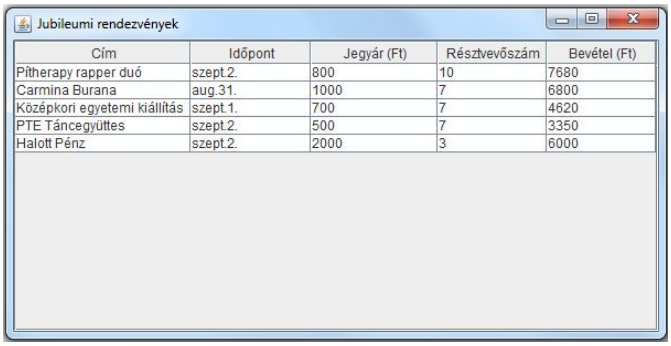
TUDNIVALÓK:
Most is és a következő gyakorlatokon is – akkor is, ha külön nem emeljük ki – az órán meg nem oldott feladatok HÁZI FELADAT-ként megoldandóak!!!
Ez fontos a tárgy sikeres teljesítéséhez!

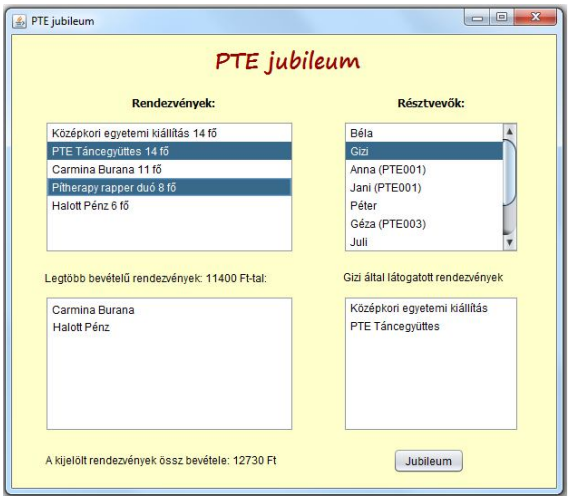
1. feladat:

a) „Fejezzük be” az előző órai feladatot, és
- írjunk egy kis teszt osztályt;
- beszéljük meg, hogyan lehet kiíratni a jubileumi rendezvényeket egy táblázatba:

b) Most alakítsuk át a múltkori megoldást grafikus felületű alkalmazássá, mégpedig úgy, hogy lehetőleg minél többet használjunk a meglévő megoldásból.



Cím	Időpont	Jegyár (Ft)	Résztvevőszám	Bevétel (Ft)
Pitherapy rapper duó	szept.2.	800	10	7680
Carmina Burana	aug.31.	1000	7	6800
Középkori egyetemi kiállítás	szept.1.	700	7	4620
PTE Táncegyüttes	szept.2.	500	7	3350
Halott Pénz	szept.2.	2000	3	6000



Az ábrán látható külsejű alkalmazás a következőket tudja:

A program indulásakor már legyenek olvashatóak a rendezvények (névsorban) és a résztvevők listájának adatai, illetve adatbeolvasáskor minden résztvevő kapjon valamennyi pénzt.

A Jubileum feliratú gomb megnyomására fusson le az a szimuláció, amely már korábban is feladat volt, vagyis az, hogy minden rendezvény esetén mindegyik résztvevő „döntse el” véletlenül, hogy részt akar-e venni a rendezvényen vagy sem.

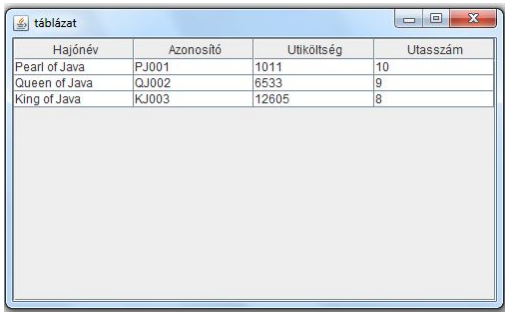
A gomb hatására jelenjen meg a legtöbb bevételt hozó rendezvények listája is, maguk a rendezvények pedig legyenek résztvevőszám szerint csökkenően kiírva (most úgy, hogy a rendezvény neve és a létszám).

A maximális bevételű rendezvények listája alatt lehessen látni a kijelölt rendezvények összbevételét.

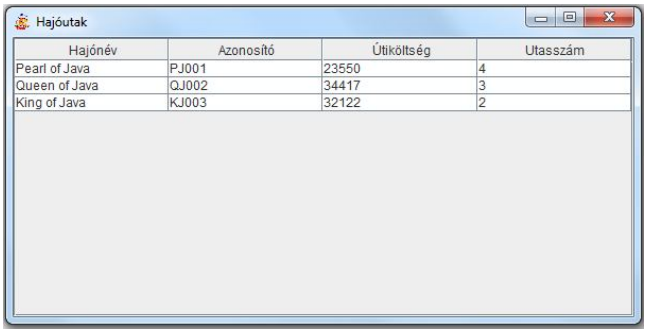
A résztvevők listája alatt pedig azt, hogy a kiválasztott résztvevő mely rendezvényeken vett részt. Mindkét felirat csak akkor jelenjen meg, ha már választottunk a megfelelő listából.

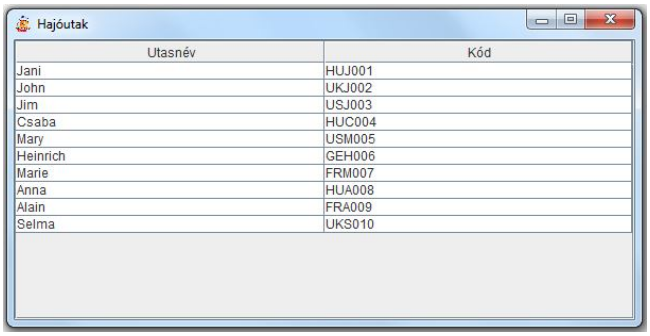
2. feladat:

Még mindig JAVA-TRAVEL. Próbáljuk meg ilyen formában is kiíratni az eredményt:



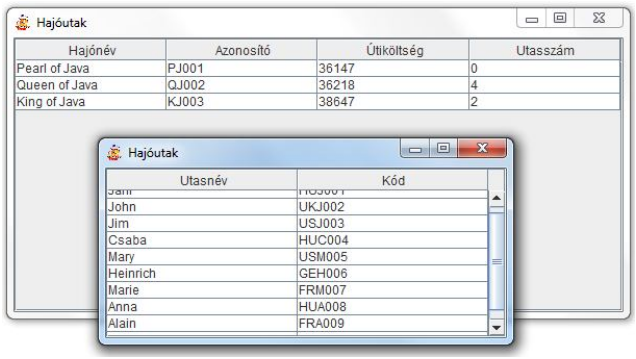
Önállóan bővítse tovább a feladatot, és most úgy oldja meg, hogy a hajók helyett az utasok adatait írja ki egy táblázatba.





Most próbálja meg úgy, hogy mindkét ablak megjelenjen

Most úgy, hogy az utaslistát tartalmazó ablak bezáró gombjára kattintva ne fejeződjön be a futás, csak tűnjön el az ablak, a hajóutak listáját tartalmazó ablakra kattintva viszont záródjon be az alkalmazás.

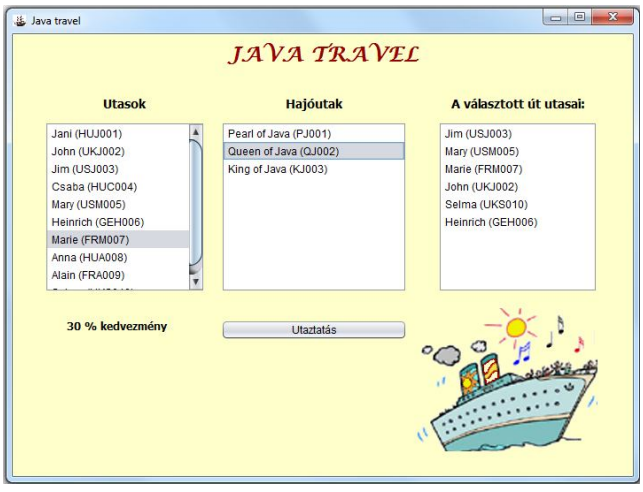


Ezeket még a konzolos alkalmazásból próbálja megoldani, mert segít a későbbiek megértésében, és az eddigiekhez képest nem nagy munka.

3. feladat

Alakítsuk át a feladatot grafikus felületűvé.

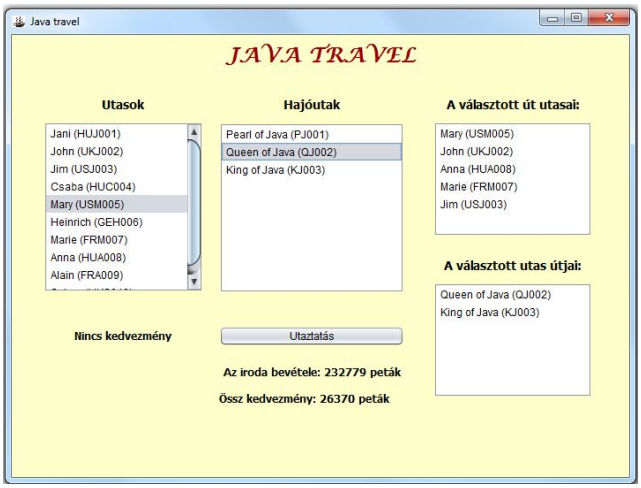
A felület 700*500-as méretű, induláskor látható a potenciális utasok és a hajóutak listája. Az utasok listájának bármelyik tagjára kattintva, a listafelület alatt olvasható, hogy az illető hány % kedvezményt kap, vagy az, hogy nincs kedvezmény.



Az Utaztatás feliratú gombra kattintva fusson le a korábbi projekt vezérlésének utaztatas() metódusa. Egy-egy hajóútra kattintva a harmadik listafelületen jelenjen meg az illető hajó utasainak listája.

A megoldás kapcsán strukturáljuk át a projektet: használjunk több csomagot, illetve vegyük külön az adatbeolvasást a vezérléstől. Az adatbevitelhez írjunk egy AdatInput nevű interfészt, és ezt implementálja majd a fájlból való olvasásra írt osztály. A fájlból való olvasást most úgy oldjuk meg, hogy az adatfájl része legyen a projektnek.
Önálló feladat: próbálja meg kódismétlés nélkül megoldani a fájlokból való olvasást.

Folytassuk a feladatot:



- Ha már költségekbe veri magát, akkor utazhasson az utas, vagyis az utazik() metódusa hatására az aktuális utat adja hozzá a hajóútjai listájához.

- A kedvezményes utas esetén azt is határozzuk meg, hogy útjai során összesen mennyi (hány peták) kedvezményt kapott a cégtől?

- Tesztelje a kedvezmény kiszámítási módjára írt kódot.

- Ha jól működik a metódus, akkor az Utaztatás feliratú gomb hatására a gomb alatt jelenjen meg a cég teljes bevétele és a kedvezményekre fordított összeg is.

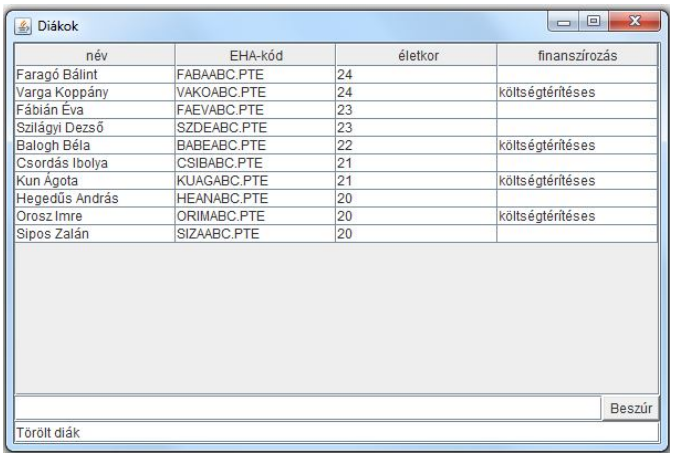
FONTOS MEGJEGYZÉS: Meglehetősen pazarló (redundáns) megoldás az, hogy minden hajóút objektum tartalmazza a hozzá tartozó utasok listáját, és minden utas objektum a hozzá tartozó utak listáját, azaz sok adatot duplán is tárolunk.
Feladat: gondolkozzon el rajta, és próbálja megoldani, hogy nem redundáns módon lehessen tárolni az adatokat. (Az adatbázisból tanultak segítenek, de elég a józan paraszti ész is ☺)

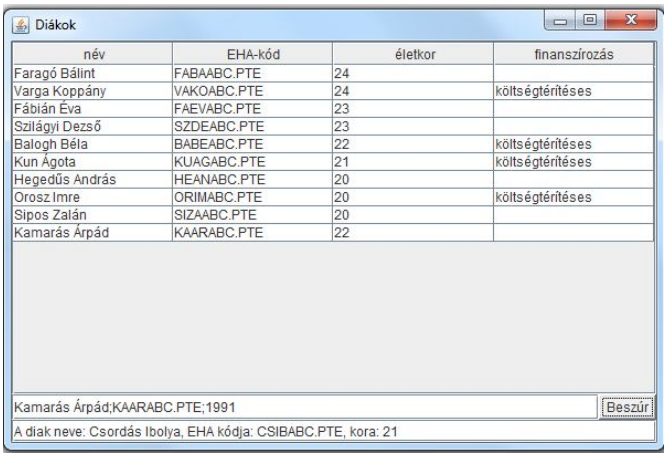
4. feladat:
Nem egyszer lehet szükség rá, hogy konzolos alkalmazás kimeneteként hozzunk létre pl. egy táblázatot, ezért van értelme az ilyen, nem „generálós” feladatnak.

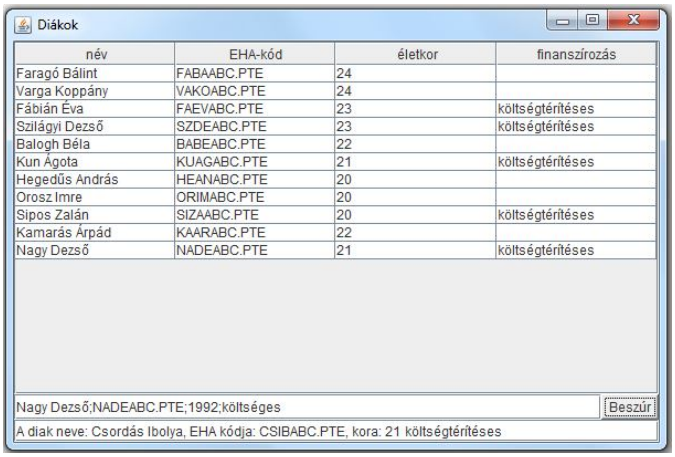
Vannak diákok és költségtérítéses diákok. Mindegyikőjük esetén meg kell adnunk a nevét, nepgun-kódját, születési évét. A diákok adatait a *diakok.txt* fájl tartalmazza. Olvassa be őket, és minden egyes diák esetén véletlenszerűen döntse el, hogy az illető költségtérítéses-e vagy sem. Vegye figyelembe, hogy a diákok kb. 40%-a lehet költségtérítéses.

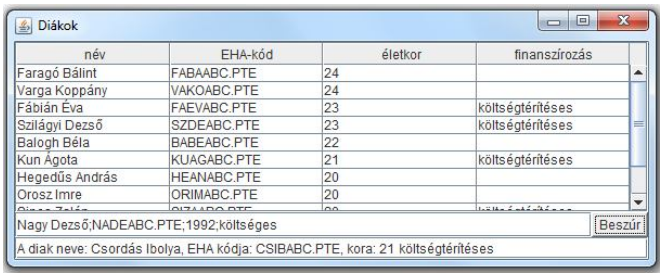
Írassa ki az adatokat egy ablakba, majd az ablak alján hozzon létre egy beviteli mezőt, mellette egy „Beszúr” feliratú gombbal, alatta pedig egy másik, nem szerkeszthető szövegmezőt.
A gombnyomás hatására kerüljön be a táblázatba a gomb előtt lévő szövegmezőbe írt adatok alapján készült sor. (Az egyszerűség kedvéért most feltételezhetjük, hogy helyesen töltik ki a sort *név;kod;szülév* vagy *név;kod;szülév;valami* formában – ha van negyedik adat is, akkor

költségtérítéses az illető. Esetleg dobhat hibaüzenetet, ha nem ilyen a beírt sor. (Hibaüzenet: `JOptionPane.showMessageDialog(this, "üzenet");`)
Ha rákattintunk a tábla valamelyik sorára, akkor a sor törlődjön ki, és a kitörölt diák adatai jelenjenek meg az alsó szövegmezőben.







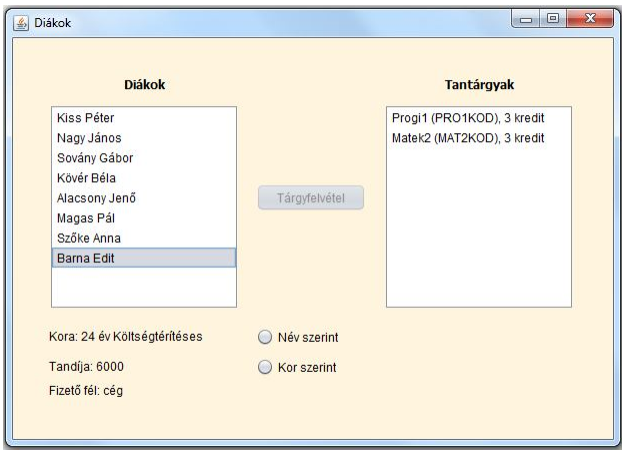


Természetesen scrollozható.

Segítség:
A komponenseket előbb rakja egy (vagy több) panelre – attól függően, hogy hányra teszi, a panel vagy FlowLayout() vagy BorderLayout() elrendezésű legyen.
A gomb figyelésére ActionListener()-t lehet rendelni, ennek ActionPerformed() metódusa kezeli az eseményt, a táblához egy MouseListener()-t lehet rendelni, itt a MouseClicked() metódust lehet használni.
Törölni, beszúrni a táblamodellből/be lehet.

5. feladat:

Oldja meg az előző feladatot teljesen grafikus alkalmazásként. Vagyis vannak diákok és költségtérítéses diákok. Mindegyikőjük esetén meg kell adnunk a nevét, kódját, születési évét. Mindegyik tárgyat tud felvenni, ekkor a felvett kreditjeinek száma növekszik.
A költségtérítéses diákoknak tandíjként a felvett kredit valahányszorosát kell fizetniük. Ez a szorzó minden költségtérítéses diák esetén azonos.



Egyelőre használjuk a már megírt beolvasást, és a program indulásakor a beolvasott diákok jelenjenek meg az ábrán látható módon egy 600*400-as belső méretű alkalmazás listafelületén.

A Tantárgyfelvétel gombot megnyomva fusson le az előző órán megírt szimuláció, majd a gomb váljon inaktívvá.

A listára kattintva alatta jelenjen meg a kiválasztott ember kora, és ha költségtérítéses, akkor ez az információ is. Ez esetben a tandíj értéke és a fizető fél is legyen olvasható. Oldja meg, hogy a listából csak egy elemet lehessen kiválasztani.

Rádiógombokkal lehessen módosítani a rendezési szempontot: vagy névsorba, vagy kor szerint, de önálló továbbfejlesztésként még kerüljön hozzá szempontként a teljesített kreditek száma is. Esetleg még azt is beállíthatja, hogy növekvő vagy csökkenő sorrendben.

Azt is oldjuk meg, hogy a program jar fájlból is futtatható legyen. (Ehhez az adatfájlokat is a projekt belsejében kell megadni.)

6. feladat (megoldását ld. külön):
Na végre, kocsmázunk. ☺

A képen látható felület belső mérete: 600x400.

Az italok ugyanúgy adhatóak meg, mint az előző feladatban, vagyis az ital definiálásakor meg kell adnunk a fajtáját (bor, tea, víz, stb), vonalkódját és literenkénti árát.

Az alkoholos italt a fentieken kívül még egy márkanév és az alkoholfok is jellemzi.

Mint látható, az itallapra az ital

fajtája és literenkénti ára kerül, alkoholos italok esetén a fajta mellett zárójelben a márkanév is olvasható. Az itallapra kattintva jelenjen meg a lista mellett egy feliraton a kiválasztott ital alkoholfoka. Alkoholmentes ital esetén ne jelenjen meg semmi. Figyeljen rá, hogy egyszerre csak egy italt tudjunk kiválasztani.

Az itallap adatait az *arlista.txt* fájl tartalmazza, ezek az adatok a program indulásakor azonnal bekerülnek a listába, de úgy, hogy hiba esetén hibaüzenetet is kapjunk. (Sőt, kapjon egyet a felhasználó, egyet pedig a fejlesztő.)
Játszadozzunk el az itallappal, és rendezgessük különféle szempontok alapján.

Szorgalmi: nézzen utána, hogyan lehetne megoldani azt, hogy az ékezetes karaktereket jó helyre rendezze. Azon is elgondolkozhat, hogy hogyan lehetne kódból felrakni a rádiógombokat.

Segítség:
1. Hibaüzenet: JOptionPane.showMessageDialog(…)
2. A rádiógombokat ne felejtse el megfelelően csoportosítani.
3. Mindkét képecske egy-egy label ikonja, nincs benne semmi különös.
(Általános segítség a feladatsor végén.)

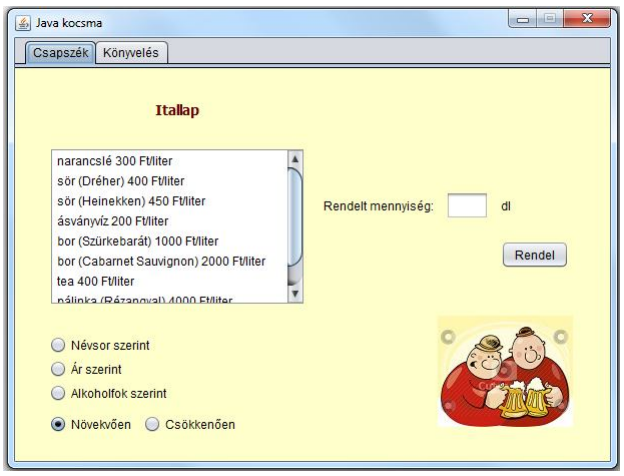
Folytassa az előző feladatot, és
a/ rendeljünk is az itallapról!

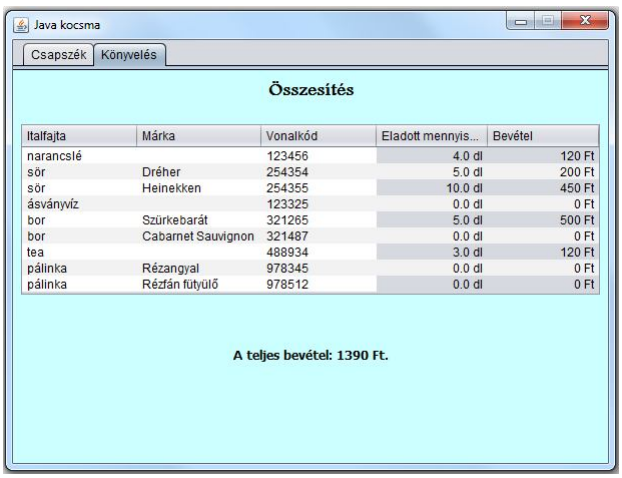
A rendelés gomb hatására írja ki a fizetendő összeget.
Egyelőre egyetlen ember rendeléséről van szó, ezért a fizetendő összeg az egyes rendelésekkor fizetendő értékek összege.

Ha nincs kiválasztva semmi, vagy hibás az adatmegadás (a negatív érték is hiba), akkor adjon hibaüzenetet.

b/ Írjon teszt fájlt a rendeléshez.

c/ Készítsen összesítést a kocsma napi forgalmáról!





Itt egy kicsit szabadjára van engedve a fantáziája:

a/ Lehet független az előzőtől, ekkor a rendelések után nem kell kiírni a fizetendő összeget.
b/ Össze lehet kapcsolni vele, ekkor kellene még egy „Fizet” gomb, ami csak annyit csinál, hogy nullázza a fizetendő értéket, azaz jöhet a következő vendég.

Ami igazán lényeges, és amit alaposan végig kell gondolni: hogyan tudja összesíteni a rendeléseket. Segítség: célszerű lenne egy Rendeles osztályt is definiálni. Azt már gondolja végig, hogy hogyan, és persze, azt is, hogy ezt hogyan lehet majd felhasználni.

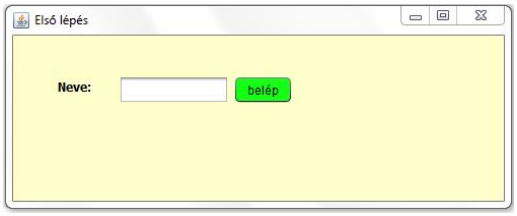
De természetesen úgy is megoldható, hogy az Ital osztályban definiál egy rendel(int mennyiseg) metódust, amely számolja, hogy összesen hány dl-t adtak el az illető italfajtából, ill. mekkora az érte járó össz-bevétel, és a felületen való italrendeléskor ezt a metódust hívja meg.

Tesztelje is ezt a metódust!

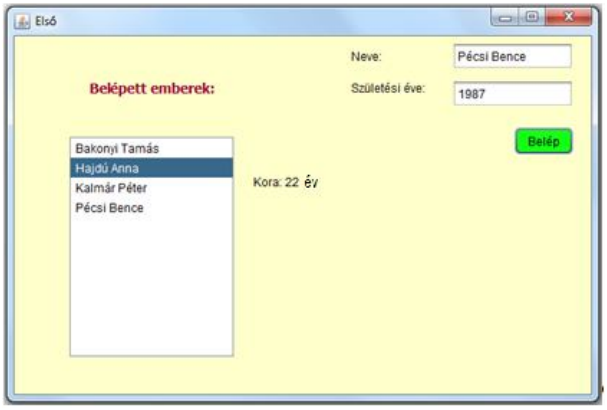
További (egyszerű) feladatok azok számára, akik nagyon kezdőknek érzik magukat:

7. feladat:
Első lépésként hozzunk létre egy „beléptető” rendszert, vagyis az 500×200-as felületen a nyomógomb megnyomásának hatására üdvözöljük a szövegmezőbe írt nevű embert. Üres mező esetén adjon hibaüzenetet. (Csak akkor, ha nem unja ☺)

8. feladat:
Játsszunk kicsit, és módosítsa az előző feladatot úgy, hogy ha az egérrel a nyomógomb fölé megyünk, akkor annak háttérszíne változzon meg zöldre, ha viszont az egér elhagyja a nyomógomb területét, akkor a színt változtassuk vissza.



9. feladat:
Az előzőek folytatásaként most egy listában jelenítsük meg a beléptetett emberek nevét.
Belépni a „Belép” feliratú gomb hatására lehet, ekkor a megadott nevű, születési évű ember bekerül a listába. Hibás születési év esetén adjon hibajelzést. (Nem szám, negatív, nagyobb, mint az aktuális dátum (évszám).)
Ha sikerült, akkor a listára kattintva, a lista mellett jelenjen meg a kiválasztott ember életkora. (Most egy fix helyre kerüljön, de azt is megoldhatja, hogy mindig a kiválasztott ember neve mellett jelenjen meg a kora.)



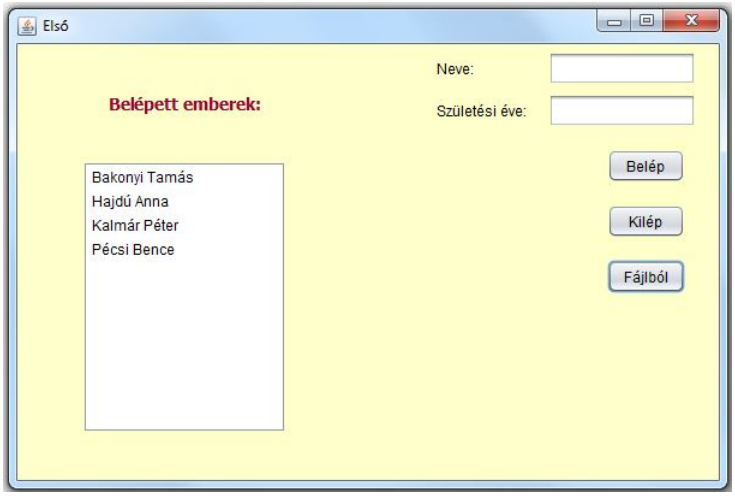
Figyeljen rá, hogy mindenki csak egyszer kerülhessen be a listába.
Sikeres adatbevitel után ürítse ki a beviteli mezőket, és fókuszáljon a névmezőre.

10. feladat

a/ A „Kilép” gombra kattintva törölje ki a listából a kijelölt embert. Ekkor a korára vonatkozó felirat is tűnjön el.

b/ Oldja meg, hogy egyszerre több embert is törölhessen a listából.

c/ A „Fájlból” gomb hatására fájlból töltse be az adatokat.
A fájl egyelőre lehet az src mappa fix helyén, de ha kedve és ideje van, akkor utánanézhet, hogy hogyan lehet interaktív módon kiválasztani a fájlt.

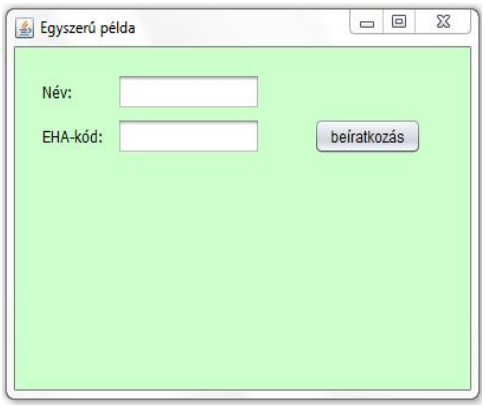


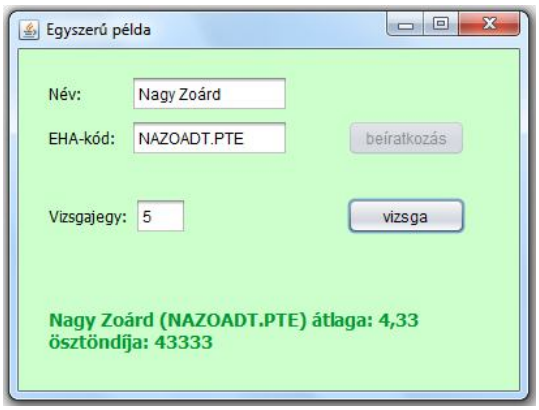
11. feladat

A program egyetlen diákot tud kezelni. Ha megadjuk a nevét, neptun-kódját, majd rákattintunk a beiratkozás gombra, akkor jelenjen meg a többi komponens is, és vizsgázhasson a diák (a beiratkozás gomb viszont váljon inaktívvá) . A legelső vizsga után jelenjenek meg az adatai is, vagyis a neve, neptun-kódja (ne zavarja, hogy a képen EHA-kód szerepel ☺), átlaga, ösztöndíja (illetve adott esetben az, hogy nem kap ösztöndíjat). Természetesen újabb vizsga esetén módosulnak az adatok.

A szükséges statikus változókat konstansként adjuk meg.

A diák akkor kap ösztöndíjat, ha átlaga egy egységes határ fölött van, ekkor az ösztöndíj az átlag valahányszorosa – természetesen a szorzó is egységes érték.





Kicsit összetettebb feladatok:

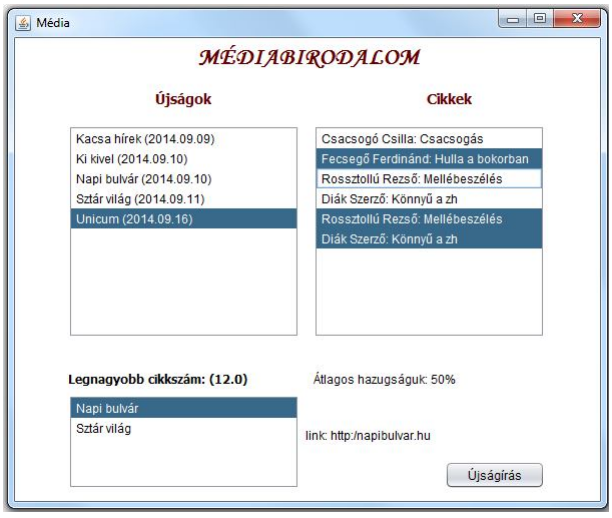
12. feladat

Ennek a korábbi feladatnak a „grafikásítása” lesz a cél.

Az osztály-leírásokat, a hivatkozott ujsagiras() metódus leírását megtalálja a korábbi feladatsorokban (gyak_2 feladatsor, 2.a, 2.b feladat).

(De van egy rövid emlékeztető a következő feladatban is.)

Az adatokat természetesen fájlból olvassuk.



a/ Írassuk ki az újságokat névsorba rendezve.

(Az ábrán látható frame mérete: 600 × 500.)

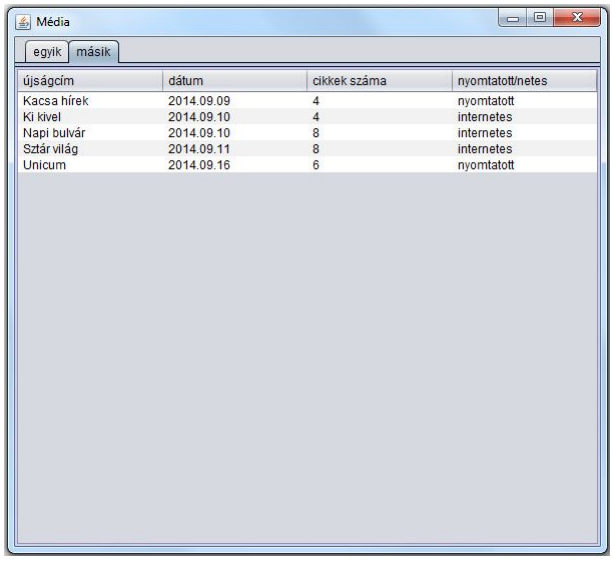
Az újságírás gomb hatására hívjuk meg a korábban megírt `ujsagiras()` metódust, amelynek hatására a beolvasott cikkek közül véletlenszerűen bekerül néhány egy-egy újságba.

Az újságra kattintva a másik oldalon 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.

Az újságokat tartalmazó lista alatt azoknak az újságoknak a nevét lehessen olvasni, amelyekben a legtöbb cikk jelent meg. Ebből a listából is csak egyetlen nevet lehessen választani, és ha ez internetes, akkor a lista mellett jelenjen meg az újság linkje, ha nyomtatott, akkor a példányszáma.

b/ Oldja meg, hogy regiszteres módon tudjuk kezelni az újságokat. Az első fülre kattintva az előző felületet lehessen látni, a másodikra kattintva a gyak_3 feladatsorban szereplő változatot, vagyis egy táblázatot.



Segítség: a tab-fülek miatt a frame-t kicsit nagyobb méretűre kell megadni, kb. 550-es magasságúra.
A táblázatot rárakhatja az előző órán vett Ablak osztály pici módosításával, de lehet úgy is, hogy a palettán található Table elemet használja – járjon utána, hogy ezt hogyan lehet, de majd a kiadott segédletben erről is lesz pár szó.

13. feladat:
Egy pszichiátriai klinikán szenvedélybetegeket vizsgálnak. Egy emberen egyszerre csak egyfajta szenvedélyt tudnak vizsgálni, ezért a kórházi felvételkor egyértelműen el kell dönteni, hogy **alkoholfüggő**, vagy **facebook-függő** egyént vesznek fel. Egyelőre csak diagnosztizálnak, vagyis a vizsgálat célja annak eldöntése, hogy az illető valóban *függő()-e*.
A vizsgálat abból áll, hogy a pácienseket engedik tetszőleges sokszor *inni()* is és *internetezni()* is (de mivel egy büfé és egyetlen számítógép van, ezért egyetlen időpillanatban csak egy ember lehet a büfében, ill egy ember a gépnél – és hogy mérhető legyen az eredmény, ezért minden más italforrást és minden saját gépet elkoboznak).

Betegfelvételkor meg kell adni a **páciens** *nevét*, *TAJ-számát*, alkoholfüggők esetén azt is meg kell adni, hogy az illető *visszaeső*-e.
A vizsgálat során bárki *iszik()* és *internetezik()*, de csak az alkoholfüggőknél figyelik, hogy mennyit isznak, és a facebook-függőknél, hogy mennyit interneteznek. Facebook-függők esetén minden egyes internetezéskor eggyel nő a *belépések száma*. Alkoholfüggők egyszerre egy pohárral isznak, és ilyenkor az *alkoholszintjük* megnövekszik a paraméterként megadott pohárban lévő alkohol mennyiségével. A facebook-függő akkor tekinthető függőnek, ha az internetes belépések száma meghalad egy, az összes facebook-függőre egyformán érvényes *határt*. Alkoholistáknál a függőség megállapításakor azt is figyelik, hogy az illető visszaeső-e. Egy visszaeső akkor tekinthető függőnek, ha alkoholszintjük egy, az összes alkoholfüggőre egyformán érvényes *határ1* érték fölött van, nem visszaeső esetén pedig akkor, ha ez az érték egy *határ2* fölött.

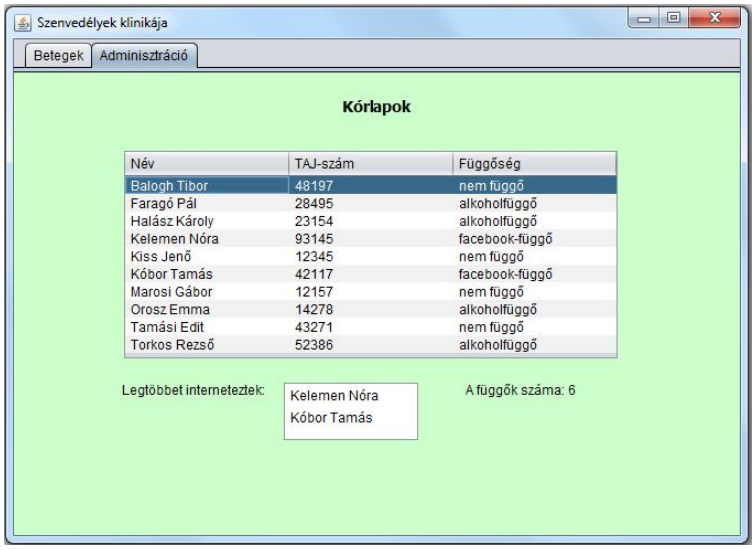
A **pohar**-at egyértelműen megadhatjuk a benne lévő ital *nevének*, *mennyiségének* és *alkoholfokának* megadásával. Ezek alapján kiszámolható a pohárban lévő *alkoholmennyiség()*. (Ezt adatfájlból illene olvasni, de ha nincs elég ideje, megadhat néhány konkrét értéket is.)

A szimulációt egy ehhez hasonló felületen végezze. (Ez most 600*400-as).

A program elindulásakor már vannak betegek, az ő adatukat egy adatfájlból olvassa be.
Az adatok soronként egy-egy pácienshez tartoznak, a páciens nevét, TAJ-számát tartalmazzák. Ha alkoholistáról van szó, akkor az is szerepel, hogy visszaeső-e vagy sem. (Mondjuk, 0 vagy 1.)

A meglévő betegek mellé újat is fel lehet venni.

A „Vizsgálat” gombra kattintva megkezdődik a *vizsgálat()*.
Ennek során véletlen sokszor fut le a következő: véletlenszerűen döntse el, hogy épp ki iszik, illetve ki internetezik a betegek közül. Aki iszik, az a rendelkezésre álló poharak közül véletlenszerűen választ egyet.
Végül összesítse az eredményeket: névsorba rendezve írassa ki az összes beteg adatát. Állapítsa meg, hogy kik interneteztek a legtöbbet, illetve, hogy összesen mennyi számít függőnek a megfigyelt betegek közül.



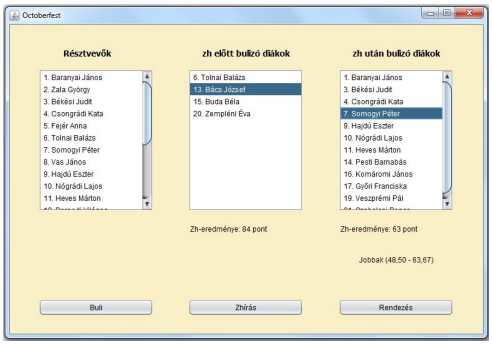
Szorgalmi: Ha látogató érkezik, akkor a rendszer meg tudja mondani, hogy a klinikán van-e a keresett páciens. Ehhez találjon ki valamilyen megfelelő felületet.

14. 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.



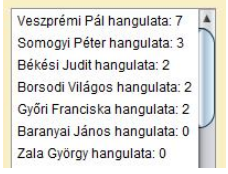
Az itt látható 800 × 550-es felület baloldali listájában az összes résztvevőt láthatjuk.
A résztvevők adatait a *diakok.txt* fájl tartalmazza, a söröket pedig a *sorok.txt*. (Adatszerkezet: név; és ha az illető diák, akkor az eha-kód is; ill. márkanév;alkoholfok)

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:



15. feladat:

Találjon ki egy feladatot, és oldja meg.
A feladatban legyen
- öröklődés (legalább 2-3 osztály)
- adattagok, metódusok
- Lista
- Lista-elemeken végzett műveletek
- véletlen szám generálás (ha frappáns feladatot talál ki, ez ki is maradhat)
- rendezés
- fájlból olvasás (esetleg írás)
- és az egészet grafikus felületen oldja meg.

Általános segítség:

1. Az újrahasznosíthatóság elve nagyon fontos az OOP szemléletben. Ezért sokkal elegánsabb, ha a panelt nem „drótozzuk be" a vezérlő osztályba, hanem külön kezeljük. Ezért hozzunk létre egy feluletek csomagot, és ebben egy JPanel form-ot. Erre húzzuk fel a megfelelő komponenseket, és értelmezzük a szükséges eseményeket.
De persze, ennek a panelnek valahogy rá kell kerülnie a frame-re. Ezért előbb fordítsuk le a projektet. Ha sikerült, akkor ugyanúgy rá lehet húzni a saját panelt a frame-re, mintha az „gyári panel" lenne. Ennek feltétele az, hogy a panelnek legyen paraméter nélküli konstruktora, vagyis úgynevezett JavaBean legyen, és persze, le is forduljon.

2. Listakezelés:
Az úgynevezett modell-view-controller (MVC) szemlélet értelmében különválasztjuk az adat-modellt, a felületet és a vezérlést. Ez a helyzet a listakezeléssel is. Az adatmodell esetünkben egy – mondjuk – italokból álló speciális lista lesz, az ú.n. `DefaultListModel`.
Deklarációja:
```
private DefaultListModel<Ital> modell = new DefaultListModel<>();
```
(vagyis csaknem ugyanolyan, mint bármelyik más listadefiníció)

Ezt a modellt hozzá kell rendelnünk a listafelülethez (`JList`). (Vigyázat, kétféle listáról van szó: a `JList` egy grafikus komponens, a swing része, a `List` pedig a util csomag listakezelésre alkalmas interfésze).
A hozzárendelést célszerű a konstruktorban megoldani, ha már létrejött a modell, egyébként pedig akkor, ha létrejön:
```
sajatJLista.setModel(modell);
```

A szétválasztás lényege: adatot mindig a modellbe rakunk, innen törlünk, itt manipuláljuk, megjelenni azonban a `JList`-ben jelenik meg (mégpedig az adott objektum `toString()`-je), itt tudjuk kiválasztani a manipulálandó elemet, stb.

Természetesen ugyanez a helyzet, ha más típusú adatokat akarunk kezelni. ☺