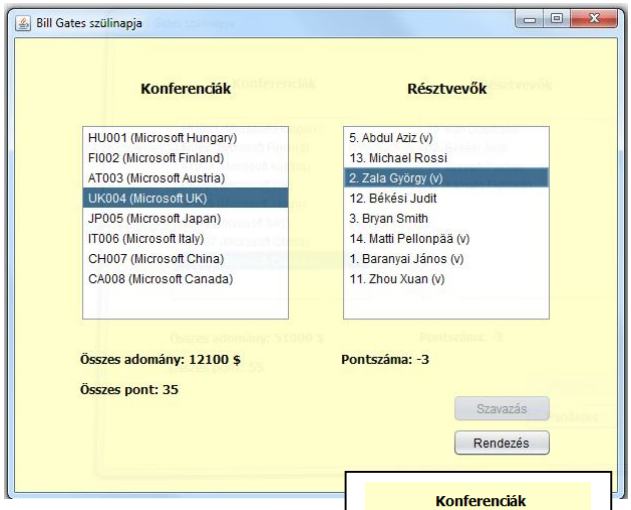
1. feladat (korábbi nappalis zh)

Ma van Bill Gates hatvanadik születésnapja. Ennek tiszteletére több helyszínen is konferenciát tartanak, melyre kétféle résztvevő regisztrálhat, a vendégek ott vannak a helyszínen, a virtuális résztvevők számítógépen kapcsolódnak az eseményhez.
Minden **résztvevő** egyértelműen megadható a *nev*ével és egy *egyedi sorszám*mal. Mindenki értékeli Bill Gates munkáját egy pontszámmal. Mindenki csak egyszer *pontoz()*hat, ekkor egy véletlen *pontszám* keletkezik (kivételesen lehet alaposztályban véletlen), mégpedig egy, az összes résztvevőre egyformán jellemző *alsó*- és *felső ponthatár* közötti érték (a felsőt is felveheti, az alsó akár negatív is lehet). (Figyeljen rá, hogy tényleg csak egyszer generálódjon ez az érték.)
A **vendég** a helyszínen vesz részt a konferencián, ez csak azért érdekes, mert ott be tud szállni Bill Gates jótékonysági tevékenységébe. (BG megígérte, hogy a konferenciákon összegyűlt adomány dupláját adja jótékonysági célokra.) Ahhoz, hogy valaki vendég lehessen, feltétlenül rendelkeznie kell valamennyi *pénz*zel. Ebből a pénzből tud adakozni, az *adomany()* pedig az adott pontszám és egy, az összes vendégre egyformán jellemző *szorzó* szorzata lesz.
Természetesen csak annyit tud adni, amennyi pénze van, illetve csak adni tud, kapni akkor sem, ha rossz véleménnyel van BG-ről.
A **virtuális** résztvevők számítógépen kísérik az eseményeket, azonban sajnos előfordulhat, hogy csak késve tudnak kapcsolódni, mert a gépük épp elkezd frissíteni. Amikor *frissit()*, akkor a metódus paraméterében lévő frissítések számától függ a *frissítési idő*, mégpedig a frissítések számának és egy, az összes virtuális felhasználóra érvényes *átlag idő* szorzata. Csak akkor tud pontozni, ha a frissítési idő nem halad meg egy, az összes virtuális részvevőre jellemző *időkorlát*ot.
Egy **konferenciá**t a *kód*jával és *helyszín*ével lehet megadni. A *regisztráció()* során egy, a metódus paraméterében lévő résztvevő kerül a *résztvevők listájá*ba – de természetesen mindenki csak egyszer vehet részt egy konferencián. A *szavazas()* során minden résztvevő pontoz. A *pontszam()* a résztvevők össz-pontszámát adja vissza, az *adomany()* pedig a vendégek adományainak összegét (ezek az értékek esetleg adattagként is számolhatók). A konferenciáknak van egy egységes *részvételi díj*a is.

Olvassa be a konferenciák és a résztvevők adatait a mellékelt adatfájlokból. Valaki akkor lehet vendég, ha egy adott határértéknél több pénze van, egyébként virtuális résztvevő lehet. (Esetünkben a konkrét határérték legyen a konferencia-díj, de általános határral oldja meg.) Ugyancsak a konferencia-díj legyen majd a vendégek szorzójának értéke is.

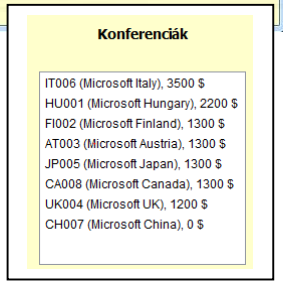
A program indulásakor azonnal jelenjen meg a konferenciák listája, és rögtön regisztráljanak is a résztvevők, azaz: valahányszor ismételje meg, hogy egy véletlenül kiválasztott konferenciára regisztrál egy véletlenül kiválasztott résztvevő (egy résztvevő több konferencián is részt vehet) .

Egy-egy elemet kiválasztva a konferenciák listájáról, a másik listafelületen lehessen látni az adott konferencián résztvevőket (a virtuális résztvevő neve mellet (v) szerepel).
Induláskor a Szavazás gomb aktív, a Rendezés inaktív, szavazás után ez megfordul.



A Szavazás gomb hatása: először minden virtuális résztvevő gépe elkezd frissíteni 0 és egy adott érték közötti véletlen frissítés-számmal. Ezután minden konferencián lezajlik a szavazás, és a konferenciák lista alatt megjelenik az összes adomány ($) és az összes pont értéke.
A résztvevők listájából kiválasztva egyetlen embert, a lista alatt az általa generált pontszám olvasható.

A Rendezés gomb hatására a konferenciák listában adományok szerint csökkenő módon jelenjenek meg a konferenciák, a név, helyszín adatok mellett az adományok mennyisége is szerepeljen.



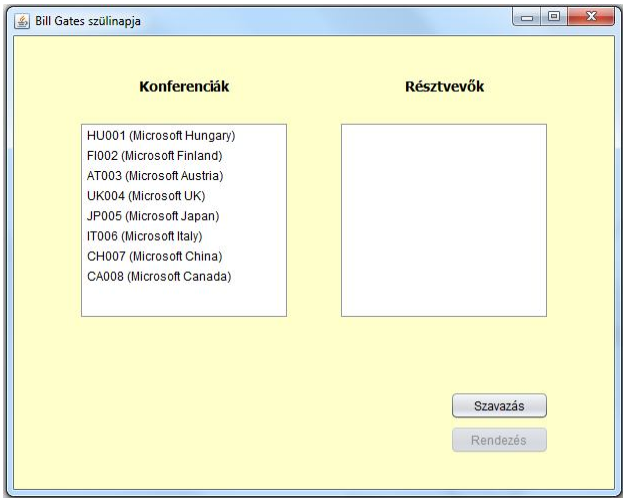
Néhány használható adat a Bill Gate-s példához

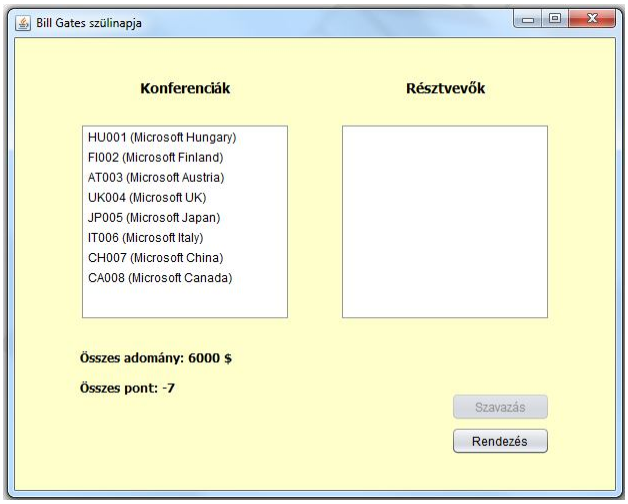
A felület belső mérete: 600*450.

Néhány javasolt érték (de el is lehet térni tőlük):

```
    private final int RESZVETELI_DIJ = 1000;
    private final int ALSO_PONTSZAM = -5;
    private final int FELSO_PONTSZAM = 5;
    private final int REGISZTRACIOK_SZAMA = 50;                    // Ennyien regisztrálnak
    private final double ATLAG_FRISSITESI_IDO = 0.5;
    private final double IDO_KORLAT = 10;
    private final double MAX_FRISSITES_SZAM = 51;                  // Legföljebb ennyi a frissítések száma
```

Néhány további képernyőkép:



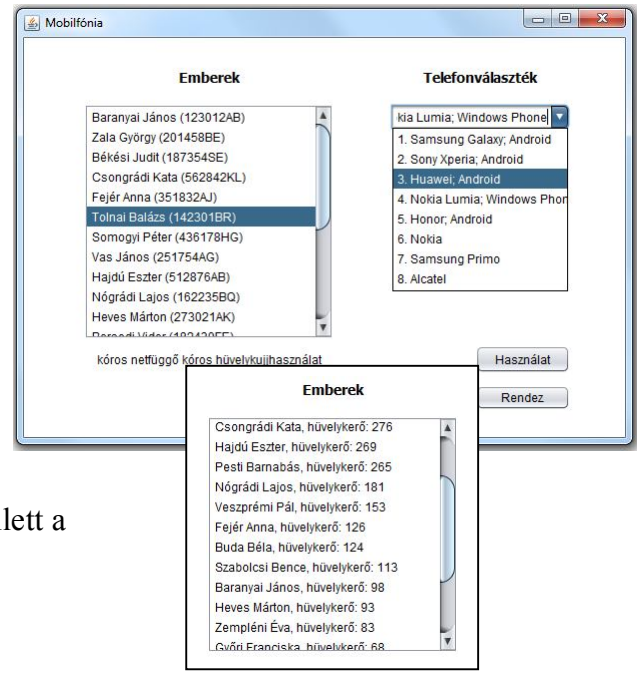


2. feladat (korábbi nappalis zh)

Október közepén volt 25 éve, hogy Magyarországon bevezették a kereskedelmi mobilszolgáltatást. (Vagyis Önök nem is éltek mobil nélküli világban.) Ennek kapcsán most azt „vizsgáljuk”, hogyan hatnak ezek a kütyük (mobil- és okos-telefon) az emberekre.
Egy **kütyü** a *típus*ával és egy *egyedi sorszám*mal azonosítható. Mindegyik tud *üzenetetküld()*eni, ekkor a kütyü által *küldött üzenetek mérete* megnövekszik a metódus paraméterében lévő karakter-számmal. Az üzenetek írása növeli az agy hüvelykujj irányítására szolgáló területét, a *hüvelykujjsejt()*ek száma (int) a küldött üzenetek méretétől függ, hasonlóan, de nem teljesen ugyanúgy számoljuk ki mobil- és okos-telefonok esetén.
Mobiltelefon esetén a hüvelykujjsejtek száma az üzenet mérete és egy, a mobiltelefonokra egyformán jellemző *billentyűerő* szorzata.
Okostelefon esetén ugyanígy számoljuk ki, csak a *billentyűerő* nagysága lesz más. Egy okostelefon a típusa mellett az *operációs rendszere nevé*nek megadásával definiálható. Az eddigieken kívül esetenként még internetezni is lehet vele. Ez azon múlik, hogy *van WiFi*, vagy nincs. A *kapcsolodik()* metódus hatására lehet kapcsolódni, ekkor a van WiFi állítás igazzá válik, a *lekapcsolodik()* metódus hatására pedig hamissá. Amikor *internetezik()*, akkor, amennyiben van WiFi, a *netezéssel töltött idő* a metódus paraméterében lévő másodpercek (int) értékével növekszik.
Egy **ember**t a *neve* és *személyigazolvány-száma* definiál. Amikor *kutyutvesz()*, akkor a kütyüinek listájához hozzáadódik a metódus paraméterében lévő kütyü. (Persze, kétszer nem veheti meg ugyanazt.) Az emberek hüvelykujjhasználatát és net-függőségét vizsgálják, ezért a *huvelykujjero()* metódus eredményeként a kütyük hüvelykujjsejt()-jeinek összegét kapjuk, a *netidő()* eredménye pedig a netezéssel töltött idők összege.
Végül a *diagnozis()* szöveges formában visszaadja a diagnózist, vagyis ha a hüvelykujj-erő nagyobb, mint egy, az összes emberre egyformán érvényes *sejthatár*, akkor „kóros hüvelykujj-használat”, ha a netezéssel töltött idő nagyobb, mint egy, szintén egyformán érvényes *függőségi határ*, akkor „kóros netfüggő”, egyébként „normális”.

A mellékelt adatfájlokból olvassa be az emberek és a kütyük adatait, a program indulásakor azonnal jelenjen meg az emberek listája, illetve egy comboBox-ban a telefonválaszték.
A comboBox-ra kattintva véletlen sokszor fusson le a következő: hozzunk létre egy ugyanolyan típusú (és esetleg operációs rendszerű) új kütyüt, amilyen a kiválasztott prototípus (ezt választjuk ki a comboBox-ból), és ezt vegye meg egy véletlenül kiválasztott ember. Természetesen többször is választhatunk telefont, és egy embernek több ugyanolyan telefonja is lehet (de nem ugyanaz).

A Használat gomb hatása: valahányszor futtassuk le a következőt: válasszunk ki egy véletlen embert. Ha van kütyüje, akkor a kütyüi közül válasszunk ki véletlenszerűen egyet. Ez a kütyü küldjön egy véletlen hosszúságú üzenetet. Ha a kütyü történetesen okostelefon, akkor valahány százalék eséllyel kapcsolódjon az internethez, egyébként kapcsolódjon le, majd véletlen hosszúságú ideig próbáljon internetezni.



A Rendez gomb hatása: hüvelykujj-erő szerint csökkenően rendezi az embereket.
Az embereket tartalmazó listában most a név mellett a hüvelykerő jelenjen meg.

Néhány használható adat a kütyüs példához

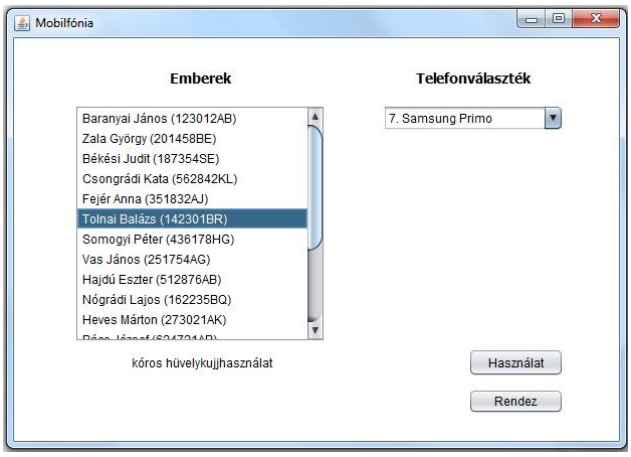
A felület belső mérete: 600*400.

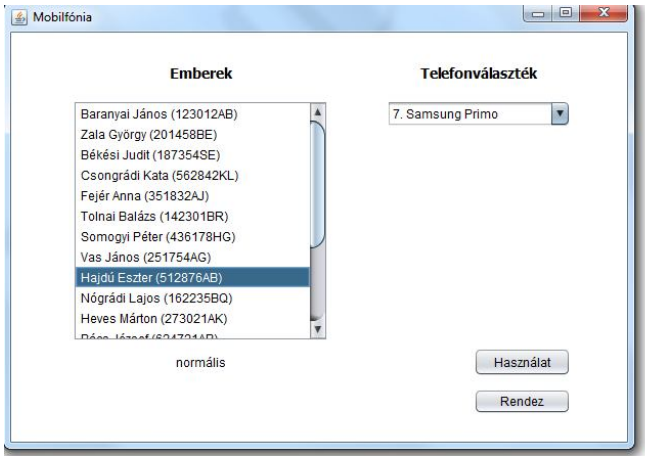
Néhány javasolt érték (de el is lehet térni tőlük):

```
private final double MAX_TELEFON_DB = 20;      // kiválasztáskor max ennyi ember vásárol
private final int HASZNALAT_SZAM =100;         // ennyiszer fut le a használat
private final int MAX_UZENETHOSSZ = 200;
private final double SZAZALEK = 0.6;// ekkora eséllyel kapcsolódik az internethez
private final double MAX_INTERNET_IDO = 200;
private final double MOBIL_BILLENTYU_ERO = 0.5;
private final double OKOS_TEL_BILLENTYU_ERO = 0.3;
private final int FUGGOSEG_HATAR = 1000;
private final int SEJT_HATAR = 200;
```

Segítség: a combobox modellje: DefaultComboBoxModel. A hozzá tartozó esemény ugyanaz, mint a gombnyomásnál.

Néhány további futási kép:





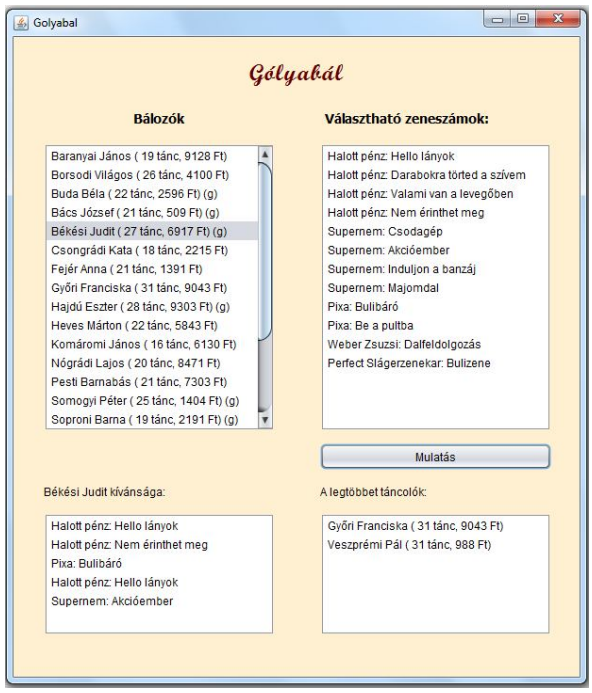
3. feladat (korábbi nappalis zh)

Elkezdték árulni a jegyeket az idei gólyabálra, igyekezzen, hogy ez a program is időben készen legyen, és szimulálja vele a gólyabált.

Minden **bálozó**nak *egyedi sorszám*a van, és természetesen *nev*e. Mindegyikük *fogyaszt()*, ekkor *költségei* a metódus paraméterében lévő értékkel növekszenek, persze, csak addig, ameddig a *zsebpénz*éből futja. Mindegyikük *táncol()* is időnként, ekkor *táncainak száma* eggyel növekszik.

A **gólyá**k természetesen kedvezményt kapnak. Bármit is fogyasztanak, kedvezményes áron kapják, mégpedig az összes gólyára egyaránt érvényes *kedvezményszázalék*kal olcsóbban. Még egy privilégiumuk van: választhatnak zeneszámot – természetesen ugyanazt többször is. Amikor egy gólya *kiválaszt()* egy zeneszámot, akkor az bekerül a *kívánt zeneszámai listájá*ba.

Egy **zeneszám** az *előadó*jával és *cím*ével adható meg.



Egy 600×650-es belső méretű felületen szimulálja a bált. (A szorgalmi induló felületet ld. a megadott segítség fájlban.)

A program indulásakor azonnal megjelenik a bálozók névsora (lehetőleg ábécé sorrendben), illetve a választható zeneszámok listája. A bálozók adatainak beolvasásakor mindenki kapjon 0 és egy egységes határ közötti véletlen zsebpénz-mennyiséget.
(Adatfájlok: *balozok.txt*, adatszerkezet:
név;évfolyam – nyilván az elsős a gólya,
ill. *zenek.txt*.)

A bálozók listájára kattintva, ha gólya a kiválasztott illető, akkor „válasszon” egy véletlen zenét az adott kínálatból (nyilván nem a másik listára való kattintással, hanem véletlen generálással, bár szorgalmiként megoldhatja, hogy kattintással választhasson, de csak a gólya). Az alsó lista-felületen az általa választott zeneszámok listája legyen látható (az eddigi összes választott), a lista fölött pedig egy felirat jelezze, hogy kinek a választását látjuk. Ha nem gólya az illető, akkor az alsó listafelület maradjon üresen, a kiírt szöveg pedig tartalmazza azt, hogy a választott ember nem gólya.

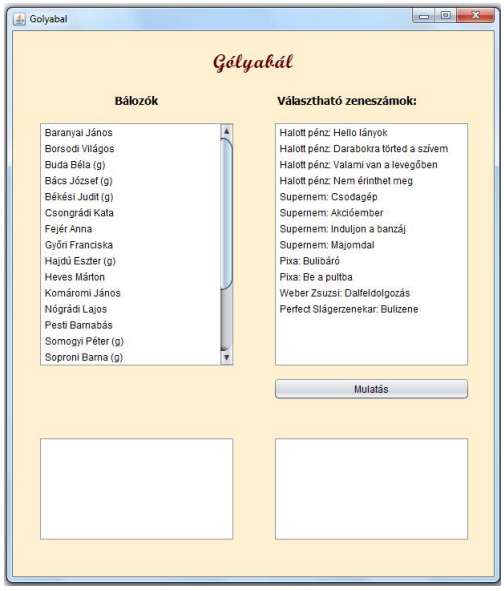
A Mulatás gombot megnyomva fusson le egy ciklus, amelyben egy véletlenül választott ember táncol (nem muszáj párban), egy másik véletlen ember pedig véletlen mennyiségű pénzt költ. A jobboldali alsó listában pedig – minden gombnyomáskor frissülve – jelenjen meg a legtöbbet táncolók listája (első alkalommal a lista feletti felirat is). Ha valaki egyszer sem táncolt, akkor csak a neve legyen olvasható a bálozók listájában (gólyák esetén mellette a g betű), ha már táncolt, akkor lássuk a táncok számát és az elköltött Ft mennyiséget is.

Néhány használható adat a gólyabálos példához

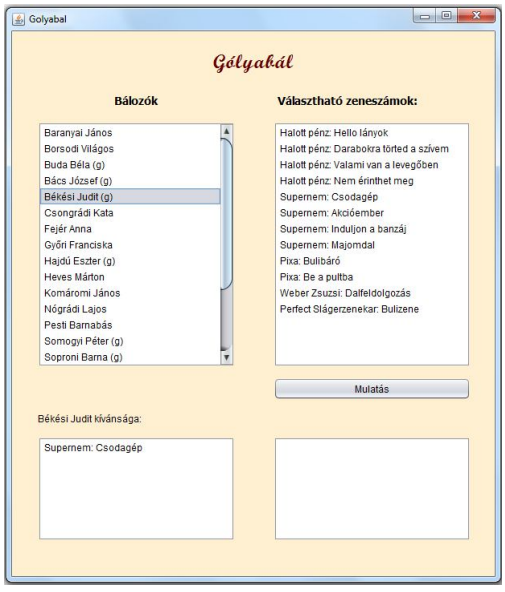
```
final int FOGYASZTAS_HATAR = 1000;
final int ZSEBPENZ_HATAR = 10000;
int MULATASI_CIKLUS_MERETE = 100;
```

Néhány további képernyőkép:

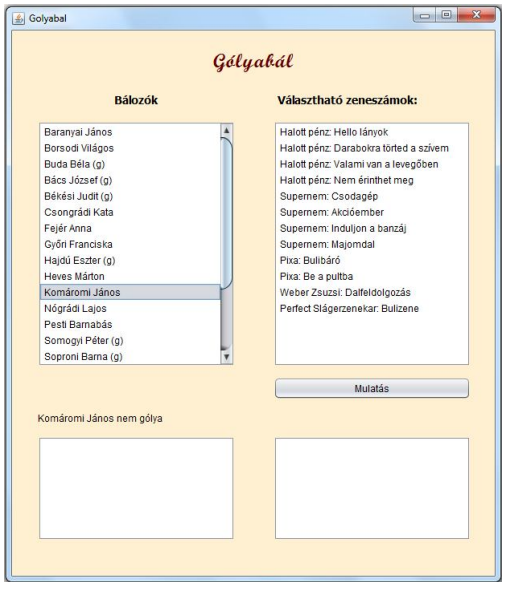
Induló állapot

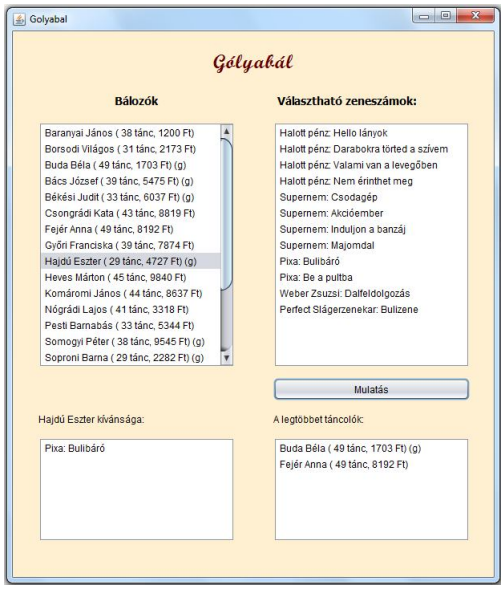


Első zeneválasztás



Nem gólyát választunkA mulatás gomb többszöri megnyomása



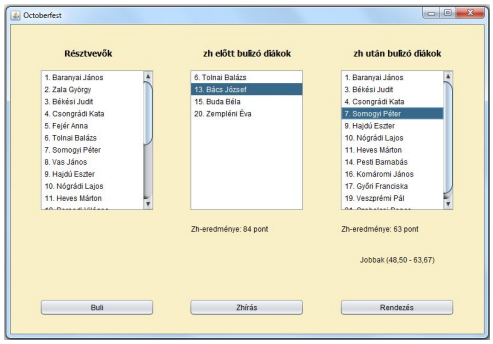


4. feladat (korábbi nappalis zh)

Ha már úgy alakult, hogy a HÖK-ös októberfeszt helyett zh-t kell írnia, írjon róla egy Java szimulációt.
Az októberfeszt lényege, hogy mindenféle személyek mindenféle söröket iszogatnak.
Egy **személy** a *nevé*vel és egy egyedi *sorszám*mal azonosítható. A sör hatására változik a *hangulat*a (egy int érték). Ha *iszik()*, akkor hangulata a metódus paraméterében lévő sör alkoholfokának és egy, az összes személyre jellemző *szorzó*nak a szorzatával növekszik, de csak egy bizonyos *határ*ig. Ha eléri ezt a határt, akkor hangulata ugyanennyivel csökken. A határ szintén azonos az összes személy esetén. A hangulatot azonban nem csak az ital befolyásolhatja, a *hangulatváltozás()* metódus hatására az illető hangulata a metódus paraméterében lévő értékkel változik. (A hangulatához hozzáadódik ez az érték.)

A mostani októberfesztre sok **diák** is elmegy. Őket a nevük és egyedi sorszámuk mellett még az *eha-kód*juk is azonosítja. Ők szegények nem csak iddogálnak, hanem még zh-t is írnak. A zh eredménye azonban valamelyest függ a hangulatától. A *zhtir()* metódus hatására keletkezik a *pontszám*a, mégpedig így: pontszáma a metódus paraméterében lévő pontérték hangulatnak megfelelő százalékkal növelt értéke. (ugyanúgy kell számolni, mint ár és áfakulcs esetén).

Az októberfeszt elengedhetetlen kelléke a **sör**. Ezt *márká*jának és *alkoholfok*ának megadásával definiálhatjuk.



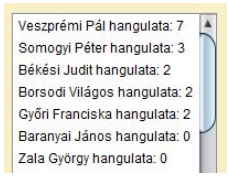
Az itt látható 800 × 550-es felület baloldali listájában az összes résztvevőt láthatjuk.
A résztvevők adatait a *diakok.txt* fájl tartalmazza, a söröket pedig a *sorok.txt*. (Adatszerkezet: név; és ha az illető diák, akkor az eha-kód is; ill. márkanév;alkoholfok)

A személyek beolvasása után a diákokról véletlenszerűen döntse el, hogy elmennek-e már zh előtt is bulizni, vagy csak utána. A diákok kb. x%-a iszik előre a medve bőrére. Az eredménynek megfelelően kerüljenek a diákok a két következő listába.

A buli gomb hatására szimuláljuk a bulit. Ez a következőt jelenti: véletlen sokszor fusson le a következő: válasszunk ki egy véletlen személyt. Ha az illető nem tartozik a zh után bulizók közé, akkor igyon egy véletlenül választott sört. A kiválasztott személy hangulata a bulitól függetlenül is változhat, vagyis bárkit is választottunk, változzon a hangulata egy véletlen értékkel. (A hangulat negatív is lehet, azaz pl. -5 és 5 között változzon.)

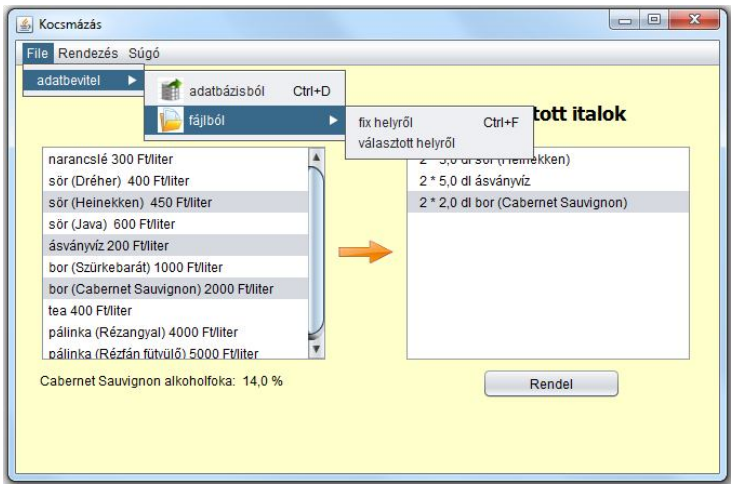
A zhírás gomb hatására az összes diák írja meg a zh-t. A paramétert véletlenül generálja. Amelyik csoport átlaga jobb, az alá írja oda, hogy jobbak, és a két átlagot is (előbb bulizók – utóbb bulizók átlaga) Ha a két átlag egyforma, akkor ne csináljon semmit. A listán kiválasztott diák zh-eredménye jelenjen meg a megfelelő lista alatt.

Rendezze a résztvevőket a hangulatuk alapján. Ekkor ilyen legyen a lista:



5. feladat

A feladatok_3.pdf 6. (kidolgozott) feladatának folytatása. Módosítsuk a múltkori projektet:

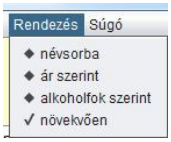


A felület legyen 50 pixellel szélesebb, mint múltkor (azaz 600*500-as), és dönthessünk, hogy honnan vesszük az adatokat.

Most engedjük meg, hogy az itallapról egyszerre egynél több italt is választhassunk (ekkor az alkoholfok kiíratását célszerű a másik listához rendelni), és a kiválasztott italok a nyíl gomb megnyomásakor kerüljenek be a választott italok listájába.

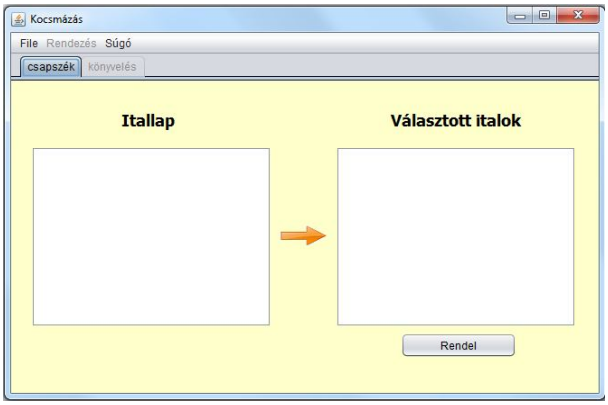
Az egyszerűség kedvéért most minden italhoz hozzárendelünk egy default rendelhető mennyiséget, a rendelések száma és ez a mennyiség jelenik meg a jobboldali listafelületen.

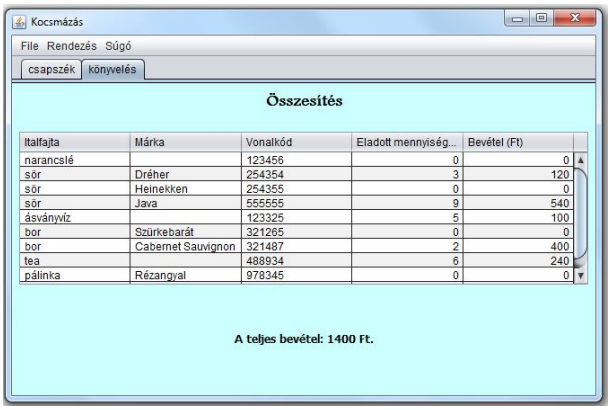
A Rendel feliratú gomb hatására realizálódik a rendelés, ekkor a választott italok eltűnnek a listafelületről, és megjelenik a fizetendő érték.



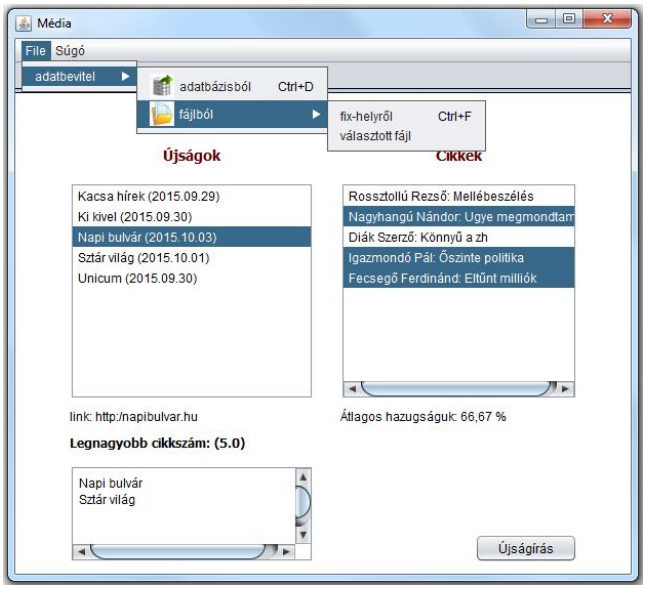
És hogy ne menjen kárba a múltkor kikínlódott rendezés, a menüpont segítségével rendezzük is az adatokat.

Ha múltkor már megoldotta, akkor most pár perces munka ez az átalakítás:





6. feladat:
Játsszunk tovább a médiabirodalommal!



Egy **újság**ot egyértelműen jellemez a *neve* és *megjelenési dátum*a (sima String, de próbálkozhat igazi dátummal is, ha kedve tartja). Az újság *cikket közöl()*, vagyis a metódus paraméterében megadott cikket hozzáadja az újságban megjelent *cikkek listájá*hoz (úgy oldja meg, hogy minden cikk csak egyszer szerepelhet egy-egy újság listájában).

Egy **cikk** egyértelműen megadható a *szerző nevé*vel, a *cikk címé*vel, a *cikk méreté*vel (karakterszám) és egy, a cikkben lévő hazugság nagyságára utaló *százalékláb* értékkel.
Az újságok azt is meg tudják „mondani”, hogy mekkora a bennük lévő cikkek átlagos hazugságszázaléka.

A médiabirodalom nyomtatott és internetes újságot is megjelentet.
A **nyomtatott újság**ot a nevén és megjelenési dátumán kívül jellemzi még az újság *példányszám*a és a *méret*e (összesen hány karaktert tud megjelentetni). Mivel itt korlátozott a méret, ezért egy cikk közlése csak akkor lehetséges, ha annak mérete még belefér az újság méretébe.

Az **internetes újság** a néven és megjelenési dátumon kívül tartalmazza még az újság *link*jét (most csak egy String).

Az *ujsagok.txt* és a *cikkek.txt* adatfájlból olvassa be az adatokat, majd az *ujsagIras()* során véletlen sokszor egy-egy véletlenül választott újságban jelentessen meg egy-egy véletlenül választott cikket. Az újságírás gomb hatására hívjuk meg ezt a metódust. A gombnyomás hatására azonnal jelenjen meg egy szövegmezőben a legtöbb cikket tartalmazó újságok névsora.

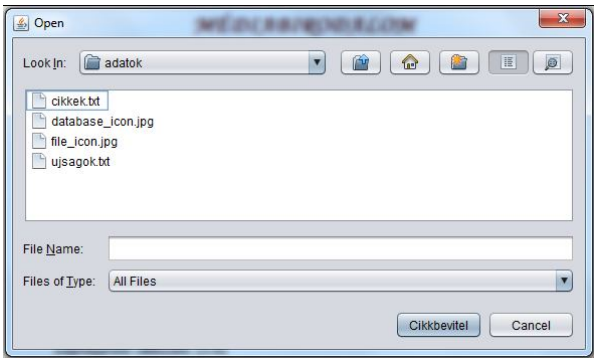
Az újságra kattintva a lista alatt lehessen látni értelemszerűen vagy a példányszámot vagy a linket, a másik listában lássuk a benne megjelent cikkeket. Egyszerre csak egy újságot lehessen kiválasztani.
A cikkek közül viszont akár többet is, a lista alatt legyen látható a kiválasztott cikkek átlaghazugság értéke.

Hogy a felhasználó tudja, hogy mikor mit is kell tennie, mindkét listához rendeljünk egy-egy ismertető tool-tip-et.
Innen folytassuk a következőkkel:

a/ Használjunk igazi dátumokat.

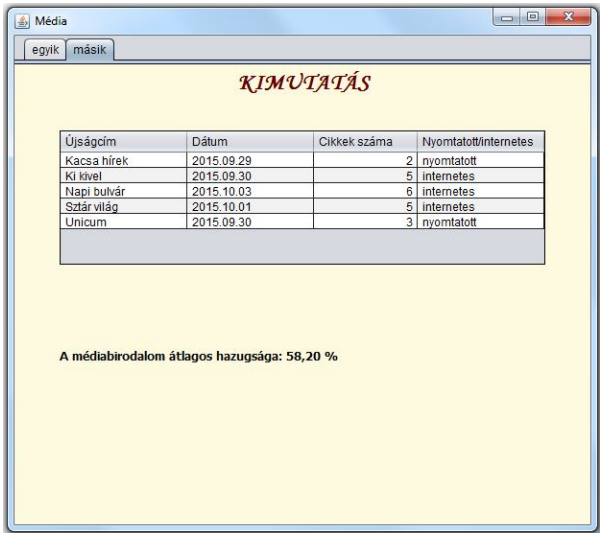
Segítség:
Az adatot `Date` típusúra kell deklarálni.
Beolvasás:
`String`-ből `Date`:
```
datum =new SimpleDateFormat(datumFormatum).parse(stringAdat);
```
ahol a `datumFormatum` a beolvasandó dátum formátuma, esetünkben "yyyy.MM.dd".
Az `Ujsag` osztályban pedig a dátum `String` alakját kell előállítani.
Ezt így lehet: `new SimpleDateFormat("yyyy.MM.dd").format(datum);`

b/ Írassuk ki az újságokat névsorba rendezve.



c/ Készítsük el az előző oldal ábráján látható menüt, és oldjuk meg, hogy mindhárom adatbeviteli mód működjön.

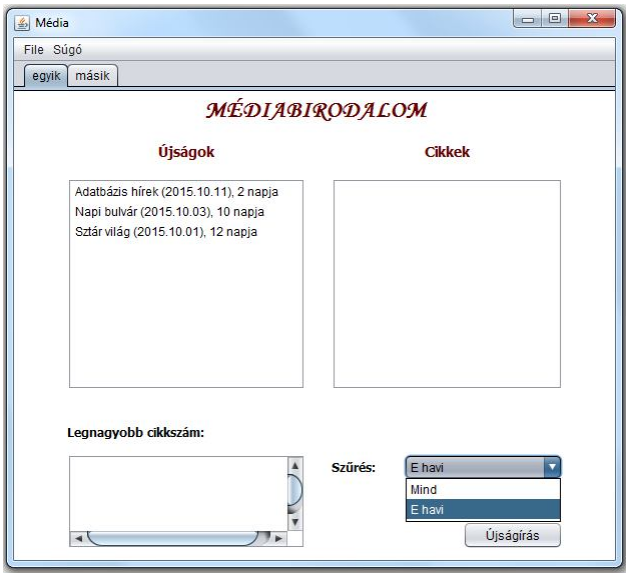
A „választott fájl” menüpont hatására nyíljon meg egymás után két fájlválasztó ablak, az egyikből válassza ki a cikkek adatait tartalmazó fájlt, a másikból az újságokét.
(A fájlválasztónak létezik „multipleSelection” módja is, úgy egyszerre lehet kiválasztani a két fájlt, de oda kellene figyelni arra is, hogy jó legyen a kiválasztott fájlok sorrendje. Ha kedve, ideje van, próbálja így is megoldani.)



A megoldás során kihasználjuk, hogy az adatbevitelt egy interfész implementálásával oldottuk meg, így most ugyanazt az interfészt implementáljuk más módon, így az `AdatBevitel` osztály példányosításán kívül gyakorlatilag semmit sem kell változtatni a már meglévő programon.

d/ Ha eddig még nem tette volna, akkor oldja meg, hogy regiszteres módon tudjuk kezelni az újságokat. Az első fülre kattintva az előző felületet lehessen látni, a másodikra kattintva ezt a táblázatot, illetve a táblázat alatt a médiabirodalom átlagos hazugság-százalékát.

Állítsa be úgy a táblázat megfelelő tulajdonságát, hogy a fejlécre kattintva az adott oszlop szerint lehessen rendezni az adatokat. (AutoCreateRowSorter)



e/ Igazi dátumok használatával az ábrán látható részfeladatok is megoldhatók, azaz:

- Az újságok neve, dátuma mellé az is kerüljön ki, hogy hány napja jelent meg.
- Rakjon bele egy szűrőt, amely alapján vagy az összes újságot látjuk, vagy csak az ebben a hónapban megjelenteket. Ez utóbbi esetben a táblázatba is csak ezek kerüljenek.
- Esetleg próbálja meg dátum szerint rendezve kiíratni az újság-lista elemeit.

Segítség: a napok számát A `Date` osztály `getTime()` metódusa alapján lehet könnyen kiszámolni. Az aktuális dátum lehet a `Date` osztály paraméter nélkül létrehozott példánya.

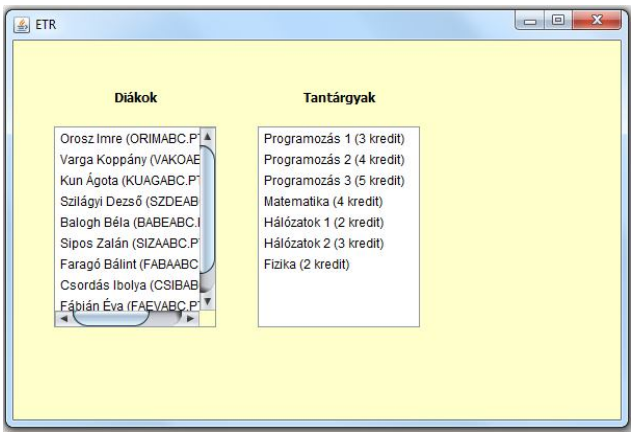
A hónapot pedig a `Calendar` osztály `get(Calendar.MONTH)` értékei alapján célszerű szűrni. De a napokat is lehet ennek az osztálynak a segítségével számolni:
(`get(Calendar.DAY_OF_YEAR)`).
Az osztály példányosítása: `Calendar.getInstance();` a dátumot pedig az adott példány `setTime()` metódusával lehet beállítani.

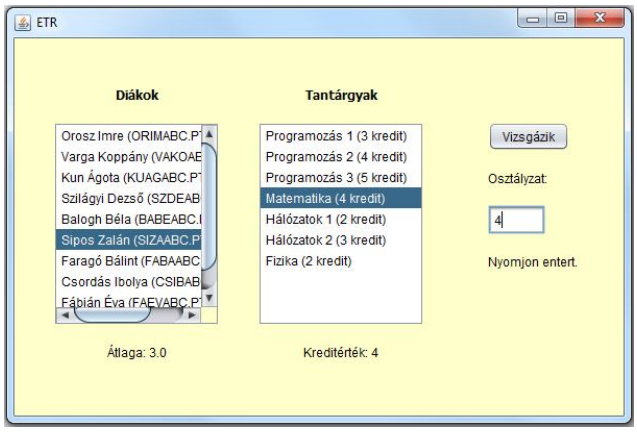
7. feladat:
Vannak diákok és költségtérítéses diákok. Mindegyikőjük esetén meg kell adnunk a nevét, EHA-kódját, születési évét. Mindegyik vizsgázik(), ekkor érvényes (azaz minimum 2-es) jegy esetén az átlaga is és a teljesített kreditek száma is változik.
Egy bizonyos átlaghatár fölött ösztöndíjként megkapják az átlag valahányszorosát. Ez a szorzó minden diák esetén azonos.
A költségtérítéses diákoknak tandíjként a teljesített kredit valahányszorosát kell fizetniük. Ez a szorzó minden költségtérítéses diák esetén azonos.

Olvassuk be az adatokat a mellékelt *diakok.txt* és a *tantargyak.txt* fájlból vagy a megfelelő adatbázisból.
Adatszerkezetek:
diakok: név;kód;születési_év;költségtérítéses-e (1 ha igen, 0 ha nem – adatbázisban true/false)
tantárgyak: név;kód;kreditszám

Jelenítsük meg a beolvasott adatokat egy 600*400-as grafikus felületen két listájában.

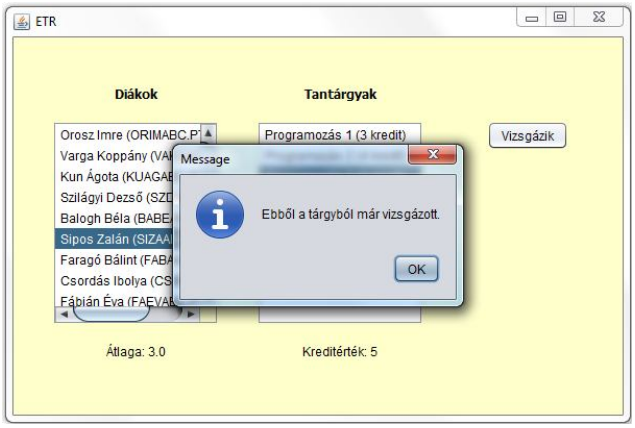
Vizsgáztassuk a diákokat, azaz a „Vizsgázik” gomb hatására jelenjen meg a korábban láthatatlan szövegmező (a feliratok is láthatatlanok), amelybe beírhatjuk a diák osztályzatát.
Természetesen ezek a komponensek csak akkor váljanak láthatóvá, ha van kiválasztott diák és kiválasztott tárgy.
Ha begépeltük a jegyet, akkor az enter hatására ismét tűnjenek el a komponensek – feltéve, hogy érvényes jegyet gépeltünk, egyébként dobjon hibaüzenetet.

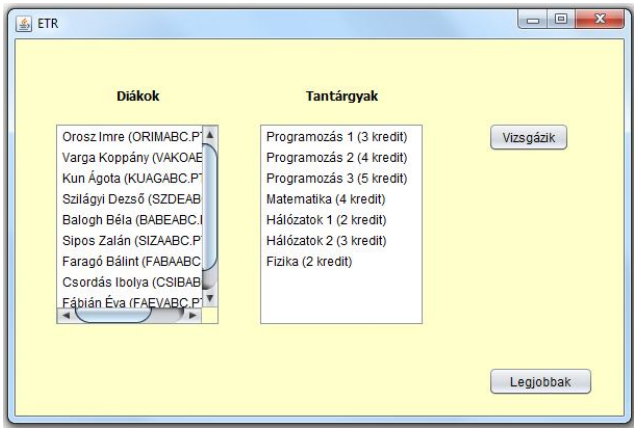




Oldja meg, hogy jelezze, ha a kiválasztott diák már érvényes vizsgajegyet szerzett a kiválasztott tárgyból.

A „Legjobbak” feliratú gombra kattintva határozza meg a legjobb átlagú diákok névsorát, és jelenítse meg a grafikus felületen. (Lehet lista, és lehet szövegmező.)





Egy másik panelre kattintva meg lehessen nézni, hogy melyik diák milyen tárgyakat teljesített.

Rendezze a diákokat ösztöndíj szerint csökkenő sorrendbe.

Még egy újabb panelen egy táblázatban jelenítse meg a diákok nevét, eha-kódját, azt, hogy költségtérítéses-e vagy sem, és az átlagát.