Vági Levente **220 - 1 - C**

Java programozás – 2. ZH - 2018.

FIGYELEM! A zh feladatsor letöltésével és beadásával kapcsolatos FONTOS információk olvashatók!

- A zh feladatsor, valamint a segédlet letölthető zip fáljban a \\zh.itk.ppke.hu\feladatok hálózati helyről.
- A feladatsor megoldására csak a fenti fájl használható, minden más eszköz nem megengedett segédeszköznek számít, amely használata esetén a TVSz szerint járunk el.
- A feladatsort 13.00-ig be kell adni, az Eclipse IDE-ből exportálva (File | Export) a \\zh.itk.ppke.hu\megoldasok\<hálózati_azonosító> helyre. (Beadni csak a projektet és a forrásfájlokat kell!)
- A feladatsor megoldását egyetlen projektben kérjük elkészíteni. A projekt neve a hálózati azonosító, a csomagnév pedig rendre *vasut* és *aknakereso* legyen.
- A fejlesztés során a helyi lemezeken dolgozz (ne a hálózaton), valamint készíts biztonsági mentéseket a munkádról.
- FIGYELMESEN OLVASD EL a feladatsort.
- A zh sikeres teljesítéséhez 50% megszerzése szükséges.
- A zip fájl jelszavát a gyakorlatvezető fogja megadni!

1. Feladat – Vasút (50%)

Ebben a feladatban vasúti pálya felújítás miatti vágányzár áthaladást kell szimulálni. A vonatok 'A' jelzésű állomásról szeretnének 'B' jelzésű állomásra eljutni, ott parkolnak némi időt majd visszamennek az 'A' jelzésű állomásra.

Az 'A' állomáson kezdetben 30 vonat parkol, míg a 'B'-n 10 vonat fér el egyszerre. 'A' és 'B' jelzésű állomások között egy vágány van. A szimulációban legyen egy 'A' és egy 'B' vasútállomás (*RailwayStation*), egy áthaladás vezérlő (*Controlling*), aki figyeli, mikor szabad a használható vágány a két állomás között. A vonatokat szálak fogják reprezentálni (*Train*). A szimuláció a következő.

1.) A vonatoknak 'A' állomás tároló vágányáról el kell jutnia 'B' állomás tároló vágányára. Ehhez szükséges, hogy szabad legyen a pálya (a *Controlling* osztály figyeli), illetve legyen 'B' állomáson szabad tároló vágány (ezt a *RailwayStation* kezeli).
2.) Ha szabad a tároló vágány 'B' állomáson és a köztük levő vágány is, akkor 0.3 másodperc alatt áthalad a vonat, ha nem, akkor várakozik.
3.) Miután megérkezett egy vonat a 'B' állomásra, ott parkol 3 másodpercet és megpróbál visszajutni 'A' állomásra. Itt azonban elég azt figyelni, hogy szabad-e a vágány a két állomás között (*Controlling* osztály). Ha szabad, akkor megint 0.3 másodperc alatt áthalad, majd kilép a vonat a szimulációból, ha nem, akkor várakozik.

A szimuláció a következő információkat ossza meg a felhasználóval, a foglaltság a 'B' állomásra, a várakozás az 'A' állomásra értendő.

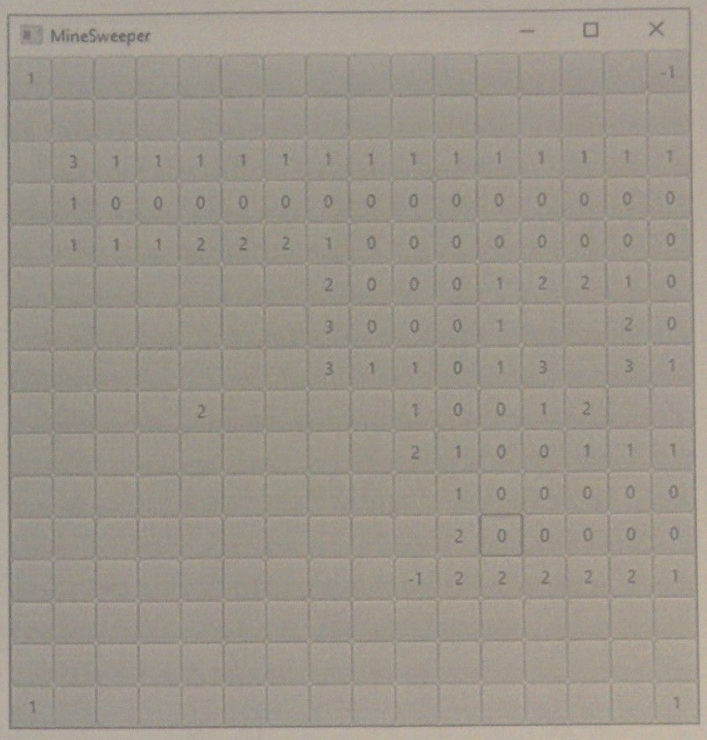
08:30:11 3 tároló vágány foglalt 20 vonat várakozik
08:30:14 4 tároló vágány foglalt 19 vonat várakozik
08:30:14 3 tároló vágány foglalt 19 vonat várakozik
08:30:15 4 tároló vágány foglalt 18 vonat várakozik
stb.

Apró hint: A tárolóvágányokat ('A' jelű állomáson 30, 'B' jelűn 10, nem kell külön osztályként létrehozni, elég a *RailwayStation*-be egy-egy int-el (vagy számos boolean-al) reprezentálni).

Fontos! A megoldás során ne használj busy waitinget, használd helyette a Java platform/library-k lehetőségeit, amiket gyakorlatokon néztünk! Figyelj arra is, hogy a programod szálbiztos legyen! Ügyelj arra, hogy ne alakulhasson ki deadlock, illetve livelock vagy starving, valamint - a lehetőségekhez mérten - valódi párhuzamosság legyen benne! A feladat megoldása közben használd az újonnan megtanult szálkezeléssel kapcsolatos irányelveket!

2. Feladat - Aknakereső (50%)

A feladat a klasszikus aknakereső játék két fős változata, szerver-kliens kapcsolattal.



Szerver

Egy dedikált szerver szükséges két kliens kapcsolódásának elfogadásához. A szerver generálja a játékteret, ami 16×16 mezőt tartalmaz (256 mező). A generált játékteret a kapcsolódó kliensek megkapják. Amikor két kliens csatlakozott, elindulhat a játék. Amennyiben valami kapcsolati hiba lép fel a játék közben, a játék álljon le hibakezeléssel.

Kliensek

A klienseknek a szervertől kapott játékteret GUI segítségével kell megjeleníteniük. Kezdetben minden mező üres, hogy a felhasználók ne lássák merre lehetnek aknák. A játék elindulását követően a szerverhez el kell jusson, hogy a felhasználó melyik mezőre kattintott.

A játék menete

Tetszőleges módon valamelyik játékos kezdi az aknák keresését. A klasszikus játékkal ellentétben nem kell jobb kattintással megjelölni az aknákat, hanem egyszerűen arra a mezőre is kattintani kell, ahol akna van. Ha a játékos aknát talál, ő kereshet tovább, az adott mezőn pedig -1 jelenjen meg (ahogy a képen is látszik). Ha a játékos nem talál aknát, akkor a kattintott mezőben megjelenik, hogy a szomszédságában mennyi akna van, és a másik játékos kapja meg a lehetőséget. (Ahogyan a klasszikus aknakeresőben is történik.) Amennyiben nincs akna az adott mező szomszédságában, jelenjen meg a kattintott mezőn a 0, és szintén a másik játékos következik. (Ez különbözik a klasszikus aknakeresőtől, ahol ilyenkor láthatóvá válik a teljes szabad terület - itt ez plusz pontért megoldható). A szerver számolja, melyik játékos hány aknát talált, az összes akna megtalálásával eredményt hirdet, és felkínálja új játék lehetőségét, vagy lezárja a kapcsolatokat.

Extra pont

Ha a kattintott mező szomszédságában nulla akna van, akkor ki kell bontani a környező mezőket (ez automatikusan történjen, nem a felhasználó által), addig, ameddig már aknák is találhatóak. Pontosan ahogyan az eredeti aknakereső játék is működik.