DAM. UNIT 3. ACCESS USING OBJECT-RELATIONAL MAPPING (ORM). DAO. NON ASSESSABLE EXERCISES

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

Unit 3. ACCESS USING OBJECT-RELATIONAL MAPPING (ORM)

Access to relational databases using DAO. 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 MySQL relational database with DAO

Activity (non assessable)

We are going to use the MySQL employee database from UNIT 2, so you can reuse code from this non assessable task. If not already created, run this script:

```sql
CREATE DATABASE ADAU3DBCompany CHARACTER SET utf8 COLLATE utf8_spanish_ci;

CREATE USER 'mavenuser'@'localhost' IDENTIFIED BY 'ada0486'; --
GRANT ALL PRIVILEGES ON ADAU3DBCompany.* TO 'mavenuser'@'localhost';

USE ADAU3DBCompany;

-- Create the employee's table
CREATE TABLE Employee (
taxID         VARCHAR(9),
firstname     VARCHAR(100),
lastname      VARCHAR(100),
salary        DECIMAL(9,2),
CONSTRAINT emp_tid_pk PRIMARY KEY (taxID)
);

-- Insert random employees
INSERT INTO Employee (taxID, firstname, lastname, salary) VALUES ('11111111A', 'José', 'Salcedo López', 1279.90);
INSERT INTO Employee (taxID, firstname, lastname, salary) VALUES ('22222222B', 'Juan', 'De la Fuente Arqueros', 1100.73);
INSERT INTO Employee (taxID, firstname, lastname, salary) VALUES ('33333333C', 'Antonio', 'Bosch Jericó', 1051.45);
INSERT INTO Employee (taxID, firstname, lastname, salary) VALUES ('44444444D', 'Ana', 'Sanchís Torres', 1300.02);
INSERT INTO Employee (taxID, firstname, lastname, salary) VALUES ('55555555E', 'Isabel', 'Martí
Navarro', 1051.45);
```

Then, create classes to handle SELECT, INSERT, DELETE and UPDATE operations on the **Employees** table. Work in a layered structure using **DAO**. Feel free to share your doubts at the UNIT forum. **You can reuse source code from previous tasks**.

ATTENTION: Use the proper exceptions when accessing to databases.

Implement a menu with the following options:

- **Press 0 to "Exit"**
- **Press 1 to "Ask for employees until user enters zero as ID and store them into the DB"**
  - For every employee we need the ID (String without spaces), first name (String with spaces), last name (String with spaces) and salary (Float), added to the ArrayList of employees.
  - Check if the tax ID already exists in the array list. If yes, you must display a message on the screen. You must ask for each value (in loop) until the user enters a valid ID.
  - Once zero is entered as ID, all employees will be saved into the database.
- **Press 2 to "List all employees"**
  - Just read the **MySQL database** and print every employee information.
- **Press 3 to "Remove all employees"**
  - Just delete the employees stored at the **MySQL database**.

Menu example:

```
*****
MENU
*****
================================================
  0. Exit
  1. Add employees
  2. List all employees
  3. Remove all employees
================================================
    Select an option:
```

2. Console mode. Managing a MySQL relational database with DAO

Activity (non assessable)

Using a **MySQL** database, run this script:

```sql
CREATE DATABASE ADAU3DBCars CHARACTER SET utf8 COLLATE utf8_spanish_ci;

CREATE USER 'mavenuser'@'localhost' IDENTIFIED BY 'ada0486'; --
GRANT ALL PRIVILEGES ON ADAU3DBCars.* TO 'mavenuser'@'localhost';

USE ADAU3DBCars;

-- Create table of drivers
CREATE TABLE Drivers (
DNI     VARCHAR(9),
name    VARCHAR(40),
age     INTEGER,
PRIMARY KEY(DNI)
);

-- Create table of race cars
CREATE TABLE Racecars (
carID   VARCHAR(10),
brand   VARCHAR(20),
price   DOUBLE,
DNI     VARCHAR(9),
PRIMARY KEY(carID),
FOREIGN KEY (DNI) REFERENCES Drivers (DNI)
);

INSERT INTO Drivers VALUES('11111111A','Carlos Sainz',30);
INSERT INTO Drivers VALUES('22222222F','Luis Moya',50);
INSERT INTO Drivers VALUES('12345678A','Ana Prost',27);
INSERT INTO Racecars VALUES('1111AAA','Toyota',30000,'11111111A');
INSERT INTO Racecars VALUES('2222EEE','Subaru',40000,'11111111A');
INSERT INTO Racecars VALUES('3333III', 'Renault', 25000, '12345678A');
```

Then, create classes to handle SELECT, INSERT, DELETE and UPDATE operations on the **Drivers** table. Work in a layered structure using **DAO**. Feel free to share your doubts at the UNIT forum. **You can reuse the code from the previous one**.

ATTENTION: Use the proper exceptions when accessing to databases.

Implement a menu with the following options:

- **Press 0 to “Exit”**
- **Press 1 to “Add drivers (until 0 as DNI is entered) and store them into the DB“**
  - For every driver we need the DNI (String without spaces), name (String with spaces) and age (Integer), added to the ArrayList of drivers.
  - Check if the DNI already exists in the array list. If yes, you must display a message on the screen. You must ask for each value (in loop) until the user enters a valid DNI.
  - Once zero is entered as DNI, all drivers will be saved into the database.
- **Press 2 to “Add race cars (until 0 as CarID is entered) and store them into the DB”**
  - For every race car we need the carID (String without spaces), brand (String without spaces), price (Double) and driver DNI (String without spaces), added to the ArrayList of race cars.
  - Check if the carID already exists in the array list. If yes, you must display a message on the screen. You must ask for each value (in loop) until the user enters a valid carID.
  - You must verify that the driver's DNI already exists in the drivers table. If it does not exist, do not allow the insertion of the new race car.
  - Once zero is entered as carID, all race cars will be saved into the database.
- **Press 3 to “List all race cars”**
- **Press 4 to “List all drivers”**
- **Press 5 to “List a race car by driver”**
  - Given the driver's DNI, list his/her data and the cars he/she owns.

- **Press 6 to “Delete a driver”**
  - Given the driver's DNI, delete him/her from the DB deleting also the cars he/she owns.
- **Press 7 to “Delete a race car”**
  - Given the race car's carID, delete this race car from the DB.

Menu example:

```
*****
MENU
*****
========================================================================
  0. Exit
  1. Add Drivers
  2. Add Racecars
  3. List all Racecars
  4. List all Drivers
  5. List a Racecar by Driver
  6. Delete a Driver
  7. Delete a Racecar
========================================================================
    Select an option:
```