Databases

JDBC - Practice n. 4

Homework n. 4

The purpose of this practice is to write the missing parts of a simple Java application that makes use of the Java DataBase Connectivity (JDBC) interface to access a database.

Attention. The code developed during the practice, and eventually completed at home, must be zipped and uploaded on the didactic portal (Portale della didattica) as it is the homework n.4. The deadline for the upload is on the website of the course.

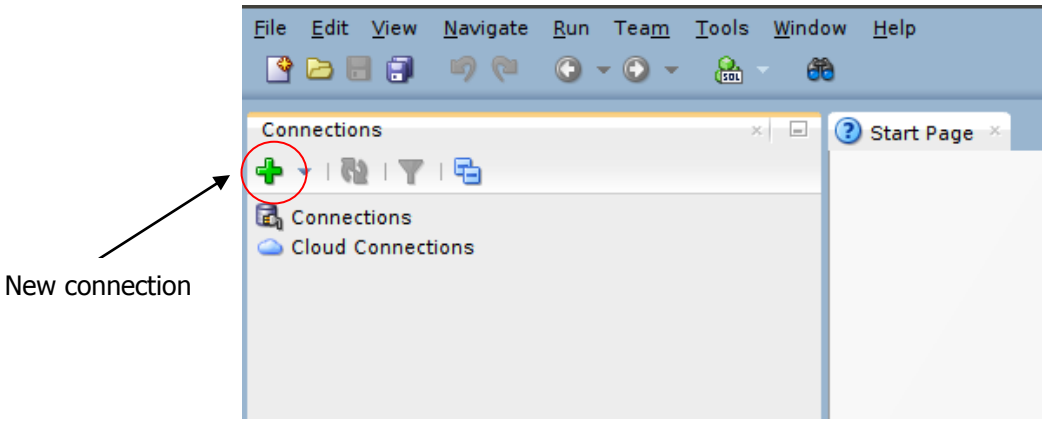
Preliminary steps

Connection to oracle server

During the practice, you can use the SQL Developer software that allows you to connect to the oracle database and to run scripts and SQL commands.

1) Database connection

- Open Oracle SQL Developer program
- Click on New connection



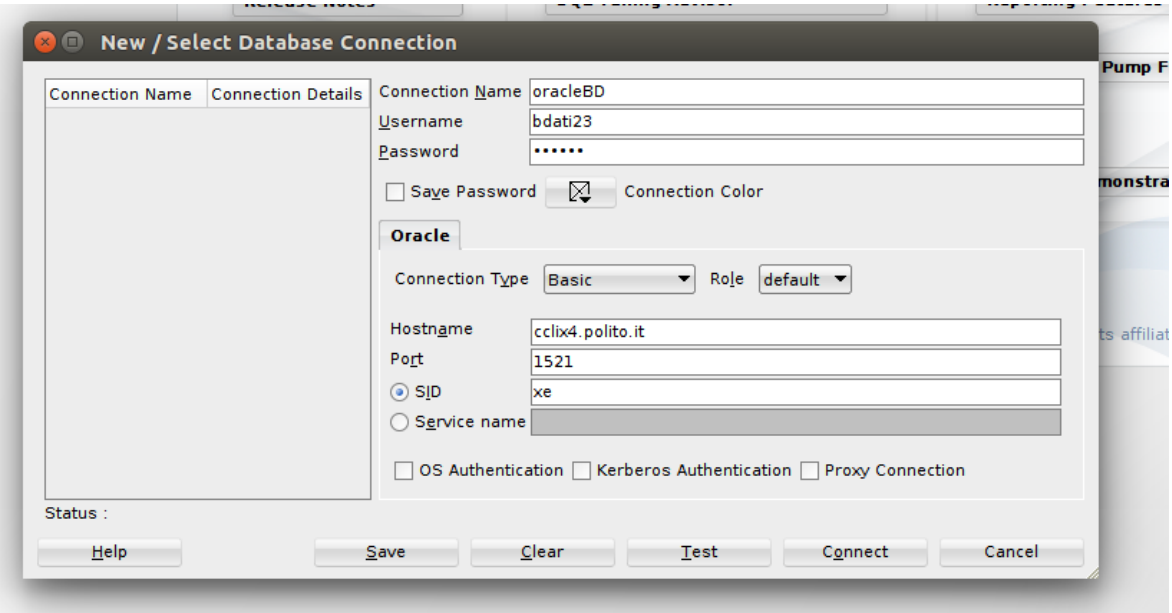
2) Login

To login, insert the following parameters:

- Connection name: oracleBD
- Username: bdati*XY*
  - *XY* last two numbers of the used PC
- Password: orac*XY*
  - *XY* last two numbers of the used PC
- Hostname: cclix4.polito.it
- Port: 1521
- SID: xe

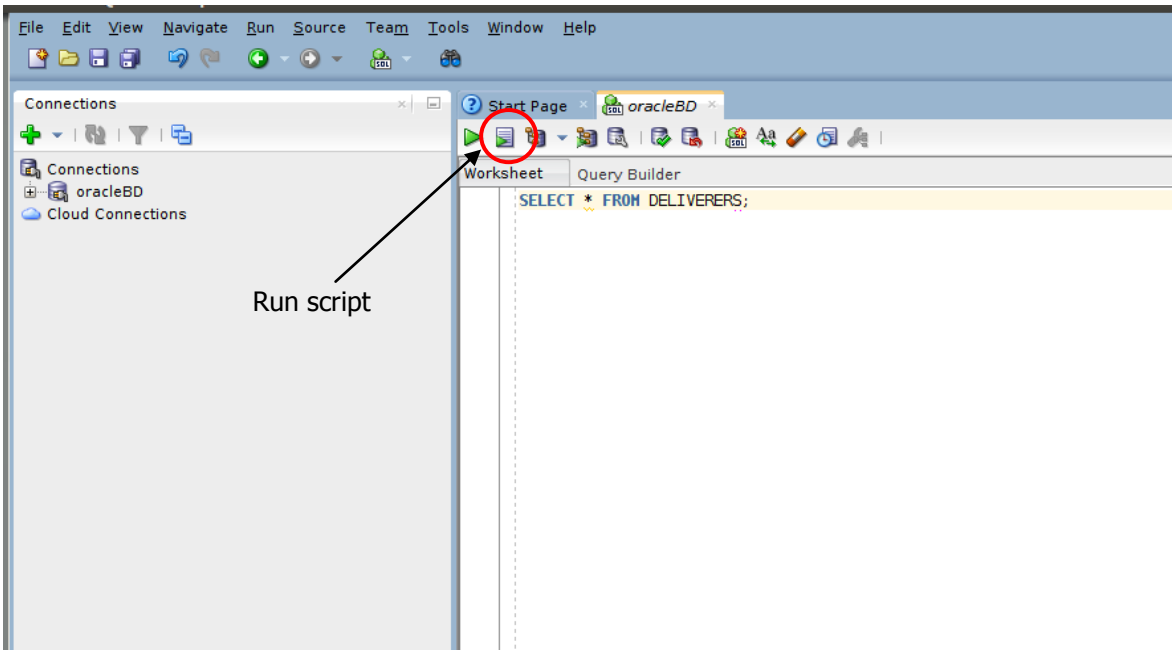
For example, if you are working on PC number 23, you will use username bdati23 and password orac23.

N.B.: if you are working on pc numbers [01, …, 09], DON'T use the "0" in the usernames/passwords: user=[bdati1, …, bdati9] and pass=[orac1, …, orac9].



Write and execute SQL scripts

Write the SQL script on the Worksheet and execute it clicking on "Run script". You can save a script or load an existing one by using commands in the "File" menu bar.



Database Creation

[TO DO ONLY THE FIRST TIME]

- Download the script **practicejdbcOracle.sql** from the web page of the course: http://dbdmg.polito.it/wordpress/teaching/databases/
- Open and execute the script using SQL Developer.

1. Description of the Computer Science Courses database

The *Computer Science Courses* database contains information about the courses of a small company that provides computer science classes. The database stores information about available courses, clients and enrollment of the clients to the courses.
The E-R model in Figure 1 represents the conceptual model of this database.

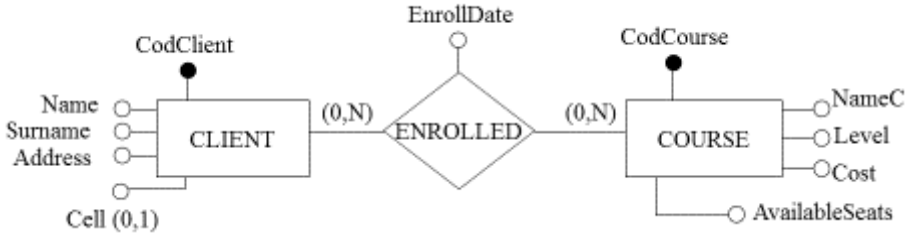


Figure 1. E-R schema of the database

The logical schema of the database is the following (primary keys are underlined); the * symbol means that attribute is optional:

CLIENT (CodClient, Name, Surname, Address, Cell*)

COURSE (CodCourse, NameC, Level, AvailableSeats)

ENROLLED (CodClient, CodCourse, EnrollDate)

Each client is identified by a code and characterized by name, surname, address and mobile number (if available).

The courses are identified by a code and are characterized by a name, the level (an integer value between 1 and 4) and the number of the available seats. **Note.** The AvailableSeats attribute contains information about the number of seats still available. Its value is updated after each new registration.

The enrolled table stores information regarding the courses in which every client is enrolled, along with the enrollment date. Each client may be enrolled in one or more courses. **Attention.** A client can enroll for a course only if there is at least one seat available (AvailableSeats greater than or equal to 1).

The **practicejdbcOracle.sql** script creates tables just described and loads some initial data.

2. Notes on compilation and execution of Java program.

The class implementing the Oracle JDBC driver to use to connect to the database is **oracle.jdbc.driver.OracleDriver**. This class is included in the Java library **classes12.jar** that is available on the course website. This library should be included among the libraries in your Eclipse project to allow you to access the Oracle database.

To create the connection to the database, the *getConnection(String url, String username, String password)* method must be executed using these 3 parameters:

- url: “jdbc:oracle:thin:@cclix4.polito.it:1521:XE”
- username: “bdati*XY*”
- password: “orac *XY*”

where *XY* indicate the last two numbers of the LABINF station.

3. Application to be implemented

The application consists in a simple Graphical User Interface that lets the user manage the following operations:

- Display information on a client and the list of the courses where he/she is enrolled
- Enroll clients in any of the courses available

The purpose of this practice is to understand how Java allows for interactions with a database. For this reason, we provide you, as a starting point, an Eclipse project that already contains all the necessary classes for implementing the application, in particular those for the GUI management. The Eclipse project is in the **practiceJDBC.zip** file you can download from the course website.

You have to implement some methods, currently empty or containing “static” code, which are used to access the database. You have to implement methods in the it.polito.db.DB class (source file DB.java). All other classes of the project do not require any change.

Before starting to modify the code, you should import the Eclipse project in the working environment (select Eclipse 3.4.1 between the different versions available on LABINF PCs), compile the project and try to run it.

To import the project into Eclipse:

- Open Eclipse
- Select from menu bar File → Import
- Now select “General → Existing Projects into Workspace” and click on “Next”.
- In the following window select as “Select root directory”, the directory where you unzipped the practiceJDBC.zip file and click on “Finish”.

At the end of the above operations, in Eclipse you get a project called **practiceJDBCOracle**. This project is your starting point.

You can find the main java class in the it.polito.Main class. Try to run the project (as a Java Application) to verify that the importation was successful. The code creates a window containing two buttons ("Client Information" and "Enrollment"), associated with the two aforementioned operations. Pressing the first button ("Client Info") will open a second window that allows you to specify the code of a client (CodClient textbox) and, pressing “Info”, it displays the client data and the related courses. The current version of the code displays the same information regardless of the code provided (since it does not access the database). Pressing “Enrollment” in the main window will open another window that allows for enrolling clients to courses. The user may select the course of interest from a drop-down menu, then enter the code of

the client to be enrolled in the text box, and finally, when pressing the "Enroll" button, the client is signed up the client to the selected course. This part, for now, is also "static" (i.e. the program will always say that the enrollment was successful, but nothing happens).

You have to implement the following methods in the it.polito.db.DB (source file DB.java) class in order to make sure that the program connects to the database. For each method there is a brief description of what it is supposed to do.

- *public DB()* – constructor of the it.polito.db.DB class

In the constructor of the DB class you must instantiate / register your JDBC driver used to access the database. For Oracle the JDBC driver is **oracle.jdbc.driver.OracleDriver**

- *public boolean OpenConnection()*

This method creates the connection to the database and stores that connection in a variable of the *conn* class

- *public String getDataClient(long cod_client)*

This method receives as a parameter the code of a client and returns the data of that client as a string. The client information must be obtained by querying the database. The string returned by the method is generated by concatenating the attribute names and the values of those attributes for the specified client. For example, if the selected client has name "Paul", surname "White", address "Random Street 24, Turin" and mobile number "3933570222" then the method returns the string: "`Name: Paul\nSurname: White\nAddress: Random Street 24, Turin\nCell: 3933570222`". Look at the code for more detail.

If there is no client with the requested identification code, the method returns the string "Client does not exist".

- *public List<String> getCourseClient(long cod_client)*

This method receives as a parameter the code of a client and returns the list of courses where the client is enrolled (using a list of strings). Each string is the name of one of the courses.

- *public List<String> getCodCourses()*

This method returns a list of courses for which there is at least one available seat (courses in the *Course* table with the *AvailableSeats* attribute greater than or equal to 1). The list is a list of strings. Each string corresponds to one of the courses selected by the query and contains the code and name of the course. Each string has the following format: "`cod_course - name_course`". Look at the code for more detail.

- *public boolean addEnroll(long codCourse, long codClient)*

This method receives as parameters the code of a course and the code of a client and enrolls the client to the course (insert a new tuple in the database). Enrolling consists in entering the appropriate tuple in the *Enrolled* table. Then, the method has to modify (decrease by one) the value of the available seats for the specified course (update of *Course.AvailableSeats* for the course *codCourse*). The method returns true if the enrollment was successful, false otherwise.

- *public void CloseConnection()*

This method closes the connection to the database.