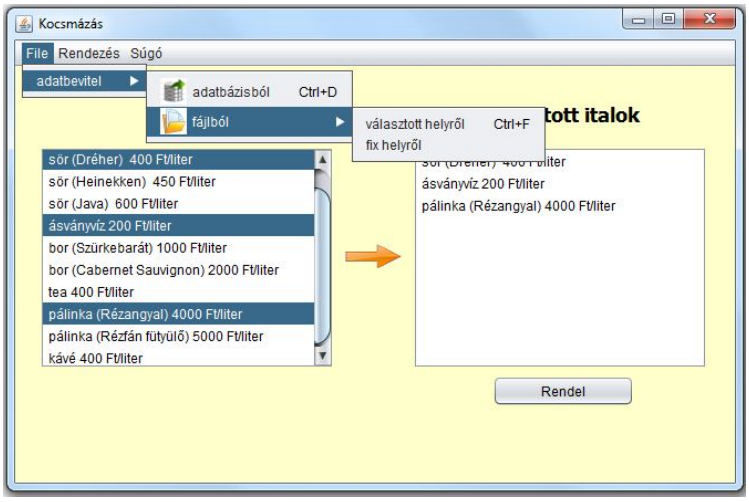
TUDNIVALÓK:
Most is és a következő gyakorlatokon is – akkor is, ha külön nem emeljük ki – az órán meg nem oldott feladatok HÁZI FELADAT-ként megoldandóak!!!
Ez fontos a tárgy sikeres teljesítéséhez!

1. feladat:

Na végre, mégiscsak kocsmázunk. ☺



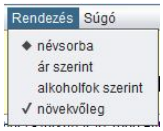
A 650*400-as belső felületű alkalmazásban választhassuk ki, hogy honnan olvassuk be az adatokat.

Ahogy látható, az itallapon szereplő italok között vannak alkoholos és nem alkoholos italok is.
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.

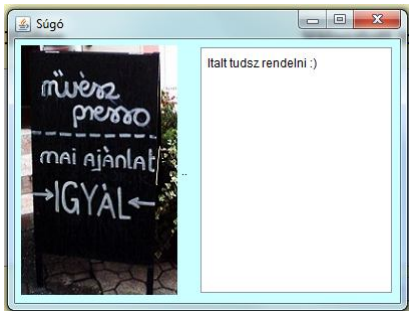
A nyíllal jelölt gomb hatása most csak annyi, hogy láthatjuk, milyen italokat választottunk.

A Rendel gomb hatását ld. a feladatkiírás végén szorgalmiként.



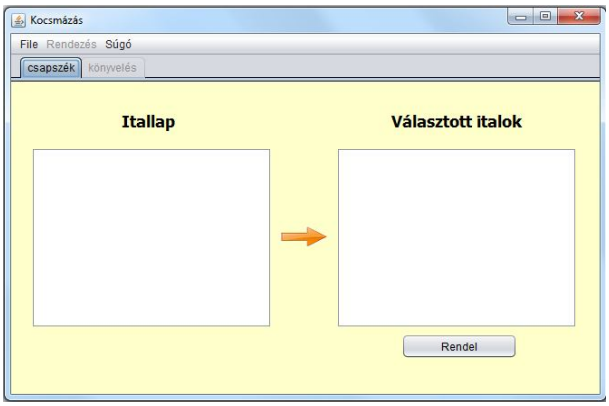
A Rendezés menüpont hatására lehessen rendezni a kiválasztott módon.

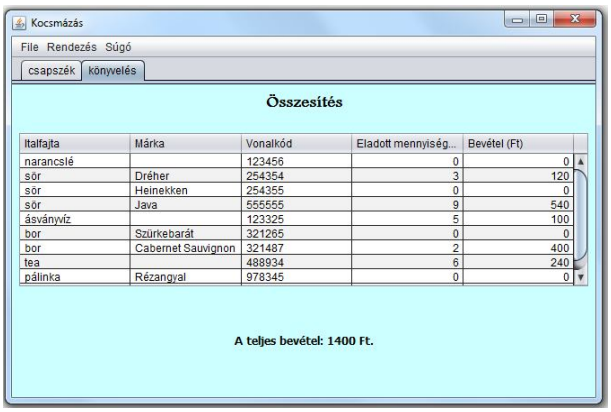
A Súgó menüpont hatása:



A Rendel gomb hatása: próbálja megoldani a rendelést – ez kicsit rá van bízva a fantáziájára, de némi ötlet: törölje ki a választott italok listáját, és írja ki a fizetendő összeget. Azt is próbálja megoldani, hogy ne a literenkénti árakat összegezze, hanem a ténylegesen rendelt mennyiségek árát. Ehhez az egyszerűség kedvéért hozzárendelhet default mennyiségeket az egyes italokhoz. Ekkor azt is megoldhatja, hogy a választott italok listája valahogy így nézzen ki:

Ha már ilyen szépen belejött, ezt is megoldhatja ☺:





Segítség az adatbázis-kezeléshez:

Először létre kell hoznunk a kapcsolatot (`Connection`):

```
// az adatbázis driver meghatározása
Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
// az adatbázis definiálása
String url = "jdbc:derby://localhost:1527/ADATBAZIS_NEV";
// kapcsolodas az adatbázishoz
kapcsolat = DriverManager.getConnection(url, "user", "password");
```

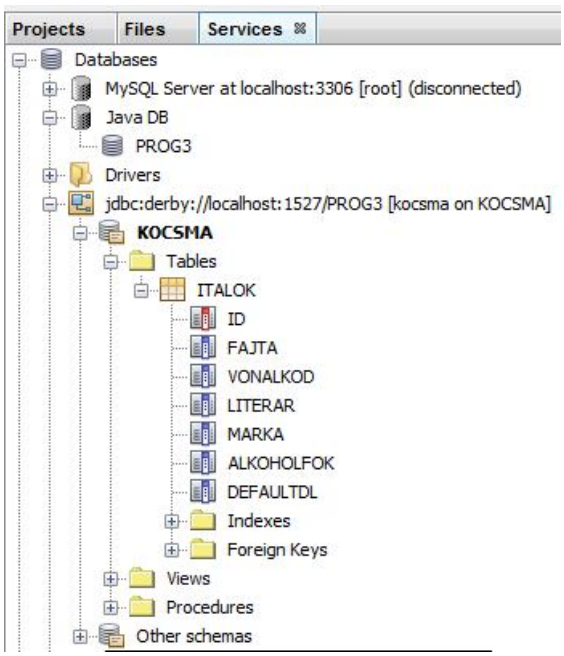
Ha van kapcsolat, megkérjük, hogy hozzon létre egy `Statement` típusú utasításobjektumot, majd ezt az utasításobjektumot kérjük meg arra, hogy hajtsa végre a megadott sql utasítást.

```
String sqlUtasitas = "…";

utasitasObjektum = kapcsolat.createStatement();

eredmenyHalmaz = utasitasObjektum.executeQuery(sqlUtasitas);
```

Ezután végigiterálhatunk az eredményhalmazon (`while (eredmenyHalmaz.next()){}`) és egyenként lekérhetjük a megfelelő attribútumokat.
Végül minden megnyitott objektumot le is kell zárni. (Emiatt érdemes a try fejében megnyitni őket.)



Ez még csak arra elég, hogy megírjuk a beolvasást, de még nincs adatbázis, és még nem kapcsolódtunk az adatbázis-szerverhez.

Ennek beállításához jó segítséget nyújt a https://netbeans.org/kb/docs/ide/java-db.html tutoriál, de itt is lesz róla néhány szó.

A Services fülön tudjuk létrehozni az adatbázist (Java DB – jobb egérgomb – Create Database):

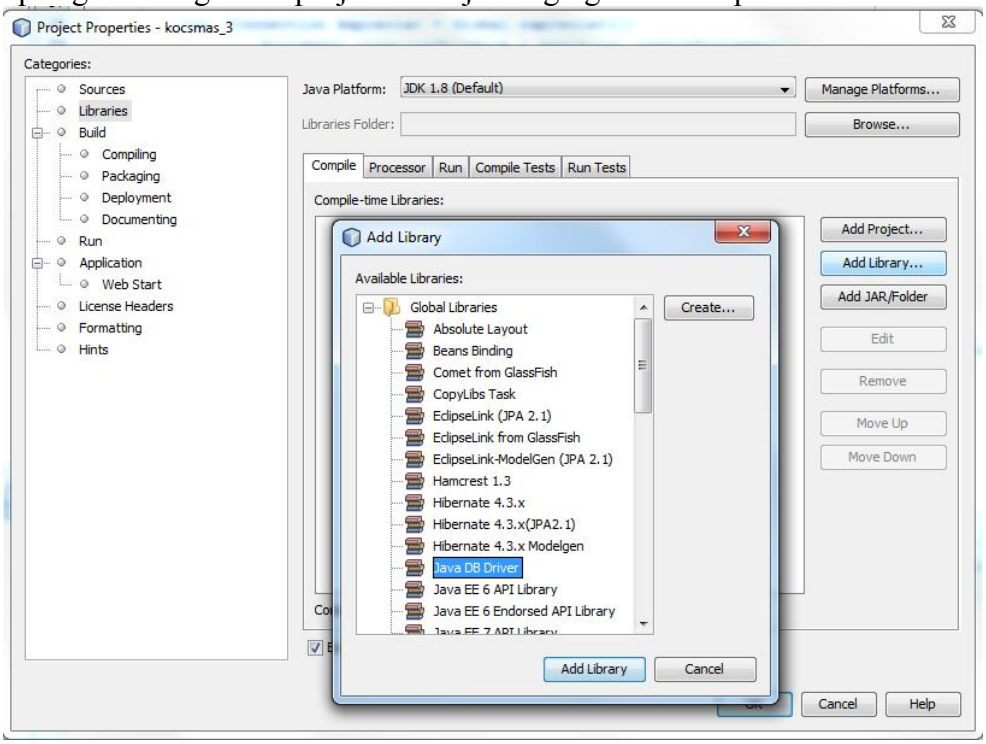
Ha kapcsolódtunk hozzá (Connect), akkor aktívvá válik az Execute Command menüpont.
Ennek hatására megnyílik egy szerkesztő-ablak, itt meg tudjuk írni az sql parancsokat, és végre is tudjuk hajtatni.

Táblanév – jobb egérgomb – View Data: hatására táblázatos formában is láthatjuk az adatokat.

Ezek után boldogan futtathatjuk az alkalmazást, ámde még csalódás ér:
`java.lang.ClassNotFoundException: org.apache.derby.jdbc.EmbeddedDriver`

Ilyen hibaüzenet esetén legelső dolgunk, hogy felkeressük google barátunkat. ☺

Egyébként pedig ez a megoldás: projektnév – jobb egérgomb – Properties és:



2. feladat

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.)

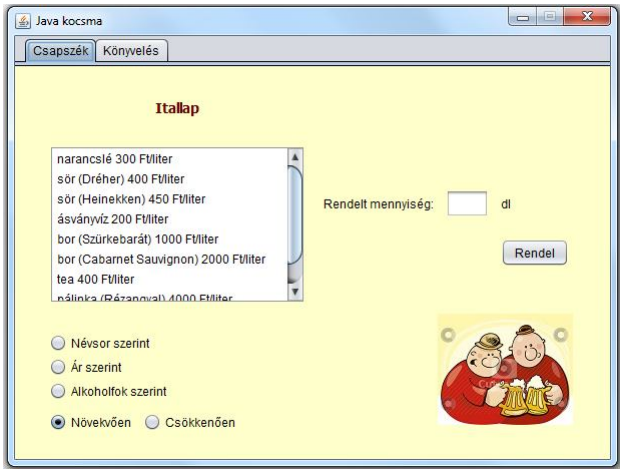
Folytassa az előző feladatot, és
a/ rendeljünk is az itallapról!

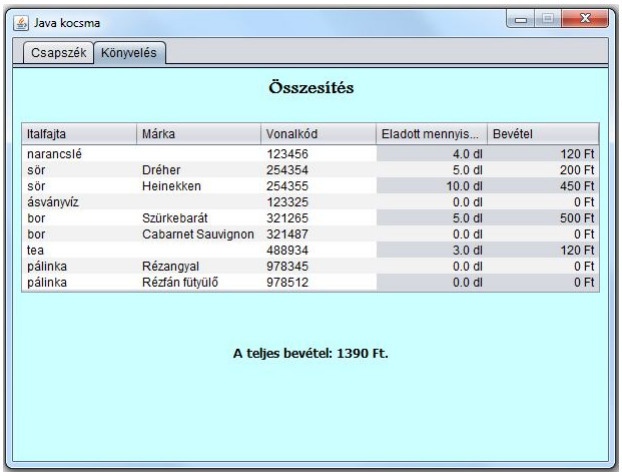
A rendelés gomb hatására írja ki a fizetendő összeget.
Egyelőre egyetlen ember rendeléséről van szó, ezért a fizetendő összeg az egyes rendelésekkor fizetendő értékek összege.

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!

3. feladat

Ha még nem oldotta meg a múltkori feladatsorból, akkor itt az ideje. Ha már megoldotta, akkor foglalkozzon a következő feladattal.

Ennek a korábbi feladatnak a „grafikásítása” lesz a cél.
Az osztály-leírásokat, a hivatkozott ujsagiras() metódus leírását megtalálja a korábbi feladatsorokban (gyak_2 feladatsor, 3.a, 3.b feladat).
Emlékeztetőül:

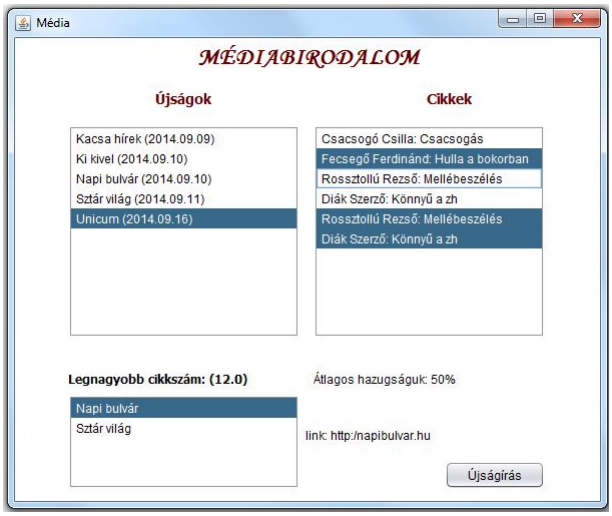
Egy **újság**ot egyértelműen jellemez a *neve* és *megjelenési dátum*a (sima String, de próbálkozhat igazi dátummal is, ha kedve tartja). Az újság *cikket közöl()*, vagyis a metódus paraméterében megadott cikket hozzáadja az újságban megjelent *cikkek listájá*hoz (úgy oldja meg, hogy minden cikk csak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

a/ A feladat „egyszerűbb” változata: Az adatokat fájlból olvassuk.



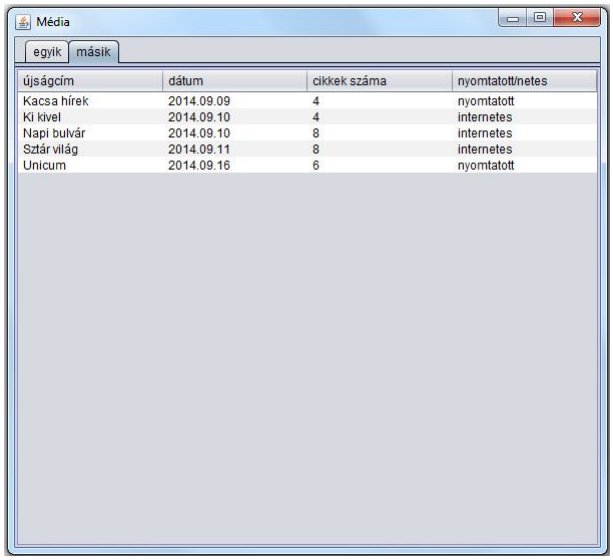
Írassuk ki az újságokat névsorba rendezve.
(Az ábrán látható frame mérete: 600 × 500.)

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.

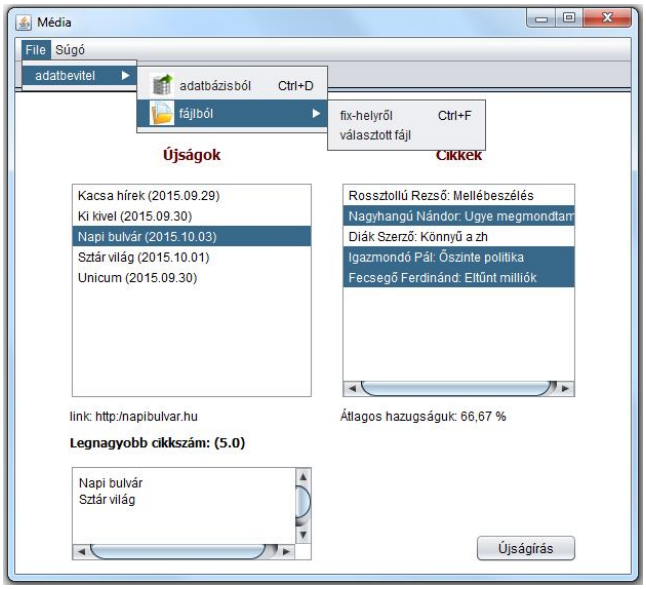
Az újságokat tartalmazó lista alatt azoknak az újságoknak a nevét lehessen olvasni, amelyekben a legtöbb cikk jelent meg. Ebből a listából is csak egyetlen nevet lehessen választani, és ha ez internetes, akkor a lista mellett jelenjen meg az újság linkje, ha nyomtatott, akkor a példányszáma.

Oldja meg, hogy regiszteres módon tudjuk kezelni az újságokat. Az első fülre kattintva az előző felületet lehessen látni, a másodikra kattintva 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.

b/ A kicsit összetettebb változat:



Választástól függően vagy fájlból vagy adatbázisból olvassa be az adatokat – az újságnevek névsorba rendezetten jelenjenek meg a listafelületen.

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. Ekkor persze illene legalább annyit tenni, hogy a gomb feliratával jelezzük, hogy épp mit választunk, pl. így:

```
fajlValaszto.setApproveButtonText
(„Cikkbevitel”);
```

(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 használja ki, hogy az adatbevitel egy interfész implementálásával is megoldható, és mivel ugyanazt az interfészt implementáljuk különböző módokon, ezért a példányosításokon kívül gyakorlatilag semmit sem kell változtatni az egyes beviteli módokban (mármint a panelen, a beviteli osztályokat nyilván meg kell írni).

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 átlag-hazugság értéke.

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

Használjon igazi dátumokat.

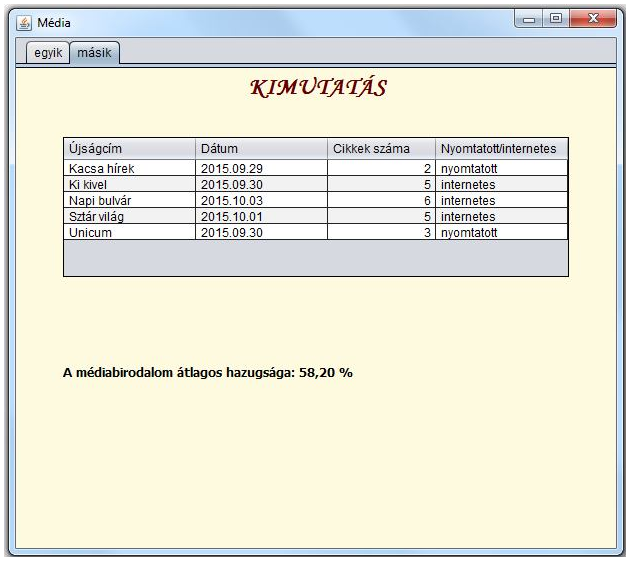
Segítség:
Az adatot `Date` típusúra kell deklarálni.
Beolvasás:
`String`-ből `Date`:
```
datum =new SimpleDateFormat(datumFormatum).parse(stringAdat);
```
ahol a `datumFormatum` a beolvasandó dátum formátuma, esetünkben "yyyy.MM.dd".

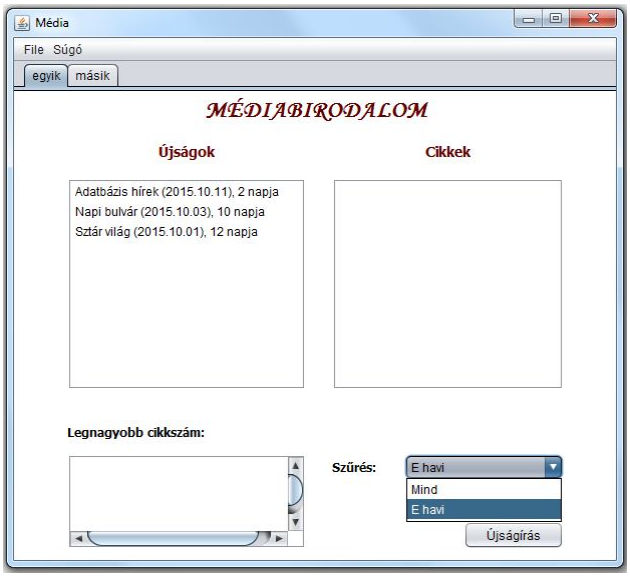
Az `Ujsag` osztályban pedig a dátum `String` alakját kell előállítani.
Ezt így lehet: `new SimpleDateFormat("yyyy.MM.dd").format(datum);`

(Majd még részletesebben is lesz szó dátumkezelésről, de ehhez a feladathoz ennyi is elég.)

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





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.

4. feladat:

Ha még mindig van kedve, energiája, találjon ki valami értelmes feladatot, és 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 adatmodellt, a felületet és a vezérlést. Ez a helyzet a listakezeléssel is. Az adatmodell esetünkben egy 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 listához (`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:
```
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. ☺