Beadott megoldások

Tárgy, csoport:	Osztott rendszerek (BSc)
Oktató:	Kitlei Robert Laszlo, Koszegi Judit, Mészáros Mónika
Feladat:	Osztott ZH 2017.05.22.
Határidő:	2017-05-22, 20:00:00

A beadott megoldásokhoz a beadás után még hozzászólások írhatóak.

A feladat szövege

Beadás

A BE-AD-ban.

1. Csomagold össze **zip**-be az összes könyvtáradat/fájlodat. A zip neve bármi lehet.
2. Görgess ennek az oldalnak az aljára.
3. Válaszd ki a fájlodat.
4. Hívd oda magadhoz valamelyik gyakorlatvezetőt, hogy beírja a jelszót.

Feltételek

- A ZH megoldása során **tilos** más ember (kivéve a felügyelő gyakorlatvezetők) segítségét bármilyen módon kérni, együttműködni, vagy segítséget adni.
- Használható eszközök:
    - Java API dokumentáció (https://bead.inf.elte.hu/files/java/api/)
    - Tömörített fájl (http://exam.inf.elte.hu/osztott20170522.zip), amely tartalmazza
        - a gyakorlatvezetők anyagait
        - az alábbi alapfeladathoz tartozó tesztelőt
        - a beadandó kiírását (ebben megtalálható a hsqldb jar fájlja is)

Az elkészítendő megoldásról

- A kettesért az alapfeladatot kell megoldani, minden további jegyért egy választható feladatot.
- A választható feladat megoldásához az alapfeladatot kell bővíteni. Ehhez másold át az alapfeladat megoldását a megfelelő csomagba (pl. a `zh20170522.alapfeladat` csomagból a `zh20170522.feladatA` csomagba), és alakítsd át/bővítsd a megoldást a leírás szerint.

Tesztelés

Az alapfeladathoz elérhető egy egyszerű tesztfájl, bemeneti fájlok és minta kimenet (http://exam.inf.elte.hu/osztott20170522.zip), ezen belül a `test.zip` -be csomagolva. A `Test.java` osztály tartalmaz főprogramot, ezt kell futtatni (parancssori paramétereket nem igényel). A futás végén a tesztelő összehasonlítja a standard outputon megjelenő szöveget a mintakimenettel, és jelzi, ha különbözőek.

Akkor fog jól működni a tesztelő, ha a csomag - és osztálynevek a feladatleírásnak megfelelőek.

Elégséges: Socketes szerver és kliens

Készíts egy alkalmazást, amellyel "Csapd le csacsi!" kártyajátékot lehet játszani. A játékot mindig 4-en játsszák, 1 pakli 32 lapos magyar kártyával. Az osztó (itt: szerver) mindenkinek 8 kártyát ad a játék kezdetén. A játékosok igyekeznek egy szín mind a 8 lapját begyűjteni. A kezdő játékost az osztó jelöli ki, aki egy tetszés szerinti lapot átad a következő játékosnak. Ő ezt átveszi, majd ő is átad egyet a lapjai közül a következőnek, stb. Így megy ez mindaddig, amíg valakinek a kezében összegyűlik egy szín valamennyi lapja. Ekkor ő lesz a nyertes, és ezt a tényt a "Csapd le csacsi" felkiáltással jelzi (itt: egy speciális üzenettel a szervernek). Az alapfeladatban eddig kell megvalósítani a játékot, az "A" feladatban kiegészítjük a játék legvégével is: a "Csapd le csacsi" felkiáltás elhangzása után a többiek is igyekeznek az asztalra dobni a lapjaikat (itt: egy speciális üzenetet küldeni a szervernek), aki ezt utolsóként teszi, ő a csacsi.

Megjegyzés: habár itt mindig fixen 4 játékossal foglalkozunk, igyekezzünk úgy megírni a kódot, hogy később ez rugalmasan kezelhető legyen, pl. 8 játékos 2 pakli kártyával.

KartyaSzerver

Készítsd el a `zh20170522.alapfeladat.KartyaSzerver` osztályt, mely tartalmazzon főprogramot. A szerver tetszőleges portot használhat.

- A magyar kártya lapjait a színük rövidítését (`P, Z, M, T`), illetve a számértéküket (`7-14`) tartalmazó Stringekkel jelöljük. A kártyapakli keverés előtti kezdeti állapota legyen a következő:

```
final String[] magyarKartya =
    {"P_7", "P_8", "P_9", "P_10", "P_11", "P_12", "P_13", "P_14",
     "Z_7", "Z_8", "Z_9", "Z_10", "Z_11", "Z_12", "Z_13", "Z_14",
     "M_7", "M_8", "M_9", "M_10", "M_11", "M_12", "M_13", "M_14",
     "T_7", "T_8", "T_9", "T_10", "T_11", "T_12", "T_13", "T_14"};
```

- Induláskor a kártyapakliban lévő kártyákat tároljuk le egy listában, és a következőképpen keverjük meg véletlenszerűen a kártyákat (ahol a `pakli` a kártyákat tartalmazó lista):

  ```
  java.util.Collections.shuffle(pakli, new Random(12345));
  ```

  A random seed fix kezdőértéke miatt minden egyes futásnál ugyanúgy fognak megkeveredni a lapok, ami most a tesztelhetőség miatt fontos.

- Ezután fogadja `Socket` en keresztül fogadja a `4` játékos kliensét. A kliensekkel szöveges üzeneteken keresztül kommunikál. Mindegyiktől kérje el a nevét, és tárolja el.
- Ossza szét a pakli tartalmát a kártyajátékoknál szokásos módon: mindenkinek sorban 1-1 lapot adjon, amíg el nem fogy a pakli. (Tehát az 1. játékos kap 1 lapot, majd a 2. játékos is 1-et, stb.)
- Az elsőnek csatlakozott játékosnak küldje el a `"START"` üzenetet.
- Ezután a szerver közvetítő szerepet tölt be: az egyik klienstől érkezett kártyát fogadja és továbbítja a körben következőnek.
- A játék végét jelenti, ha valaki a `"CSACSI"` üzenetet küldi a szervenek (a kártya helyett). Ekkor a szerver a győztesen kívül még mindenkinek sorban elküldi a `"VEGE"` üzenetet, majd leáll.
- Amikor a szerver üzenetet kap bármelyik klienstől, írja ki a képernyőre `"<nev>: <uzenet>"` formában.

KartyaJatekos

Készítsd el a `zh20170522.alapfeladat.KartyaJatekos` osztályt, mely tartalmazzon főprogramot.

- Két parancssori argumentumot kap: az első a neve, a második pedig egy fájlnév lesz, ez a fájl tartalmazza az

egyes körökben kiválasztásra kerülő kártyák sorszámát (a sorszámozást `1` -el kezdjük, és feltehetjük hogy minden sorban egy `1-9` közötti egész szám szerepel, hiszen lesz olyan pillanat, amikor 9 lap van egy játékos kezében).

- A kliens kapcsolódjon a szerverhez, és küldje el a nevét. Ezután fogadja a 8 kártyát, amit a szerver küld, és tárolja le ugyanabban a sorrendben, mint ahogyan a szerver küldte.
- A játék során ezután 3-féle üzenet érkezhet:
    - Ha a `"START"` üzenet érkezett, a játék elején vagyunk és a kliensünk a kezdőjátékos. Olvassa ki a megadott fájl első sorában lévő sorszámot, küldje el a megfelelő sorszámú kártyát a szervernek, és vegye ki a saját kártyái közül.
    - Ha a " `VEGE` " üzenet érkezett, vége van a játéknak, valaki más nyert. A kliens álljon le.
    - Ha a kapott üzenet nem a `"START"` és nem `"VEGE"` , akkor az üzenetben egy kártya érkezett: tárolja el a kapott kártyát a saját kártyái közé (a lista végére), majd ellenőrizze le, hogy van-e nála 8 egyforma színű kártya (ha éppen 9 kártya van nála, akkor 1 különbözhet). Ha megvan a 8 egyforma színű kártya, küldje a szervernek a `"CSACSI"` üzenetet, majd álljon le. Ha még nincs meg a 8 egyforma kártya, olvassa ki a küldendő kártya sorszámát a megadott fájl következő sorából, küldje el a kártyát szervernek, és vegye ki a saját kártyái közül.

A: Többszálúság

A megoldás fájljait a `zh20170522.feladatA` csomagba készítsd el, miután ide átmásoltad az alapfeladat megoldását.

Implementáljuk a "Csapd le csacsi" játék befejeződésének szimulációját.

A `KartyaSzerver` módosítása: amikor a győztes játékostól fogadjuk a `"CSACSI"` üzenetet, a győztesen kívül a többi játékosnak ne egymás után sorban küldjük el a `"VEGE"` üzenetet, hanem indítsunk el 3 szálat, és párhuzamosan küljdük ki a `"VEGE"` üzeneteket. A kliensek erre a `"LECSAP"` üzenettel fognak válaszolni, amit fogadjuk a szálakban külön-külön. A válaszüzenetek beérkezési idejét tároljuk el játékosonként. Várjuk be a 3 szál befejeződését, majd nézzük meg, hogy kinek a " `LECSAP` " üzenete érkezett be legkésőbb. Ő lett a csacsi, amit hirdessünk is ki a szerver oldalon `"Csacsi: <nev>"` kiírással.

A `KartyaKliens` módosítása: a kliens 3. parancssori argumentumként kap még egy egész számot, ami az ő reakcióideje. Ha a `"VEGE"` üzenetet érkezik a játék során, várjon ennyi másodpercet, majd a szervernek válaszoljon egy `"LECSAP"` üzenettel, és csak ezután álljon le.

Megjegyzés: az alapfeladathoz mellékelt tesztelő elején az import utasításban ki lehet cserélni a csomagnevet, így ezt a megoldást is lehet tesztelni. A kimenet végén 1 plusz sornak kellene szerepelnie: `Csacsi: Jatekos1`

B: Távoli metódushívás (RMI)

A megoldás fájljait a `zh20170522.feladatB` csomagba készítsd el, miután ide átmásoltad az alapfeladat megoldását.

Alakítsuk át a klienst úgy, hogy minden körben a küldendő kártyát nem a kapott fájlból olvassa ki, hanem az aktuális kártyái alapján egy RMI szerver segít meghatározni, hogy melyik kártyát érdemes továbbadnia.

Ehhez készítsük el `StategiaDeploy` osztályt, mely tartalmazzon főprogramot. A főprogram indítson egy új registry-t a `8888` porton, és jegyezzen be egy távoli objektumot `"strategia"` néven, mely megvalósítja a `StrategiaIface` távoli interfészt. Az interfész az alábbi metódust biztosítsa:

- `int tovabbkuldendo(List<String>)` : a paraméterben kapott listában található kártyák közül meghatározza, hogy melyik színű kártyából van a legkevesebb, és az első ilyen kártya sorszámát (`1-9` közötti számot) adja vissza a visszatérési értékben (a kapott listából nem töröl elemet).

A kliensek indulásukkor kapcsolódjanak ehhez a bejegyzett távoli objektumhoz. Minden alkalommal, amikor kártyát kell kiválasztaniuk továbbküldésre, hívják meg a `tovabbkuldendo` metódust az aktuális kártyáival, és az RMI

szerver által visszaadott sorszámnak megfelelő kártyát adják tovább a következő játékosnak. A továbbküldött kártyát ugyanúgy töröljük a saját kártyák közül, mint korábban.

Megjegyzés: az alapfeladathoz mellékelt tesztelő elején az import utasításban ki lehet cserélni a csomagnevet, így ezt a megoldást is lehet tesztelni. A `Test` indítása előtt "üzemeljük be" az RMI szervert a `StrategiaDeploy` indításával. Ezután az RMI szerver által biztosított stratégia alapján ugyanannak a játékmenetnek kell lezajlania, mint az alapfeladatnál.

C: Szerializáció

A megoldás fájljait a `zh20170522.feladatC` csomagba készítsd el, miután ide átmásoltad az alapfeladat megoldását.

Készíts egy szerializálható `Kartya` típust. Egy példány külön-kölün adattagban tartalmazza a kártya színét (`char`) és számértékét (`int`).

A szöveges üzenetváltást mindenhol cseréljük ki szerializált objektumokkal történő kommunikációra: a `KartyaSzerver` a kártyaosztáskor `Kartya` objektumokat küldjön a klienseknek, majd a kártyák továbbküldése egyik klienstől a másikhoz is `Karyta` objektumokkal történjen. A többi üzenetet (`"START", "VEGE", "CSACSI"`) is szerializálva küldjük (`String` objektumként).

Megjegyzés: az alapfeladathoz mellékelt tesztelő elején az import utasításban ki lehet cserélni a csomagnevet, így ezt a megoldást is lehet tesztelni. A kimenetnek meg kell egyeznie az alapfeladat kimenetével.

D: Adatbázis-kezelés

A megoldás fájljait a `zh20170522.feladatD` csomagba készítsd el, miután ide átmásoltad az alapfeladat megoldását.

Szeretnénk egy toplistát készíteni a kártyajátékosokról. Minden névhez tároljuk, hogy eddig hányszor sikerült már nyernie a játék során. Ehhez a játék végén a szerver nyisson egy adatbáziskapcsolatot, és tárolja le egy alkalmas táblába az újabb nyertest. Ha az adott névvel még nem volt bejegyzés, tárolja le a játékos első győzelmét, ha pedig már korábban szerepelt a név az adatbázisban, növelje 1-gyel a győzelmei számát.

A szerver a leállása előtt még írja ki azt is, hogy ki vezeti a toplistát (azaz mi a neve annak a játékosnak, aki az adatbázis alapján a legtöbbször nyert eddig), és hányszor nyert eddig.

Megjegyzés: az alapfeladathoz mellékelt tesztelő elején az import utasításban ki lehet cserélni a csomagnevet, így ezt a megoldást is lehet tesztelni. A kimenet végén 1 lefutás után egy plusz sornak kellene lennie:
`Toplista vezeto: Jatekos3, eddigi gyozelmei szama: 1` . Még egy lefutás után:
`Toplista vezeto: Jatekos3, eddigi gyozelmei szama: 2` , stb.