száma}{játékidő}$$

Az a játékos nyer, akinek a pontszáma magasabb. A játék eltárolja az időket, így egyéni csúcs felállítására is lehetőség van.

2.2.2 Körökre osztott játékmód

A két játékos egy közös pályán játszik, körökre osztva. Egy játékos körönként egy mezőt nyithat fel, a játékosok körönként váltják egymást. A játékos a saját körében akárhány aknát megjelölhet.

Ez a játékmód lehetővé teszi, hogy a játékosok akár egymás ellen, akár egymást segítve játszanak. Amennyiben az összes, aknát nem tartalmazó mező felfedésre kerül, mindketten győztek. Amennyiben valamelyik játékos aknát tartalmazó mezőt nyit fel, ő veszített, a másik játékos pedig győzött.

3. Felhasználói felület

A játékprogramban két különböző ablak van. Az egyik az indításkor is előtűnő menü, a másik pedig maga a játéktér, ahol a küzdelem zajlik.

3.1 Menü

Itt választhatja ki a játékos, hogy a fentebb ismertetett játékmódok közül melyiket szeretné játszani. Kiválaszthatja a tábla méretét előre meghatározott gyorsgombokkal, illetve egyéni pályaméret is beállítható. Meghatározható az aknák mennyisége is, akár automatikusan, akár explicit egy számot megadva. Ezeken felül a játékos beállíthatja a játék során használt becenevét.

A menü két lehetőséget kínál a játék indítására egy host és egy join gomb formájában. Amelyik játékos a host gombot nyomja, meghatározza a játék paramétereit az előző bekezdésben foglaltak szerint. Ennek a játékosnak az IP címét kell megadni a másik játékosnak, amikor a join gombra nyom.

Amikor az első játékos megnyomja a host gombot, a játék kiírja, hogy várakozik a másik játékosra. Amint a másik játékos kapcsolódok a join gombbal, a játék elkezdődik és mindkét játékos a játéktéren találja magát.

Ezeken felül a menüben látható a legjobb idők táblázata is, valamint a gomb, amivel ez a táblázat törölhető.

3.2 Játéktér

Ezen a képernyőn zajlik a játék, így a látható tér nagy részét az aknamező teszi ki. Emellett az aknamező fölött van egy szalag, amin a játékhoz köthető információk jelennek meg, valamint itt található egy kilépés gomb is.

A megjelenítendő információk az alábbiak: játék kezdete óta eltelt idő, még megjelöletlen aknák száma, illetve a játékosok becenevei. A körökre osztott mód során az aktuális körön lévő játékos neve kijelölésre kerül, míg a szimultán játék során végig a saját becenév a kijelölt.

4. Feladatok felosztása

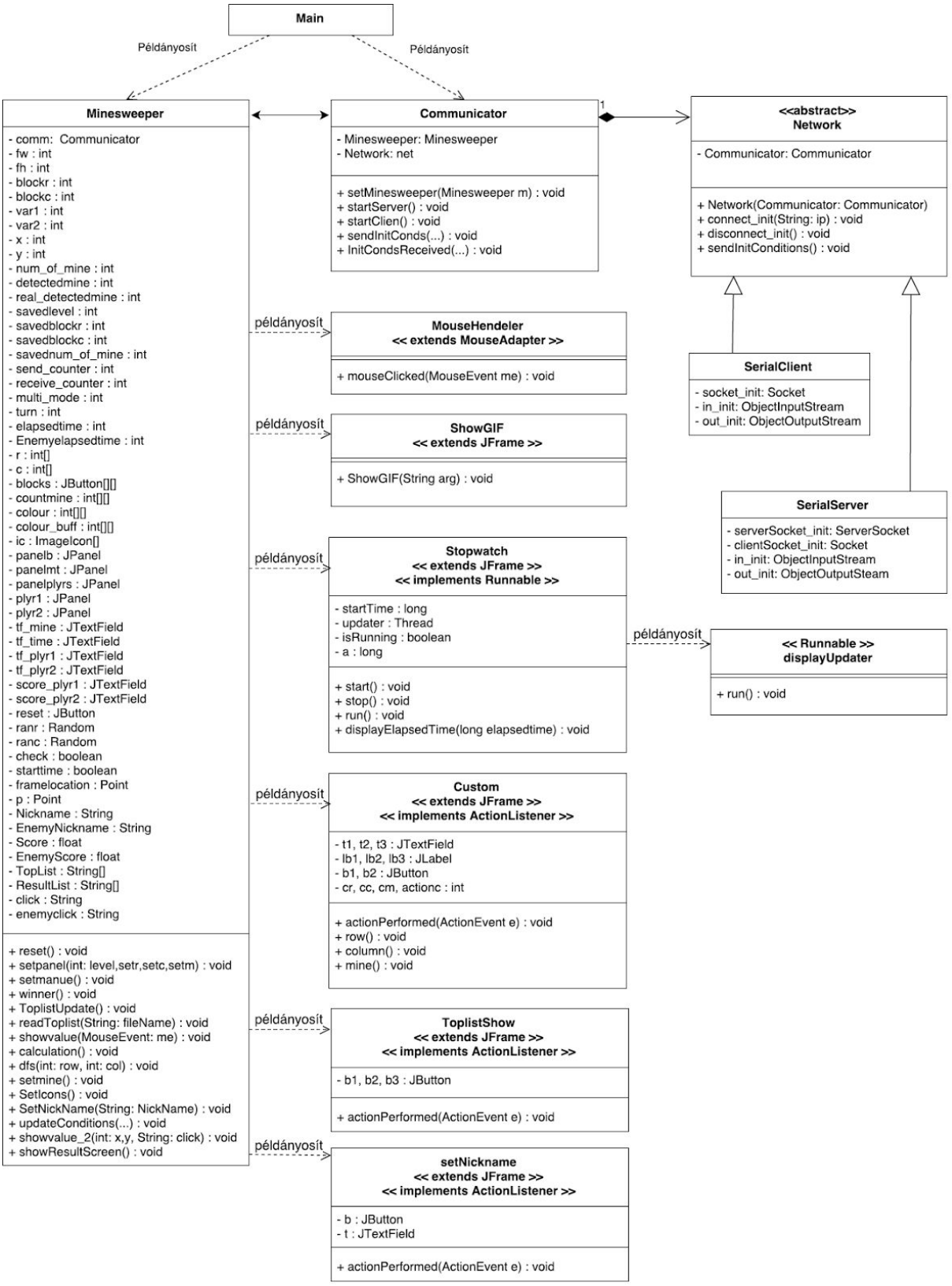
A program elkészítését három különböző részre osztottuk, így a csapat mindhárom tagjának egy-egy terület jutott:

1. Hálózati kommunikáció – Hódos Gergő
2. Játéklogika – Korpa Antal
3. Felhasználói felület – Tolnai Dániel Zoltán

5. Rendszer tervezés

5.1 Alapkoncepció

A tervezés során az alapvető elgondolásunk az volt, hogy először készítünk egy singleplayer változatot, majd ezt egészítjük ki a multiplayer változathoz szükséges egyéb funkciókkal és a hálózati kommunikáció lehetőségével. A singleplayer változatot megvalósító osztály a *Minesweeper* osztály, mely az 5.2-es alpontban található statikus osztály diagramon látható. A hálózati kommunikációhoz szükséges osztályok a *Communicator, Network, SerialClient* és *SerialServer* osztályok.



5.2 Statikus osztály diagram

5.3 Fontosabb objektumok leírása

5.3.1 Main

Ez az osztály tartalmazza a *main()* függvényt és példányosítja a *Minesweeper* és *Communicator* osztályokat.

5.3.2 Minesweeper

Ez az osztály valósítja meg a játék logikát és a megjelenítéssel kapcsolatos funkciókat, a *Communicator* osztály példányán keresztül pedig képes kommunikálni egy másik *Minesweeper* osztály példánnyal. A következőkben a *Minesweeper* osztály tagfüggvényeinek rövid leírása olvasható:

- ➔ **reset() : void**

  A játék alaphelyzetbe állításáért felelős függvény.

- ➔ **setpanel() : void**

  A játéktér (GUI) létrehozásáért felelős függvény.

- ➔ **setmanue() : void**

  A játék menü rendszerének kialakításáért felelős függvény.

- ➔ **winner() : void**

  A győztes állapot megállapításáért és az ekkor szükséges tevékenységek elvégzéséért felelős függvény

- ➔ **ToplistUpdate() : void**

  Szükség esetén a legjobb eredmények listájának frissítésért felelős függvény.

- ➔ **readToplist(String *fileName*) : void**

  A legjobb eredmények listájának beolvasásáért felelős függvény.

- ➔ **showvalue(MouseEvent: *me*) : void**

  Az aknamezőn egy kattintás eredményeként bekövetkező változásokat végrehajtó függvény.

➔ **calculation() : void**

Az aknát nem tartalmazó mezőkbe írandó szám értékek (szomszédos aknák száma) meghatározásáért felelős függvény.

➔ **dfs(int *row*, int *col*) : void**

Az kattintott mezőt körülvető aknát nem tartalmazó mezők automatizált felnyitását elvégző rekurzív függvény.

➔ **setmine() : void**

Meghatározott számú akna aknamezőn történő véletlenszerű elhelyezését végrehajtó függvény.

➔ **SetIcons() : void**

A játéktéren megjelenítendő ikonok betöltését szolgáló függvény.

➔ **SetNickName(String *NickName*) : void**

A játékos becenevét beállító függvény.

➔ **setvalue_2(int *x*, int *y*, String *click*) : void**

Hasonló funkcionalitással rendelkezik, mint a *setvalue()* függvény, a különbség csupán az argumentum listában van.

➔ **showResultScreen() : void**

Multiplayer játék esetén, szimultán játékmódban (multi_mode = 2) a játék végén az eredménytáblát megjelenítő függvény.

➔ **updateConditions(String *nickname_rec*, int *level_rec*, int *setr_rec*, int *setc_rec*, int *setm_rec*, int *x_rec*, int *y_rec*, String *click_rec*, int[][][] *countmine_rec*, int[][][] *colour_rec*, float *Score_rec*, int *elapsedtime_rec*) : void**

A másik játék példánytól érkező üzenetet fogadó és feldolgozó függvény.

5.3.3 Communicator

➔ **setMinesweeper(Minesweeper m) : void**

Ezen a függvényen keresztül kapja meg a *Communicator* osztály a *Minesweeper* osztály példányát.

➔ **startServer() : void**

Ez a függvény egy szerver socket elindítását kezdeményezi.

➔ **startClient() : void**

Ez a függvény egy kliens socket elindítását kezdeményezi.

➔ **sendInitConds(String *nickname*, int *level*, int *setr*, int *setc*, int *setm*, int *x*, int *y*, String *click*, int[][] *countmine*, int[][] *colour*, float *Score*, int *elapsedtime*) : void**

Üzenet küldést lehetővé tevő függvény.

➔ **InitCondsReceived(String *nickname*, int *level*, int *setr*, int *setc*, int *setm*, int *x*, int *y*, String *click*, int[][] *countmine*, int[][] *colour*, float *Score*, int *elapsedtime*) : void**

Üzenet fogadást lehetővé tevő függvény.