DAM. UNIT 4. ACCESS USING COMPONENTS. PART 2. NON ASSESSABLE EXERCISES

DAM. Acceso a Datos (ADA) (a distancia en inglés)

Unit 4. ACCESS USING COMPONENTS

Part 2. JavaBeans Advanced. Non assessable exercises

Abelardo Martínez

Based and modified from Sergio Badal (www.sergiobadal.com)

Year 2024-2025

Aspects to bear in mind

Important

If you look for the solutions surfing the Internet or asking the oracle of ChatGPT you will be fooling yourself. Keep in mind that **ChatGPT is not infallible or all-powerful**.

It is a great tool to speed up your work once you have mastered a subject, but using it as a shortcut when acquiring basic skills and knowledge seriously undermines your learning. If you use it to get solutions or advice on your own, check the proposed solutions carefully as well. Try to solve the activities using the resources we have seen and the extended documentation you will find in the "Virtual Classroom".

Tips for programming

We advice to follow the next coding standards:

- One instruction per line.
- Add comments to make your code clearer and more readable.
- Use the Hungarian notation to recognise the type of variables at first sight.
- If necessary, we strongly recommend using buffer-based solutions.
- Remember that there are several ways to implement a solution, so choose the one you like best.

1. Console mode. Managing a DB library with JavaBeans

Activity (non assessable)

Make the suitable changes to the FOUR beans you created last week to implement the situations described here using an auxiliary bean and JDBC/MySQL.

1.1. JavaBeans objects

Using these beans (same as last week):

- **BEAN 1: BookBean (SOURCE)**
  - Represents a book in a library, available for the students as a loan.
  - TABLE books (idb, name, isAvailable)
- **BEAN 2: FineBean. (LISTENER)**
  - Represents a fine/penalty in a library, to prevent students to keep the books more than the allowed period (15 days).
- **BEAN 3: StudentBean (LISTENER)**
  - Represents a student borrowing/returning a book.
  - TABLE students (ids, name, bannedDays)
- **BEAN 4: LoanBean (LISTENER)**
  - Represents a loan of a book.
  - TABLE loans (idl, ids, idb, startDate, actualReturnDate)

Implement the following features in the application:

1) Create a MySQL database to hold those beans into 3 tables.

2) Create a fifth BEAN called DBBean to deal with the database using JDBC.

Database script (MySQL)

This is the script to create the Library database and some example data to interact with:

```
CREATE DATABASE IF NOT EXISTS ADAU4DBLibrary CHARACTER SET utf8mb4 COLLATE utf

CREATE USER 'mavenuser'@'localhost' IDENTIFIED BY 'ada0486'; --
GRANT ALL PRIVILEGES ON ADAU4DBLibrary.* TO 'mavenuser'@'localhost'; --

USE ADAU4DBLibrary;
```

```sql
CREATE TABLE books(
idb          INTEGER,
name         VARCHAR(20),
isAvailable BIT DEFAULT b'1',
CONSTRAINT boo_idb_pk PRIMARY KEY (idb)
);

CREATE TABLE students(
ids          INTEGER,
name         VARCHAR(20),
bannedDays  INTEGER DEFAULT 0,
CONSTRAINT stu_ids_pk PRIMARY KEY (ids)
);

CREATE TABLE loans(
idl                  INTEGER AUTO_INCREMENT,
ids                  INTEGER,
idb                  INTEGER,
startDate            DATE,
actualReturnDate    DATE,
CONSTRAINT loa_idl_pk PRIMARY KEY (idl),
CONSTRAINT loa_ids_fk FOREIGN KEY (ids) REFERENCES students(ids),
CONSTRAINT loa_idb_fk FOREIGN KEY (idb) REFERENCES books(idb)
);

INSERT INTO books (idb, name) VALUES (1, 'Oracle Databases');
INSERT INTO books (idb, name) VALUES (2, 'Mongo DB Advanced');
INSERT INTO books (idb, name) VALUES (3, 'MySQL Databases');
INSERT INTO books (idb, name) VALUES (4, 'NoSQL Distilled');
INSERT INTO books (idb, name) VALUES (5, 'Everlasting Data');
INSERT INTO books (idb, name) VALUES (6, 'BigData for Dummies');
INSERT INTO students (ids, name) VALUES (1, 'Gabriel Adare');
INSERT INTO students (ids, name) VALUES (2, 'John Harris');
```

1.2. JavaBeans events

Recreate these situations:

a) A student borrows a book (when property BookBean.bAvailableForLoan turns to FALSE):

- Insert into **LOANS with** startDate=today
- Update **BOOKS** setting isAvailable to false
- Update **STUDENTS** setting bannedDays to 0 (reset the counter)

b) A student returns a book (when property BookBean.bAvailableForLoan turns to TRUE):

- Update **LOANS** setting actualReturnDate to today
- Update **BOOKS** setting isAvailable to true
- If it’s a delayed return (more than 15 days):
  - Update **STUDENTS** setting bannedDays to 5 days

To check how the components interact, we suggest the piece of code you can find below.

```
        BookBean objBookBean = DBBean.getRandomBook();

        StudentBean objStudentBean = DBBean.getRandomStudent();
        FineBean objFineBean = new FineBean();
        LoanBean objLoanBean = new LoanBean();

        objStudentBean.setBook(objBookBean);
        objBookBean.addPropertyChangeListener(objStudentBean);

        objLoanBean.setBook(objBookBean);
        objLoanBean.setStudent(objStudentBean);
        objBookBean.addPropertyChangeListener(objLoanBean);

        objFineBean.setLoan(objLoanBean);
        objBookBean.addPropertyChangeListener(objFineBean);

        //System.out.println("****** book.setAvailableForLoan(false):");
        objBookBean.setAvailableForLoan(false);
        //System.out.println("****** book.setAvailableForLoan(true):");
        objBookBean.setAvailableForLoan(true);
```