Hello Wien – adatbázis-kezelés része

A feladat adatbázisa katonák adatait tartalmazza, mégpedig a katonák nevét és rangját.
A Java megvalósításban nyilvánvalóan szükségünk lesz egy `Katona` osztályra. A további bővíthetőségi lehetőséget szem előtt tartva a rangok alapján különböző utód-osztályokat határozunk meg (`Lovas`, `Gyalogos`, `LovasParancsnok`, `GyalogosParancsnok`, stb.).
Az adatbevitel szempontjából csak a konstruktor az érdekes – a `Katona` osztály konstruktorának paraméterlistáján csak a katona neve szerepel.

Az utód példányokat a név és rang alapján egy gyártó függvény (gyártó metódus) hozza létre. Ez az úgynevezett *factory method* tervezési minta.
Egy másik tervezési mintát is felhasználunk, ugyanis elég, ha a gyártó csak egyetlen példány jön létre. Ezt az egyetlenséget biztosítja a *singleton* tervezési minta.

A hosszú, teljes megoldást tartalmazó leírásban három megvalósítási módot is említettem, most csak az általam legszebbnek tartott megoldást részletezem. E szerint a létrehozott példányokat listába olvassuk (mert ez a legáltalánosabb beolvasási mód), a rangokat pedig `enum`-ként kezeljük (mert ez kényelmesebben kezelhető, mint ha string adatokkal dolgoznánk), a `default` ágon pedig kivételt dobunk (mert így akár absztrakt is lehet a `Katona` osztály).

Magyarázatokat már nem fűzök hozzá, remélem, a kód magyarázza magát.

```
public interface AdatInput {

    public List<Katona> katonaListaBevitel() throws Exception;
}
```

```
public class AdatBazisInput implements AdatInput{

    private Connection kapcsolat;

    public AdatBazisInput(Connection kapcsolat) {
        this.kapcsolat = kapcsolat;
    }
```

```
    @Override
    public List<Katona> katonaListaBevitel() throws Exception {
        List<Katona> katonak = new ArrayList<>();
        String sqlUtasitas = "SELECT * FROM KATONAK";
        String nev, rang;
        Katona katona;
        try(Statement utasitasObj = kapcsolat.createStatement();
                ResultSet eredmenyHalmaz = utasitasObj.executeQuery(sqlUtasitas)
                ){
                while (eredmenyHalmaz.next()) {
                    nev = eredmenyHalmaz.getString("nev");
                    rang = eredmenyHalmaz.getString("rang");
                    //kivesszük a string-ből a fölösleges szóközöket
                    // A reguláris kifejezés jelentése: az összes "fehér szóköz"
                    rang = rang.replaceAll("\\s+","");
                    KatonaGyar.RANGOK rangTipus =
                            KatonaGyar.RANGOK.valueOf(rang);
                    katona = KatonaGyar.getPeldany().getKatona(nev, rangTipus);
                    katonak.add(katona);
            }
        }
        return katonak;
    }
}
```

```
public class KatonaGyar {

    private static KatonaGyar peldany;

    public KatonaGyar() {
    }

    public static KatonaGyar getPeldany() {
        if(peldany == null){
            peldany = new KatonaGyar();
        }
        return peldany;
    }

    public static enum RANGOK {gyalogos, gyalogosparancsnok,
                               lovas, lovasparancsnok, zaszlos}
```

```
    public Katona getKatona(String nev, RANGOK rang) throws Exception {
        switch (rang) {
            case gyalogos:
                return new Gyalogos(nev);
            case gyalogosparancsnok:
                return new GyalogosParancsnok(nev);
            case lovas:
                return new Lovas(nev);
            case lovasparancsnok:
                return new LovasParancsnok(nev);
            case zaszlos:
                return new Zaszlos(nev);
            default:
                throw new Exception();
        }
    }
}
```

A `Vezerlo` osztály beolvasó metódusa:

```
    private void beolvasas() throws Exception {
        Connection kapcsolat = kapcsolodas();
        AdatInput adatInput = new AdatBazisInput(kapcsolat);
        List<Katona> katonak =
                adatInput.katonaListaBevitel();
        harcosokPanel.listabaIr(katonak);
    }

    private Connection kapcsolodas() throws ClassNotFoundException, SQLException {
        // az adatbázis driver meghatározása
        Class.forName("org.apache.derby.jdbc.EmbeddedDriver");
        // az adatbázis definiálása
        String url = "jdbc:derby://localhost:1527/HELLOW";
        // kapcsolodas az adatbázishoz
        return DriverManager.getConnection(url, "csata", "csata");

    }
```

A `HarcosokPanel` listába író metódusa:

```
    private RendezhetoListModel<Katona> katonaModell =
                                        new RendezhetoListModel<>();
    public void listabaIr(List<Katona> katonaLista) {
        for (Katona katona : katonaLista) {
            katonaModell.addElement(katona, true);
        }
        lstKatonak.setModel(katonaModell);
    }
```

Ez a megoldás azért is jobb a többinél, mert a vezérlésnek semmi köze ahhoz, hogy a panel hogyan oldja meg a listába írást, vagyis a vezérlésnek (és az adatbeolvasó osztálynak) nem is kellene tudnia, hogy a panel milyen modellt használ.

Végül még egy megjegyzés: bár mivel már többedik alkalommal van szó adatbázis-elérésről, ezért nyilván tudja is, de mégis megemlítem, hogy ne felejtse el hozzárendelni a Library-hoz a Java Db drivert: