Tartalomjegyzék

BattleSudoku

A kezdeti fájlokat itt találjátok: BattleSuduku2.zip (ez a múltheti gyak megoldásait tartalmazza, kicsit átstruktúrálva).

Mezõ átszínezés

- Írjátok át a **Field** osztályt, hogy tudjon piros háttérrel és háttér nélkül is kirajzolni.
  - Ezt egy **boolean** adattag bevezetésével a legkönnyebb megoldani.
  - Legyen ennek a neve **selected** és alapból legyen hamis, de írjatok két metódust, egyik amelyik ezt true-ra állítja, másik amelyik false-ra (ezeknek a metódusoknak nem kell kapnia semmit). Legyen a metódusok neve **select** és **deselect**.
- Teszteljétek ezt le úgy, hogy a **GameBoard initBoard** metódusában valamelyik mezõnek meghívjátok a **select** metódusát. Ehhez az kell, hogy a **drawables** tömb valahanyadik elemét **Field** típusúvá konvertáljuk, mivel ezeknek a típusa **Drawable**, aminek csak **draw** metódusa van, **select** nincs:

```
((Field)drawables[5]).select();
```

MouseControl

- Írunk egy osztályt, ami majd az egérrel való irányításért felel. Ez már el van kezdve a **MouseController** osztályban és hozzá van adva a Game-ben a **JFrame**-hez.
- Írjatok konstruktort neki ami megkapja a Game objektumot és mentse el magának adattagként. Ehhez a **Game.java**-ban is a konstruktorhívásánál argumentumként be kell írnotok a **this**-t, hogy átadjuk neki a **Game** objektumot.
- Ha rákattintunk egy mezõre akkor színezze át az adott mezõt. (Ehhez lehet, hogy a korábbi osztályokon is változtatni kell.) Eddig csak annyit tesz, hogy az (x, y) koordinátát írja ki ahova kattintunk.
  - (Opcionális) A **Field** osztályban írjunk egy **isInside** metódust, ami két **int**-et kap (rx, ry)-t és eldönti, hogy az ezáltal meghatározott pont az adott **Field**-en (**this**-en) belül helyezkedik-e el.
  - Hozzatok létre a **GameBoard**-ban egy metódust **getFieldByCoord** névvel, ami két **int**-et kap (x, y) és adja vissza azt a **Field** objektumot aminek a belsejében van ez az (x, y). (Használjátok az **isInside** metódust, ha megírtátok.)
  - Ha ez megvan, akkor írjatok egy **selectByCoord** és egy **deselectByCoord** metódust a **GameBoard**-ban, ami paraméterként kap két **int**-et (x, y) és meghívja a **select** vagy a **deselect** metódusát annak a **Field**-nek ami ott van. (Használjátok az elõzõ metódust.)

- Majd hozzatok létre a **Game**-ben egy metódust **mouseClick** névvel, ami két **int**-et kap. Ezek a koordináták ahol az egérkattintás történt. Ez hívja meg a **gameBoard** változó **selectByCoord** metódusát az adott **int**-ekkel. Valamint hívja meg saját magán a **repaint**() metódust, ezzel újrarajzolja a táblát, hogy a módosított rajzolás érvényesüljön.
- Végül a **MouseController** osztály **mousePressed** metódusában hívjátok meg a mentett **Game** objektum **mouseClick** metódusát a kattintott koordinátákkal. (**e.getX()** és **e.getY**())

Ez így elég túlkomplikáltnak tûnik, mintha csomó metódust hoztunk volna létre teljesen feleslegesen. Cserébe viszont sokkal átláthatóbb / olvasható lett a kód.

Select / deselect

Oldjátok meg, hogy ha egy mezõ ki van választva és újra kattintunk rá akkor **deselect**-elje. Próbáljátok minél átláthatóbban megoldani, nem baj ha sok metódust kell hozzá csinálnotok.

SQLite próba

Ez a rész csak azoknak szól akik érdekeltek adatbázis kezelésben, nem lesz számonkérve, a nagyháziban sem kötelezõ használni.

Kézzel

sqlite3-on keresztül hozzátok létre az alábbi táblázatokat, és töltsétek fel õket pár próba adattal:

- *users*: *id* (sorszám), *username*, *password*, (esetleg további adatok a userrõl)
- *highscore*: *id* (sorszám), *user*, *points*

Java-ból

Egy olyan megoldást fogunk használni amihez semmilyen sql telepítés nem lesz szükséges. Az omnibuson van sqlite3 így azzal játszhattok, de saját gépen nem kell majd telepítenetek, hogy javaban tudjátok használni.

- Elsõ lépés, hogy töltsétek le innen a *sqlite-jdbc-3.7.2.jar* file-t.
- Majd hozzatok létre egy új projectet. Ebbe dependency-ket importáljátok ezt a .jar-t. (A projectet kijelölve, jobb klikk és properties, majd java build path, itt a libraries fül, végül add external jars gomb.)
- Ha a souce-t is hozzá akarjátok rendelni (hogy lássatok osztályleírást meg ilyesmit):
  - Töltsétek le a korábbi linkrõl a javadoc.jar-ra végzõdõ file-t, a megfelelõ verzióhoz.
  - Nyissátok le a projectet, majd referenced libraries itt jobb klikk valamelyikre, properties, gyõzõdjetek meg, hogy a java source attachment van kijelölve bal oldalt. Ha igen, akkor external jar gomb, és jelöljétek ki azt a source file-t, ami az épp kiválasztott library-hez tartozik.
- Végül próbáljátok ki ezt a kódot, ha minden jól ment akkor a project mappátokban keletkezik egy sample.db file.

```
import java.sql.*;

public class SqlTest
{
  public static void main(String[] args) throws ClassNotFoundException
  {
    // load the sqlite-JDBC driver using the current class loader
    Class.forName("org.sqlite.JDBC");

    Connection connection = null;
    try
    {
      // create a database connection
      connection = DriverManager.getConnection("jdbc:sqlite:sample.db");
      Statement statement = connection.createStatement();
      statement.setQueryTimeout(30);  // set timeout to 30 sec.

      statement.executeUpdate("drop table if exists person");
      statement.executeUpdate("create table person (id integer, name string)");
      statement.executeUpdate("insert into person values(1, 'leo')");
      statement.executeUpdate("insert into person values(2, 'yui')");
      ResultSet rs = statement.executeQuery("select * from person");
      while(rs.next())
      {
        // read the result set
        System.out.println("name = " + rs.getString("name"));
        System.out.println("id = " + rs.getInt("id"));
      }
    }
    catch(SQLException e)
    {
      // if the error message is "out of memory",
      // it probably means no database file is found
      System.err.println(e.getMessage());
    }
    finally
    {
      try
      {
        if(connection != null)
          connection.close();
      }
      catch(SQLException e)
      {
        // connection close failed.
        System.err.println(e);
      }
    }
  }
}
```

Ez egy olyan kód ami vigyáz mindenre. Ha nem tudja elérni az adatbázist nem hal meg teljesen, se akkor ha nem tudja módosítani, ha túl sokat kell várnia mire bármit is kapna belõle stb.