4. gyakorlat

Feladat

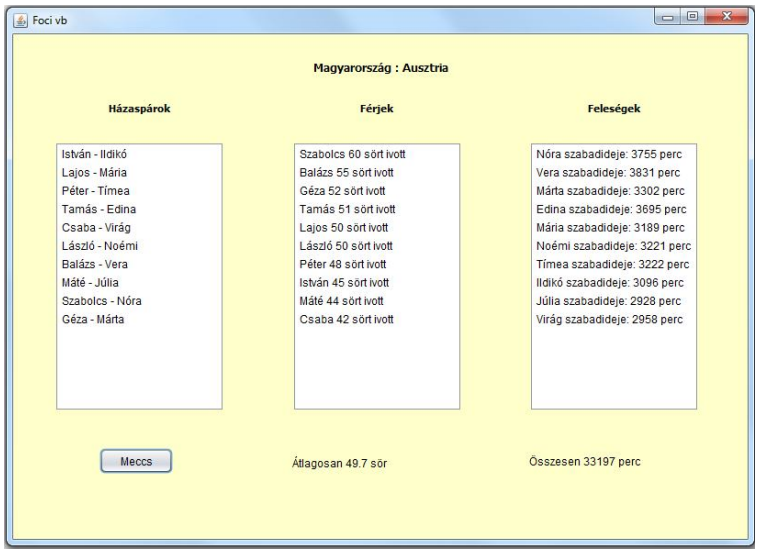
Nem tudom, néz-e még valaki focit Magyarországon, de a tavalyi foci EB-n is életbe lépett egy korábbi szabályzat feleségek és barátnők számára. Ennek szellemében írjon egy EB-szimulációt.
A szimulációban házaspárok vesznek részt. A programban résztvevő minden egyes **ember**nek van *neve*, és mindegyikükre jellemző a *meccsnézés()*, de teljesen eltérő módon, az viszont közös, hogy ennek során a megnézett meccs (vagyis a metódus paramétere) bekerül a *megnézett meccsek* listájába. Természetesen kétszer nem lehet megnézni ugyanazt a meccset.
A **férj**ek meccsnézése során a meccsek közben elfogyasztott *sörök száma* növekszik, mégpedig ha jó a meccs, akkor egy, az összes férjre egyformán jellemző *darabszám*mal, ha nem jó, akkor pedig ugyancsak egy egyformán jellemző, de *másik darabszám*mal.
A **feleség**ek esetében a metódus hatására a *szabadidejük mennyisége* növekszik a paraméterben adott meccs hosszával.

A **házaspár**okat egy férj és egy feleség alkotja ☺, és *meccsnézés()*ük során mindkét fél „nézi” a meccset.

Egy **meccs** megadásához két csapat kell. Jellemző rá a *meccshossz()*, amely a minden meccsre egyforma értékű *játékidő* és a *ráadás* összege.
Végezetül egy **csapat** a *nev*ével jellemezhető. Jelenleg csak ennyi érdekes belőle, de nem kizárt, hogy valamikor később még bővíteni lehet majd.

Olvassa be a házaspárok és a csapatok adatait a mellékelt adatfájlokból (a házaspárokat tartalmazó fájl szerkezete soronként: férj;feleség). A program indulásakor azonnal jelenjen meg a házaspárok névsora, illetve külön-külön a férjek is és a feleségek is. Már induláskor is látható a sörök és szabadidők értéke (nulla), de az átlag, összeg és persze, az aktuális meccs nem. Ezek csak az első meccsnézés után válnak láthatókká.

A Meccs feliratú gomb hatása:
A panel tetején megjelenik az aktuális meccs – ez úgy áll elő, hogy véletlenszerűen kiválasztja a meccshez tartozó két csapatot (figyeljen rá, hogy egy csapat ne játsszon saját magával, de persze, több meccs is lehet ugyanazon két csapat között). Állítsa be a meccs ráadás-idejét – ez egy 0 és adott határ közötti véletlen érték, majd azt is, hogy jó-e a meccs – ennek esélye valahány százalék.
Ezek után a házaspárok véletlenszerűen „eldöntik”, hogy nézik-e a meccset – valahány százalék az esélye annak, hogy igen. Ennek hatására a férjekhez és feleségekhez kiírt adatok is változnak, illetve az átlag és összeg értékek is. A férjek, feleségek a két külön listában is „egymás mellett” legyenek, lehetőleg a házaspár férj tagjának sörszáma szerinti csökkenő sorrendben. A férjek névsora alatt jelenjen meg az összes férjre vonatkoztatott átlagosan fogyasztott sörszám, a feleségek névsora alatt pedig a feleségek összes szabadideje.



A felület belső mérete: 800*550.

1. Módosítsa úgy a feladatot, hogy az adatokat adatbázisból olvassuk. Használja a beépített Derby adatbázis-kezelőt:

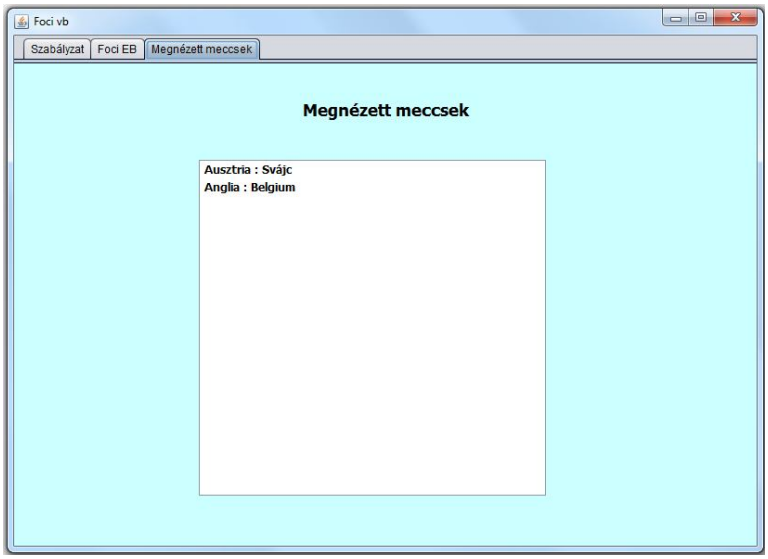
2. Módosítsa a feladatot úgy, hogy ne kelljen „kézzel” felépíteni az adatbázist, és ne kelljen „kézzel” beállítani a drivert.

3. Tesztelje a projektet (na jó, egy kicsit).

a4. Legyenek angol nyelvűek a feliratok.

5. Módosítsa a feladatot úgy, hogy a szabályzat is olvasható legyen az első tabulátor-fül hatására.

6. További módosításként a harmadik tabulátor felületen lehessen látni a megnézett meccsek listáját:



7. Szúrja be adatbázisba is a megnézett meccseket.

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.

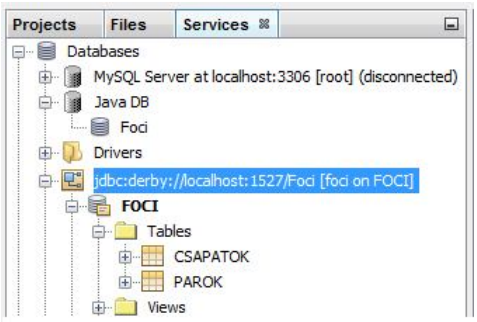
```
String sqlUtasitas = "…";

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.
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):

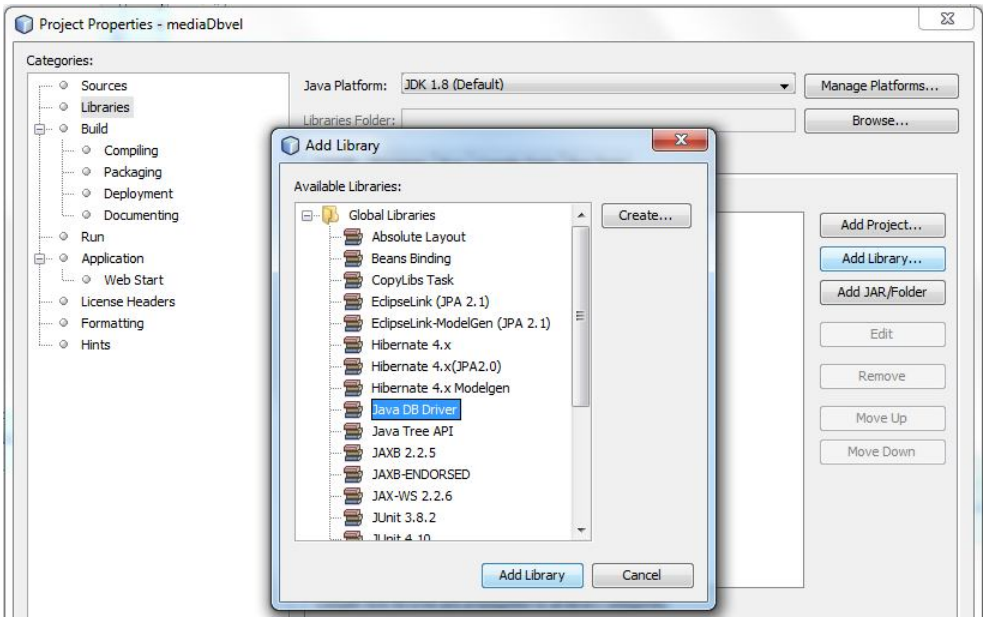
Ha kapcsolódtunk hozzá (Connect), akkor aktívvá válik az Execute Command menüpont.
Ennek hatására megnyílik egy szerkesztő-ablak, itt meg tudjuk írni az sql parancsokat, és végre is tudjuk hajtatni.

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:


Project Properties - mediaDbvel
Categories:
Sources
Libraries
Build
Compiling
Packaging
Deployment
Documenting
Run
Application
Web Start
License Headers
Formatting
Hints
Java Platform:
JDK 1.8 (Default)
Manage Platforms...
Libraries Folder:
Browse...
Add Project...
Add Library...
Add JAR/Folder
Edit
Remove
Move Up
Move Down
Add Library
Available Libraries:
Global Libraries
Absolute Layout
Beans Binding
CopyLibs Task
EclipseLink (JPA 2.1)
EclipseLink-ModelGen (JPA 2.1)
Hibernate 4.x
Hibernate 4.x(JPA2.0)
Hibernate 4.x Modelgen
Java DB Driver
Java Tree API
JAXB 2.2.5
JAXB-ENDORSED
JAX-WS 2.2.6
JUnit 3.8.2
Create...
Add Library
Cancel