**Guided task on Accessing relational databases from Java with MySQLWorkbench/XAMPP**

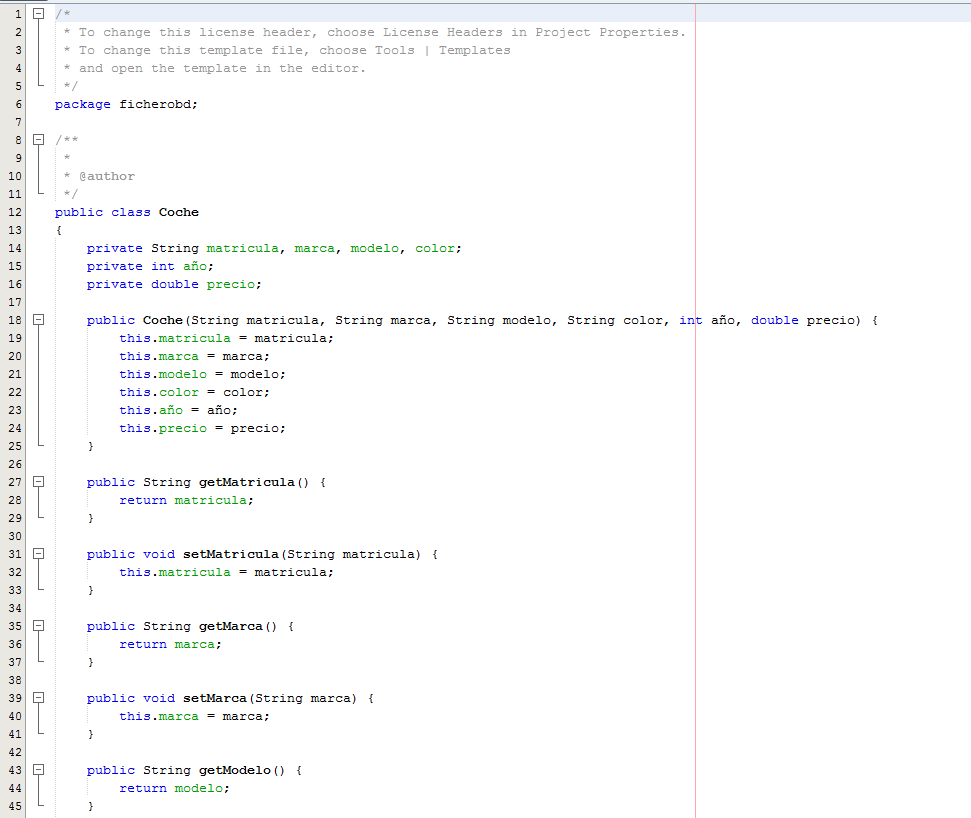
**Preface**

The way to access the MySQL Workbench database from Java has changed, since the updated connector has a different way of registering the drivers, plus you have to put something in it to correctly configure the time zone (Or fix it from the server) It will be explained how to access a XAMPP server, but the MySQL Wokbench correction will be maintained so that the program is valid for both database servers.

Once the following task has been completed, both the code and a pdf document explaining the development of the practice will be sent to the task prepared for it in Moodle.

We have, for example,

**Car Class**



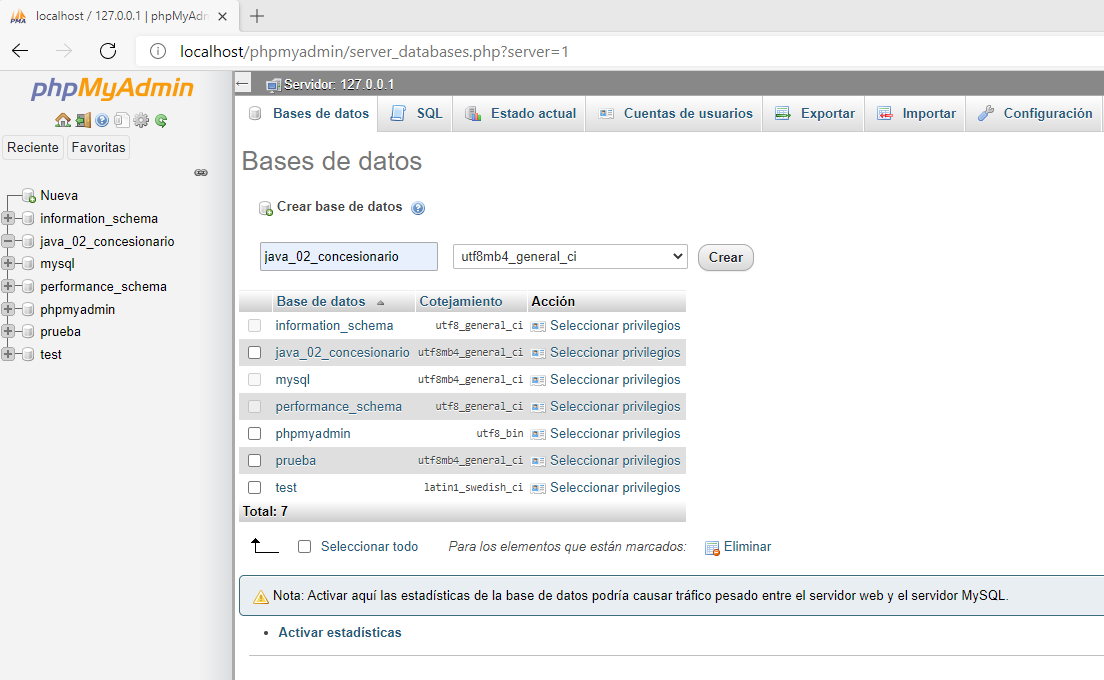


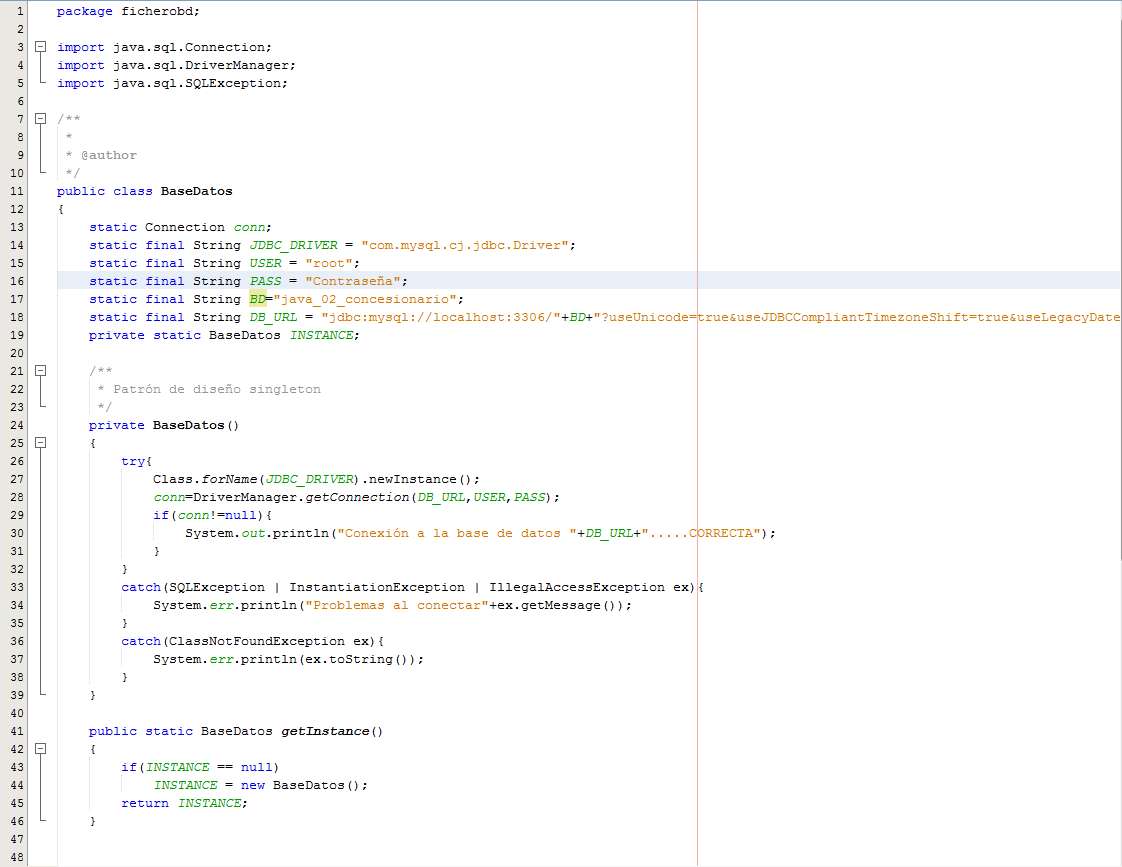
This is a class with six attributes, their corresponding setters and getters and a toString method that overrides the default toString method, showing us all the information about the objects of the class that are instantiated.

**DatabaseClass​**

It will be used to create a new database instance (it is understood that the root password is “” in the MySQL Workbench)

Logically, the database must be created on the database server, in this case XAMPP (In the screenshot it is already created, but it shows how it would be done)







The attributes of the BaseData class (All of them are static)

An attribute of type connection, conn

A final attribute of type String with the Driver that we are going to use in the program, which is included in the following connector:



A final attribute of type String in which the user with whom the database will be accessed will be stored

A final attribute of type String in which the password corresponding to the user with whom it will be linked to the database will be stored.

A final attribute of type String in which the name of the database to which we want to connect will be stored.

A final attribute of type String in which the url address to access the database will be stored. Additionally, if you want to access a database in MysQLWorkbench from Netbeans, you must use

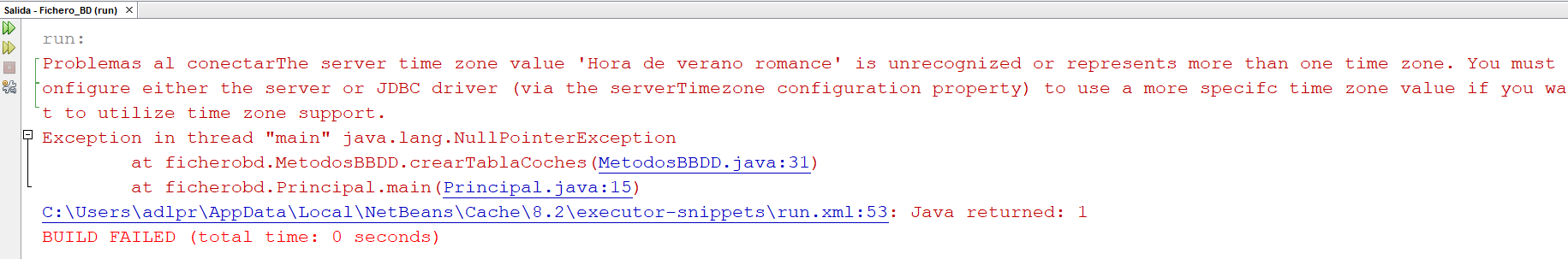
+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

After the database url.

**NOTE:** If the driver has not been configured correctly, using the line

static final String DB\_URL = "jdbc:mysql://localhost:3306/"+BD+"?useUnicode=true&useJDBCCompliantTimezoneShift=true&useLegacyDatetimeCode=false&serverTimezone=UTC";

The program shows us the following on the screen:



(Can also be fixed by modifying the my.ini file in MySQL Workbench)

(In XAMPP it is not necessary, in principle, but it is better to put it in case we are going to change the DBMS in the future)

The last attribute is a private object of type BaseData (It is only declared, not instantiated)

The BaseData() constructor method allows us to create a new instance of the driver and achieve a connection using the lines:

Class.forName(JDBC\_DRIVER).newInstance();

conn=DriverManager.getConnection(DB\_URL,USER,PASS);

where conn is the connection type attribute

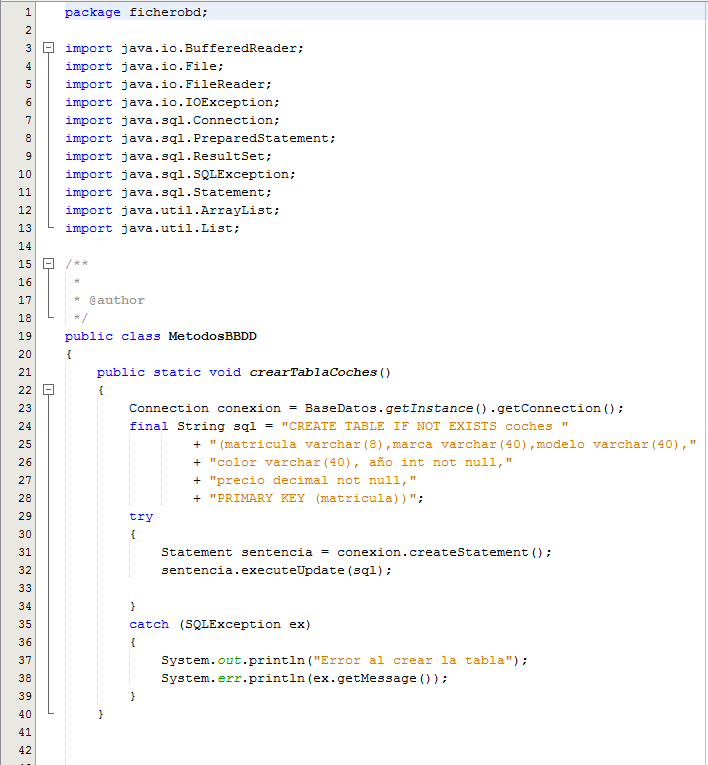
(Logically included within a try-catch to control possible exceptions)

The last two methods are:

getInstance, of type BaseData, public and static, which does is create an instance of BaseData if it does not already exist and matches the created instance to the attribute of type BaseData. Return that instance to be able to work with its methods

getConnection, of type Connection, public and static, which returns the connection created in the with attribute.

**BBDDMethods.java**



This class is composed entirely of static methods,

In the createCarTable() method, the line

Connection connection = Database.getInstance().getConnection();

It creates a BaseData instance for me and returns its connection to me, if everything goes well

Then a query is stored that allows the cars table if it does not exist and in the try, the query is executed using the commands:

Statement statement = connection.createStatement();

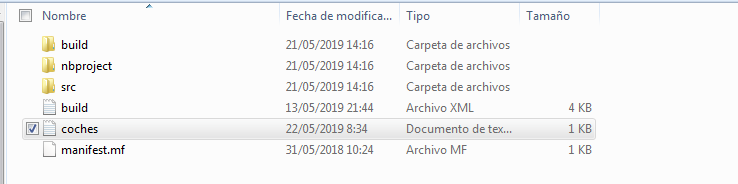
statement.executeUpdate(sql);

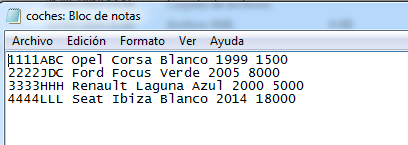
This creates the car table (As long as there are no errors such as those provided in the catch)



The loadCars() method requires that there be a cars.txt file to read from to create the Car objects and then to save them to the table by calling the insertCar() method. First, the document is read and separated using the Split by spaces command, each of the fields, then these fields are used to generate the Car object, which will be inserted using the insertCar method

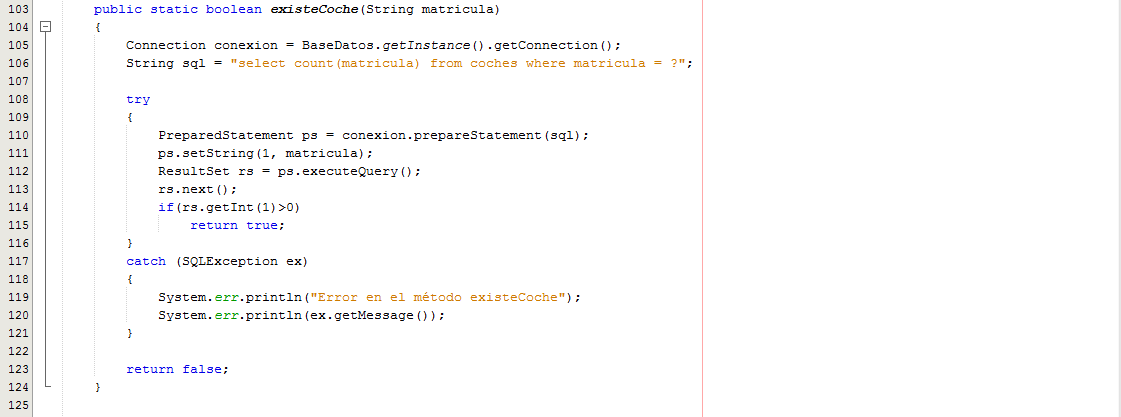
Status of the Coches.txt file:







The getcoches() method allows you to query all the information in the MySQL table, creating an instance of the Car class with each of the rows in the table, which will be added to a list of Car objects called cars, which will be which the method returns. It should also be noted that the line System.out.println(c) will write the information of the object



The existsCar() method informs us whether the car whose license plate we pass exists or not. A connection is created to the database and a string of characters is created in which the place that would correspond to the license plate, we put a question closure (question mark). Then, using the following three lines, we prepare the query and execute it:

PreparedStatementps = connection.prepareStatement(sql);

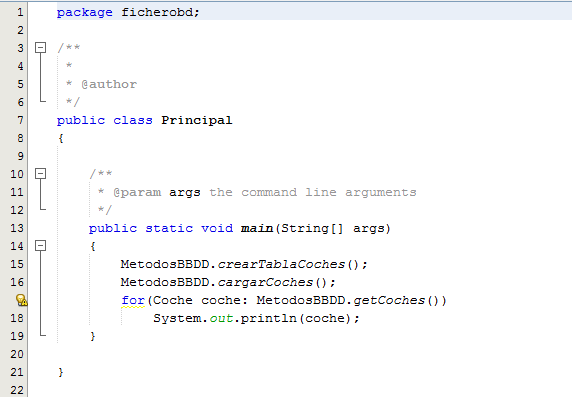
ps.setString(1, enrollment);

ResultSetrs = ps.executeQuery();



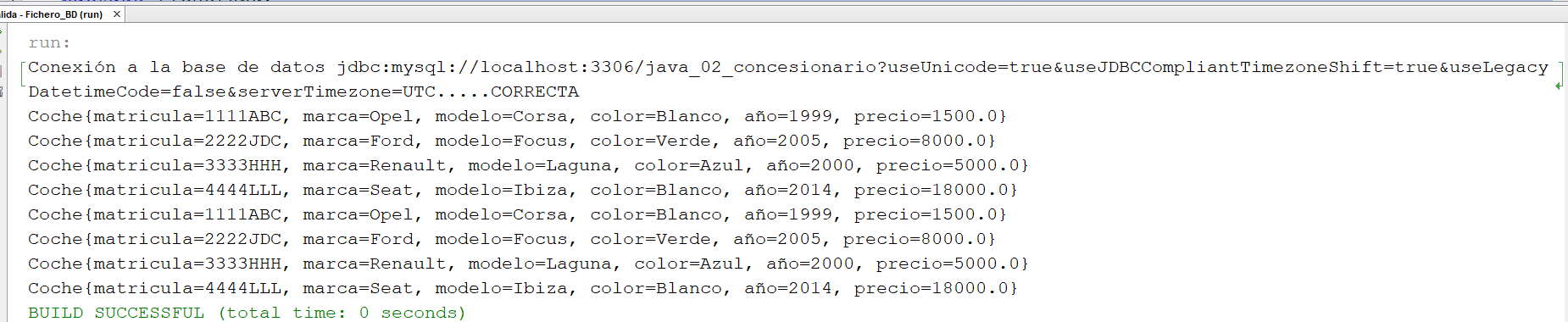
This method allows us to insert a Car into the table, if the car does not exist in the table (comparing it using c.getMatricula()) a connection and a character string that stores a query are created, then as in the previous method, we prepare and we execute.

**The main class**

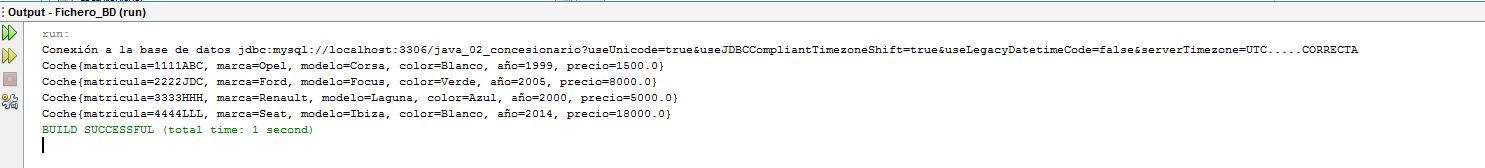


The table is created, the cars are loaded and the list provided by the getCars() method is read, extracting their information through the console.

Running this program will produce the following output:



If we remove the line from getCars() in which the cars are written, we have:



We can also examine how the database is on the server.

