

HW4. (Basic DB Programming) Databases-060

부산대학교 전기컴퓨터공학부-정보컴퓨터공학전공

201624632

Marzhan Alenova

제출일: 2019-11-15

1. 숙제 구현 내용(50점)

- Describe the important data structures used to implement the program.

First of all, the program, that is going to be described in this report, is written in Java language. More detailed explanation about the organization of data and interfaces that took a place in the implementation of program will be given. Despite the regular storage formats of data, I would like to focus mainly on JDBC. JDBC stands for Java DataBase Connectivity and it is a Java API for executing SQL database queries. Since Java is an object-oriented language, the API means a set of classes and interfaces. These classes and interfaces are described in a special package **java.sql**.

Moving to the description of the basic interfaces and the driver interface, in order to make it clearer to understand, I will start from the most important ones and how they are used in the given program implementation.

java.sql.DriverManager: provides loading drivers and creating new database connection; this is the core JDBC interface that determines the correct choice and initialization of the driver for a given DBMS in these conditions;

```
Connection connection = DriverManager.getConnection(
    "jdbc:oracle:thin:@Qwerty:1521:XE", "sys", "password");
```

Figure 1. java.sql.DriverManager

As you can see, it is really important to give the database specifications; otherwise connection cannot be successfully done. In order to be sure about the database configurations, I checked the **tnsnames.ora** in the **oraclexe** folder where the **ADMIN** folder is located. In my case, it was the following location:

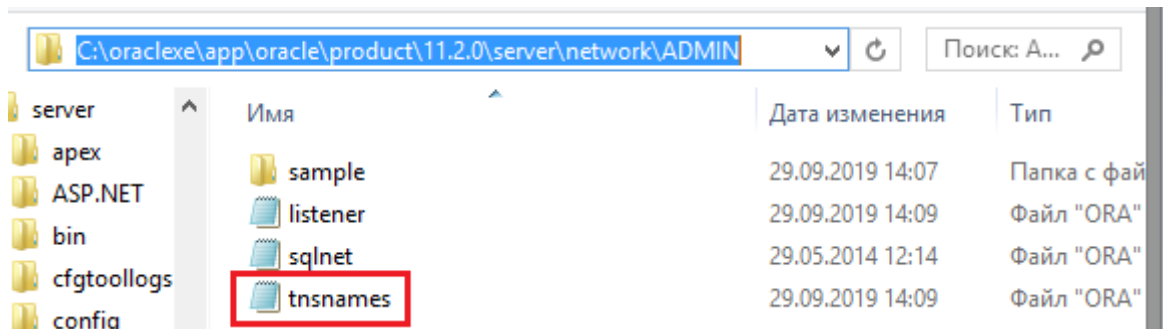


Figure 2. tnsnames.ora file location

Here, you will have the information about your database's host name, port number and service name/SID. As previously stated, it is extremely important to identify the parameter for this DriverManager class in a right way. In, my tnsnames.ora file I get the following information, so I was able to figure out the database configuration.

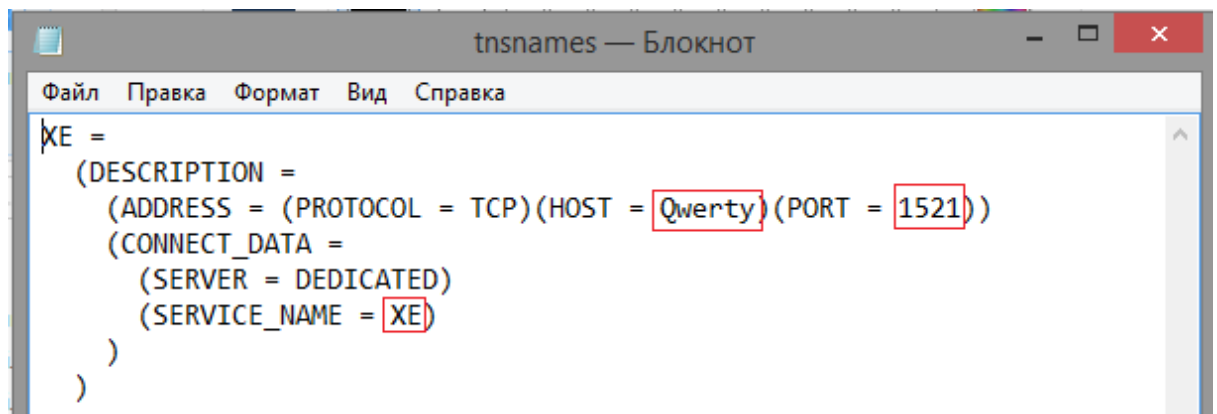
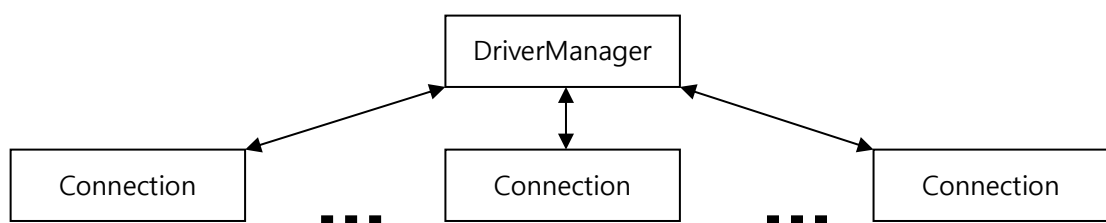


Figure 3. tnsnames.ora file

What about the username and password, it should be the username and password from which SQL command prompt can be accessed. Also, I granted the required permission to the user 'sys' and give the administrator privilege.

The next is the **java.sql.Connection**, it defines the characteristics and status of the database connection; in addition, it provides tools to control transactions and their isolation level; the relationship of DriverManager and Connection, can be illustrated in this way.



java.sql.Statement performs the functions of a container with respect to an SQL statement; at the same time, an expression means not only the query text itself, but also such characteristics as parameters and state of the expression; When transferring the Statement object to the database using the established connection, the DBMS will run the specified SQL command and return the result of its execution as a ResultSet(described after java.sql.Statement) object. It is better to methods of the statement object to be contained within the try {} catch construct. Because, it makes easier to catch the exceptions. For example, java.lang.NumberFormatException.

```
Statement stmt1 = connection.createStatement();  
ResultSet resultSet1 = stmt1.executeQuery("SELECT Fname, Minit, Lname, Ssn,
```

Figure 4. java.sql.Statement

java.sql.ResultSet provides access to the row set obtained by executing this SQL statement. In other words, to retrieve the query results as a ResultSet object, we should create an instance of ResultSet and call the function executeQuery() using the instance of java.sql.Statement instance.

In addition, java.sql.Statement expression interface acts as the ancestor for the other two important interfaces: java.sql.PreparedStatement and java.sql.CallableStatement, the first of which is used to execute precompiled SQL-expressions, the second - to execute stored procedure calls. I used both of the interfaces in the program implementation, the first one in the while performing the first operation and the second one for remaining two.

```
String sql1 = "INSERT INTO EMP(Fname, Minit, Lname, Ssn, Bdate,  
PreparedStatement pstmt1 = connection.prepareStatement(sql1);
```

Figure 5. java.sql.PreparedStatement

Since, given tables contain the date type, I used the TO_DATE() function in order to convert the string into date type and insert it into the table. So, program has the separate caution related to the form of the date type as an input.

```
TO_DATE(?, 'mm/dd/yyyy')
```

Figure 6. Date type form

- Description of important functions written to implement the program. Describe key functions in pseudo code form when describing critical functions

Let's start from the function which is responsible for the connection part: It has another interpretation in the form of following 4 steps.

1. Import the JDBC package into our Java code;
2. Register JDBC driver;
3. Send information for connecting to the database (URL, username and password);
4. Create a connection using the getConnection() method;

We will consider each step separately to make clearer. The first one is the importing a JDBC Package. To import the JDBC package, we use the standard import keyword, which tells the compiler to include the necessary classes. To connect JDBC, we need to connect the **java.sql package**. *

In code, it looks like this:

```
1 package hw4_db;
2 import java.sql.*;
```

Figure 7. java.sql package. *

The next step is registration of JDBC driver. After importing JDBC, we must register our driver before using it. This is the process by which a driver class file is loaded into memory. After that, it can be used as an implementation of the JDBC interface. Actually, there are two ways how we can do that, but I used including Class.forName(). By the way, this method is the most common. We use the Class.forName() method to dynamically load the driver class into memory, after which it is automatically registered.

```
try{
    int operation;
    Scanner input = new Scanner(System.in);

    Class.forName("oracle.jdbc.driver.OracleDriver");
```

Figure 8. Class.forName()

Database Information Transfer: after we registered our driver, we can establish a connection using the DriverManager.getConnection() method. It has the following option: getConnection (String url, String user, String password) As we said before, it requires the URL of our database. Those we need to transfer information about our database. I

already told about that in the previous section.

Creating the connection: after we have transmitted all the necessary information, we can create a physical connection to the database.

The next important function is the insertion new values into one of the table, in other words updating. It has another interpretation in the form of following steps:

1. Take all appropriate inputs(table name, values) from the user;
2. Create an object of Statement class and SQL query statement;
3. Execute query by using executeUpdate() method via newly created Statement object;

First of all, we take all of the inputs necessary for execution of SQL query. Below you can see the input list for the table EMP.

```
int num_table;
System.out.println("Enter the Table number:\n");
System.out.println("1. EMP\n");
System.out.println("2. DEPT\n");
num_table = input.nextInt();

if(num_table == 1) {
    String Fname, Minit, Lname, Bdate, Sex, Address;
    int Ssn, Salary, Super_ssn, Dno;
```

Figure 9. Input list for EMP table

Then, we have create an SQL query statement in order to insert a new records.

```
String sql1 = "INSERT INTO EMP(Fname, Minit, Lname, Ssn, Bdate, "
+ "Address, Sex, Salary, Super_ssn, Dno) "
+ "VALUES (?, ?, ?, ?, TO_DATE(?, 'mm/dd/yyyy'), ?, ?, ?, ?)";
```

Figure 10. SQL query

```
PreparedStatement pstmt1 = connection.prepareStatement(sql1);
```

Figure 11. Object of PreparedStatement

Eventually, we can just set all of the values and in the end, call the function executeUpdate().

```

pstmt1.setString(1, Fname);
pstmt1.setString(2, Minit);
pstmt1.setString(3, Lname);
pstmt1.setInt(4, Ssn);
pstmt1.setString(5, Bdate);
pstmt1.setString(6, Address);
pstmt1.setString(7, Sex);
pstmt1.setInt(8, Salary);
pstmt1.setInt(9, Super_ssn);
pstmt1.setInt(10, Dno);
pstmt1.executeUpdate();

```

Figure 12. executeUpdate() function

Moving to the function, here, we have to display all of the records from the table after getting its name as an input from the user. It has an interpretation in the form of following steps:

1. Take a table name as an input;
2. Create a statement and then execute query with the SELECT statement;
3. Display the result by the getString()(getInt()) method;

Firstly, we need a user to enter the table name, since it is one of the conditions. It is the same as it was for the previous function. But, now it is for the DEPT table.

```

int num_table;

System.out.println("Enter the table number:\n");
System.out.println("1. EMP\n");
System.out.println("2. DEPT\n");

num_table = input.nextInt();

```

Figure 13. Getting a table name as input

Now, we create an object of Statement class using createStatement() method and define the object of ResultSet class(it was described in the previous section) too.

```

Statement stmt2 = connection.createStatement();
ResultSet resultSet2 = stmt2.executeQuery("SELECT Dname,Dnumber,"
+ "Mgr_ssn,Mgr_ssn_date FROM DEPT");

```

Figure 14. executeQuery() method using ResultClass object

The records are simply displayed using an appropriate getString(), getInt() methods and below you can see this part for the DEPT table.

```

while(resultSet2.next()) {
    numofRows = numofRows + 1;
    System.out.println(resultSet2.getString("Dname"));
    System.out.println(resultSet2.getInt("Dnumber"));
    System.out.println(resultSet2.getInt("Mgr_ssn"));
    System.out.println(resultSet2.getString("Mgr_ssn_date"));
}

```

Figure 15. Displaying the records

Only the remaining one is the SQL JOIN Query execution. In general, it not so different from the function that was described before, but here user by himself enters the Query statement. In consists from the following steps:

1. Take an SQL JOIN Query from the user;
2. Create statement and execute query;
3. Display the result by the getSting()(getInt()) method;

Firstly, we take an input from the user by using the netLine() method.

```

System.out.println("Please, enter the SQL JOIN query that you want to execute. "
    + "All in one line!\n");
String sql = input.nextLine();

```

Figure 16. Get SQL Query as an input

Now, we create an object of Statement class using createStatement() method and define the object of ResultSet class(it was described in the previous section) too. Here, a newly received SQL Query is executed.

```

Statement stmt = connection.createStatement();
ResultSet resultset = stmt.executeQuery(sql);

```

Figure 17. SQL Query execution

Displaying of the result is exactly the same as it was for the previous function.

Also, we have used a function that is responsible for the closing of the connection. It is important not to forget to do that from the secure considerations.

```

connection.close();
System.out.println("Connection is closed!");

```

Figure 18. Closing the connection

2. 프로그램 데모 소개 (40점)

- Describe program implementation environment

In order to implement this program, we used the Java object-oriented language with JDBC. In the first section of this report, we briefly mentioned about that, but now let me to concentrate on it in a more detailed way.

In order to better understand the essence of the approach used in JDBC, let me remind what Java interfaces are. Unlike classes that contain both a declaration and an implementation of methods, interfaces provide a higher level of abstraction, describing only method declarations. Given the possibility of inheritance, and multiple inheritances, this approach allows the user to create programs designed to work with databases that are independent of the specific implementation of the DBMS itself or its access methods. In order to access a specific DBMS, the developer needs a JDBC driver. In the concept of universalizing data access through standard interfaces, the JDBC driver is a collection of classes that implement JDBC interfaces. Implementation of an interface in Java means the creation of a class that refers to an interface in its declaration and offers a concrete implementation of interface methods already in the form of methods of this class.

Now, it is clear that I downloaded an appropriate JDBC driver for the Oracle database and add it to project library. In my case, it was ojdbc6.jar. The detailed explanation about the setting it up will be given in the section with manual.

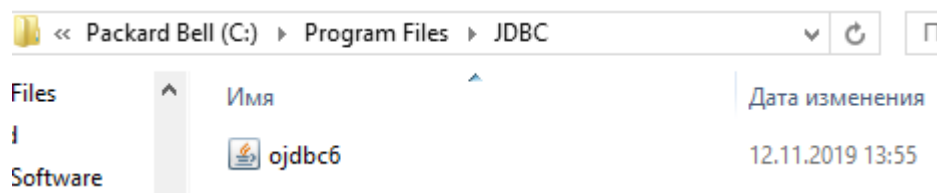
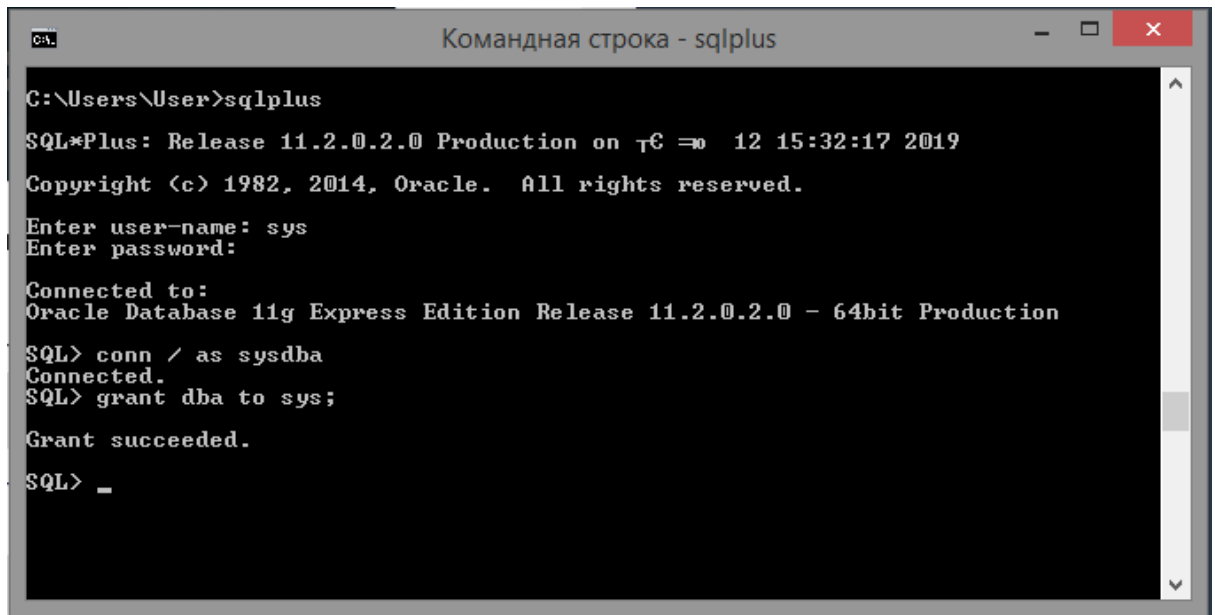


Figure 19. JDBC driver

Also before the setting the connection, user should enter his/her own account and grant the administration privilege to it. In other words, he/she should prefer the startup of the database. I did it via cmd.



```
C:\Users\User>sqlplus
SQL*Plus: Release 11.2.0.2.0 Production on ТЭ 12 15:32:17 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.
Enter user-name: sys
Enter password:
Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
SQL> conn / as sysdba
Connected.
SQL> grant dba to sys;
Grant succeeded.
SQL> _
```

Figure 20. Startup the database

But installing all of the drivers and libraries is not enough to build the connection. In my case, I had a quite big problem related to the NLS_LANG error and could not connect to the database for 2 days. The problem was that environment language in my laptop is Russian and Database supported the English. Oracle's JDBC drivers support NLS (National Language Support). NLS lets the user retrieve data or insert data into a database in any character set that Oracle supports. If the clients and the server use different character sets, then the driver provides the support to perform the conversions between the database character set and the client character set. I have two different languages for the server and client, but fixing it took more time than I expected. Simple changing the language of environment was not a solution, since Russian and English character set codes cannot be directly converted. So, I have a solution through the regedt32. Here, we search for the NLS_LANG and changes the value from Russian to English.



Figure 21. regedt32

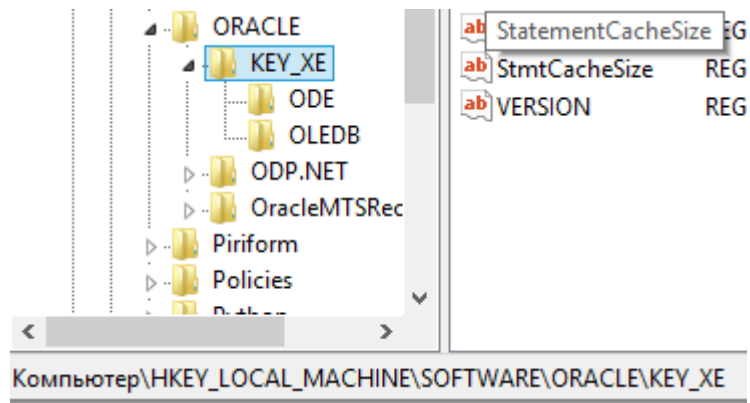


Figure 22. Changing the value of NLS_LANG

After setting all necessary files and libraries, user can test the connection. It can be done while preferring the database connection in a separate way.

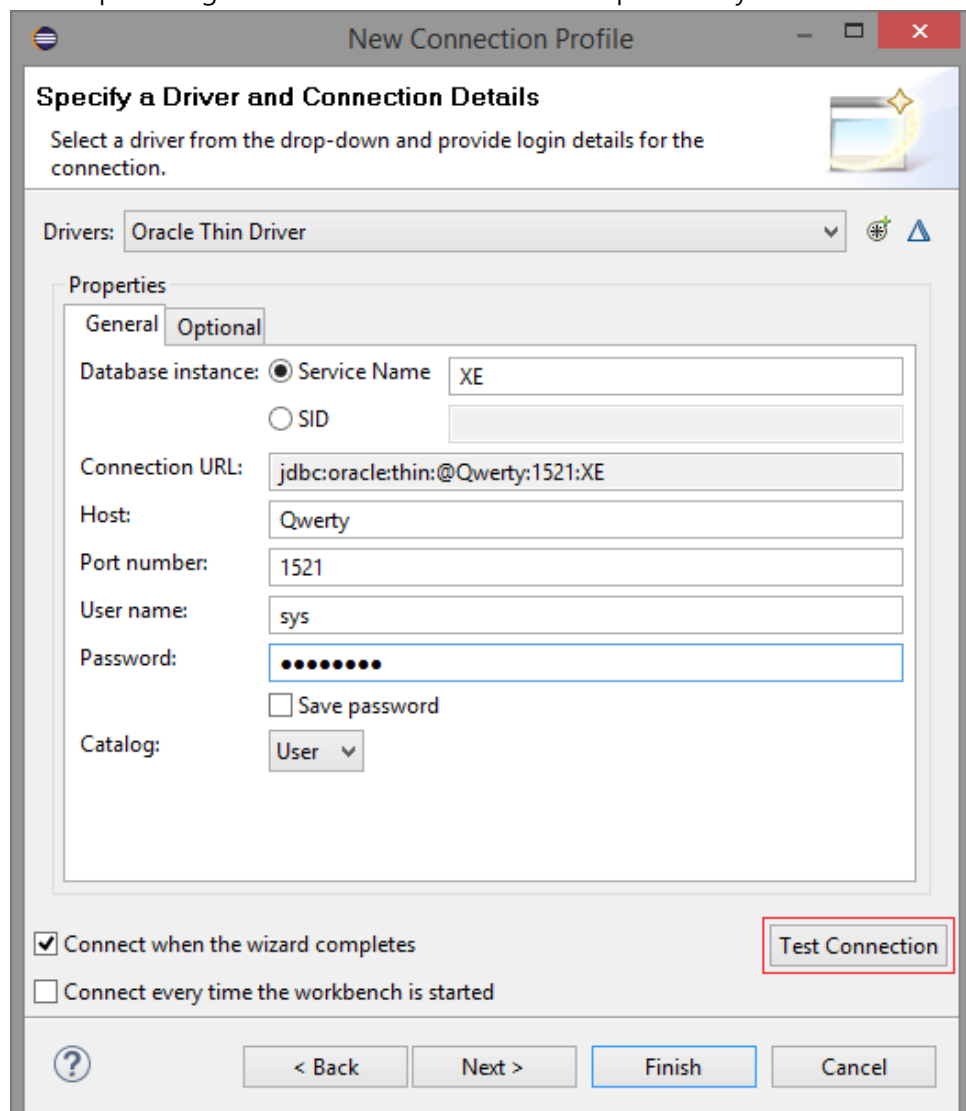


Figure 23. Testing the connection with database

- Describe how to run the program

Since, we have two different section related to the description of process how to run the program, in this section I will focus mainly in the output that we got from the program and does it correspond to the condition of the task.

In order to run the first user should have all of the drives and necessary files, what were listed above, to be set. We will be back to this step in the next section. In case, if everything is okay, we can start the implemented program. When you first run it, you should get the first line to be the result of connection to the database. In my opinion, it is very important to notify the user about the connection status before performing other operations.

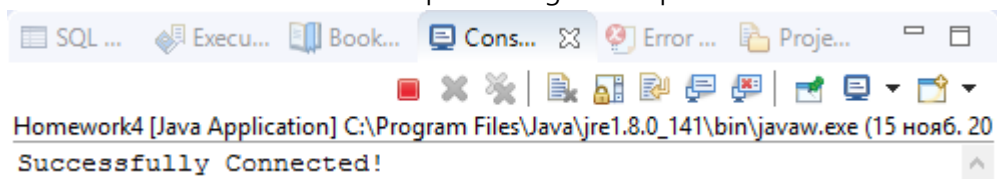


Figure 24. Connection status

At the same time number of operations will be displayed. There are four of them, and you can choose the desired one just by entering the number in the beginning of the line.

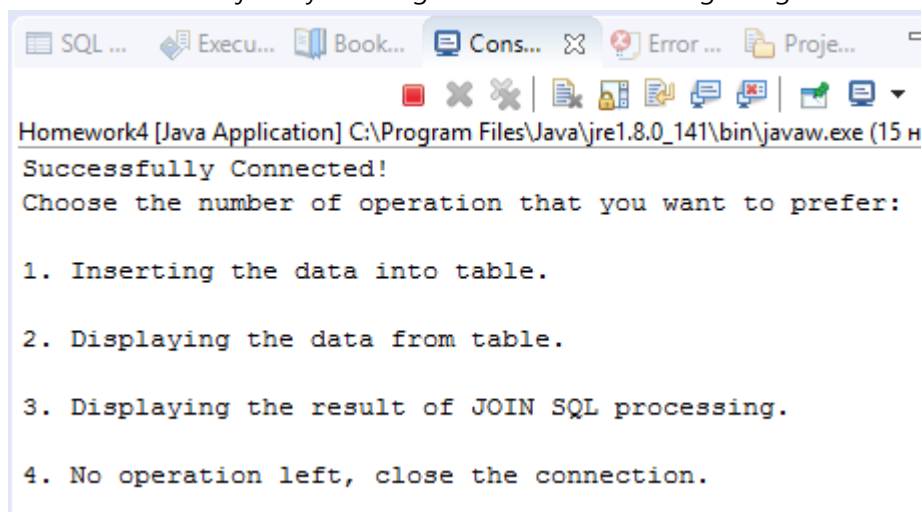
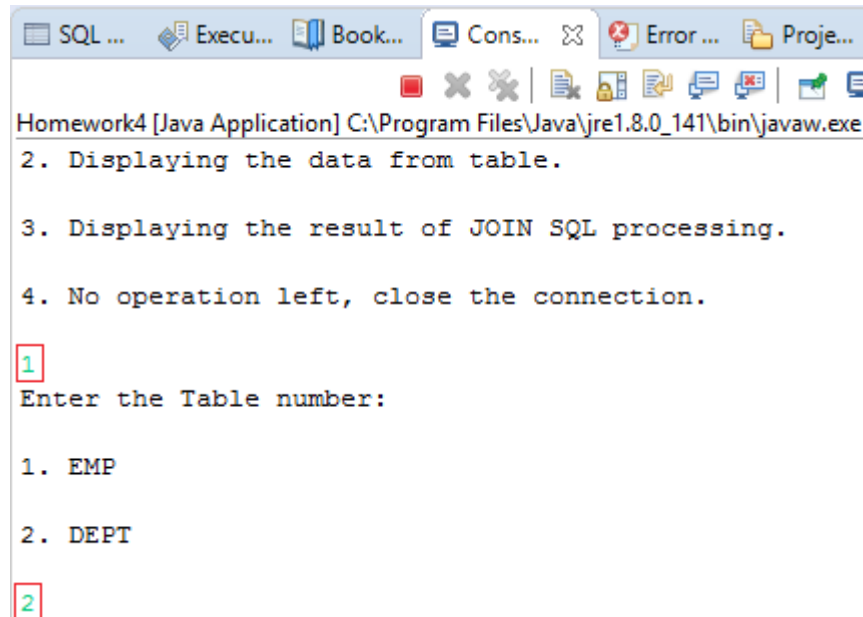


Figure 25. List of operations

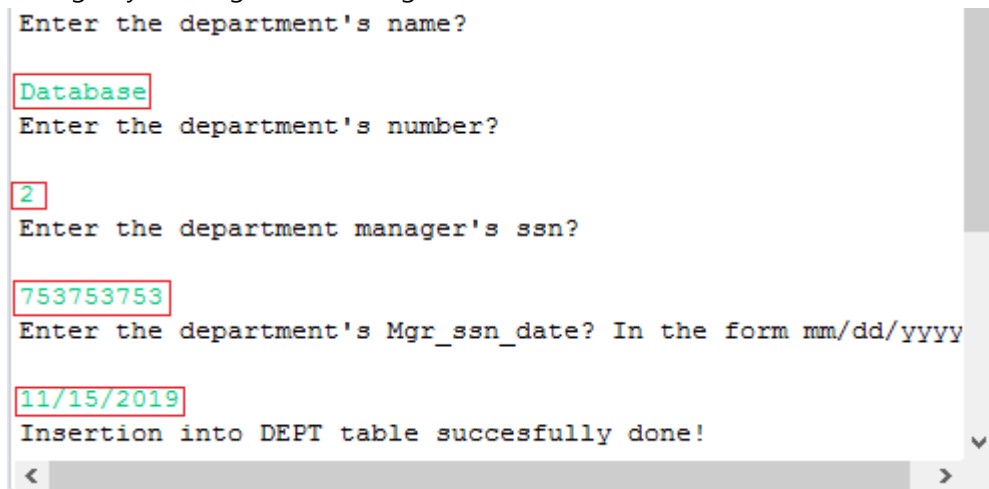
Let's perform each operation one by one and see the output that we will get. When I type the number 1 which is the inserting the new record, then we have another prompt from the program. It is the choosing number of table. User should choose into which table he/she wants to insert the values. I typed the number 2 which is the DEPT table.



```
SQL ... Execu... Book... Cons... Error ... Proje...
Homework4 [Java Application] C:\Program Files\Java\jre1.8.0_141\bin\javaw.exe
2. Displaying the data from table.
3. Displaying the result of JOIN SQL processing.
4. No operation left, close the connection.
1
Enter the Table number:
1. EMP
2. DEPT
2
```

Figure 26. The first operation: inserting a new record-1

After that, one by one different questions will be displayed. You should enter the appropriate input for them and they are related to the column attributed of the table. Please, read the questions carefully, otherwise exception/error can occur! If everything is entered right, you will get the message that insertion is done.



```
Enter the department's name?
Database
Enter the department's number?
2
Enter the department manager's ssn?
753753753
Enter the department's Mgr_ssn_date? In the form mm/dd/yyyy
11/15/2019
Insertion into DEPT table succesfully done!
```

Figure 27. The first operation: inserting a new record-2

Let's check if the insertion really took a place via cmd. As you can see a new record was inserted successfully.

```
SQL> SELECT Dname,Dnumber FROM DEPT WHERE Dnumber = 2;
DNAME          DNUMBER
-----
Database        2
SQL> _
```

Figure 28. The first operation: inserting a new record-3

Let's move to the next operation which the displaying the table's records. After finishing the first operation we automatically will be requested to choose the next operation's number. Like with the first operation, we should choose the table number. I typed the 1 which is the table EMP.

```
1. Inserting the data into table.
2. Displaying the data from table.
3. Displaying the result of JOIN SQL processing.
4. No operation left, close the connection.
2
Enter the table number:
1. EMP
2. DEPT
1
```

Figure 29. The second operation: displaying all of the records -1

Then one by one each row will be displayed. At the end of each row, there will be message stating that displaying of whole is done. It will help the user to not be confused and focus on the date. Also, at the end of the total number of rows and message about the operation status will be displayed.

```
John
B
Smith
123456789
1965-01-09 00:00:00.0
731 Fondren, Houston, TX
M
30000
333445555
5

1 row data displayed.

Franklin
T
Wong
333445555
1955-12-08 00:00:00.0
638 Voss, Houston, TX
M
```

Figure 30. The second operation: displaying all of the records – 2

```
James
E
Borg
888665555
1937-11-10 00:00:00.0
450 Stone, Houston, TX
M
55000
0
1

1 row data displayed.

Displaying the EMP Table data is done!
In total, 8 rows displayed.
```

Figure 31. The second operation: displaying all of the records - 3

Let's move to the next operation which the processing SQL JOIN Query. Here, the process almost the same, just enter number 3 but you are not requested to enter the table number, instead there is a message stating that user should enter the SQL Join Query Statement.



The screenshot shows an IDE console window with tabs for SQL Results, Execution Plan, Bookmarks, Console, Error Log, and Project Explorer. The console output is as follows:

```
<terminated> Homework4 [Java Application] C:\Program Files\Java\jre1.8.0_141\bin\javaw.exe (15 нояб. 2019 г., 5:15:41)
Choose the number of operation that you want to prefer:

1. Inserting the data into table.
2. Displaying the data from table.
3. Displaying the result of JOIN SQL processing.

3
Please, enter the SQL JOIN query that you want to execute. All in one line!

SELECT Fname FROM EMP, DEPT WHERE EMP.Ssn = DEPT.Mgr_ssn
```

Figure 32. The third operation: SQL JOIN Query execution - 1

I entered my SQL Join Query. Please, read the message carefully in order to avoid the wrong execution of the query. And we got the following result. Let's compare it with the cmd compilation.

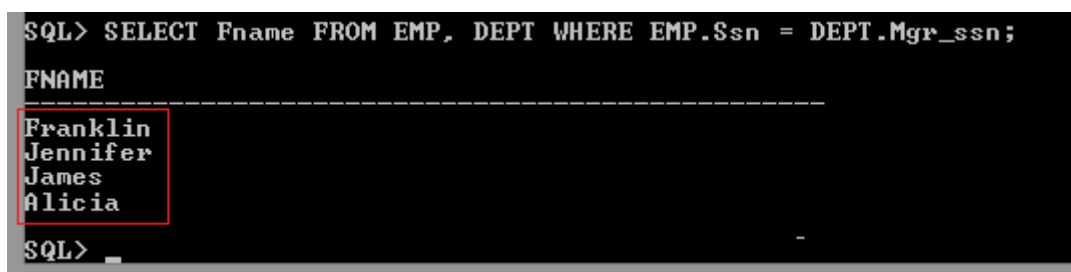
```
SELECT Fname FROM EMP, DEPT WHERE EMP.Ssn = DEPT.Mgr_ssn
Fname:
Franklin

Fname:
Jennifer

Fname:
James

Fname:
Alicia
```

Figure 33. The third operation: SQL JOIN Query execution - 2



The screenshot shows a command prompt window with the following output:

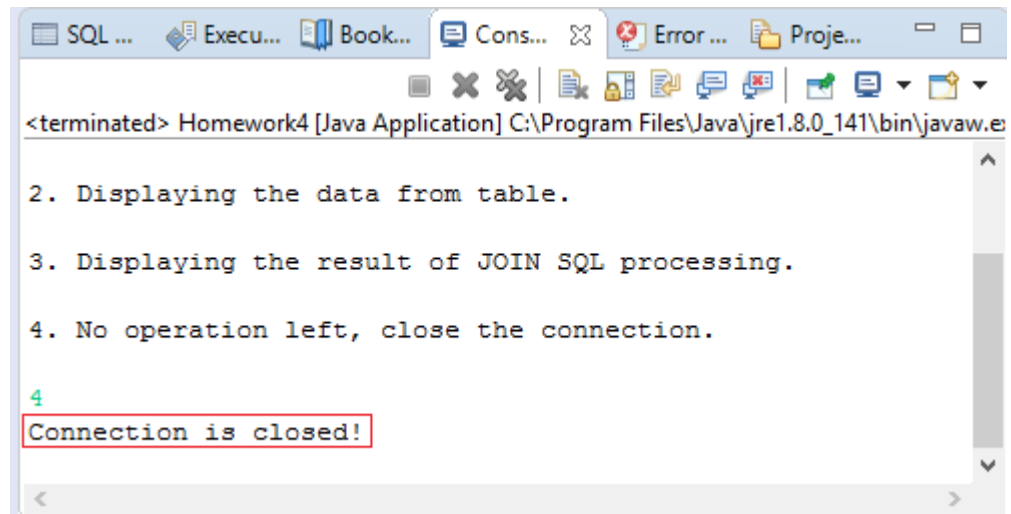
```
SQL> SELECT Fname FROM EMP, DEPT WHERE EMP.Ssn = DEPT.Mgr_ssn;
FNAME
-----
Franklin
Jennifer
James
Alicia
SQL>
```

The names 'Franklin', 'Jennifer', 'James', and 'Alicia' are listed under the header 'FNAME' and are enclosed in a red rectangular box.

Figure 34. The third operation: SQL JOIN Query execution - 3

Let's move to the last operation which is the closing the connection, When the user is done with all operations, he/she just enter the number 4. And he will get the message that

connection is closed.



```
<terminated> Homework4 [Java Application] C:\Program Files\Java\jre1.8.0_141\bin\javaw.e  
2. Displaying the data from table.  
3. Displaying the result of JOIN SQL processing.  
4. No operation left, close the connection.  
4  
Connection is closed!
```

Figure 35. The fourth operation: closing the connection

- How to run the implemented program in manual format

Introduction to the program. The following program is the program written in Java object-oriented language using JDBC. This program perform connects to the database according to its url and performs 4 different operations with the 2 different tables. Below, the detailed description and instruction for running this program are provided.

Setting the development environment. Program requires 3 prerequisites: **JDK**, **Oracle Database** and **JDBC driver** for the Oracle database. In case, if one of the programs is not installed/downloaded, choose the appropriate version, download the execution file and install it. Below, links for download are provided. (Remember the location of JDBC driver, since the Eclipse IDE will prompt the user to add the address of the folder.)

-Oracle Database: <https://www.oracle.com/database/technologies/xe-downloads.html>

-JDBC Driver: <https://www.oracle.com/database/technologies/jdbcdriver-ucp-downloads.html>

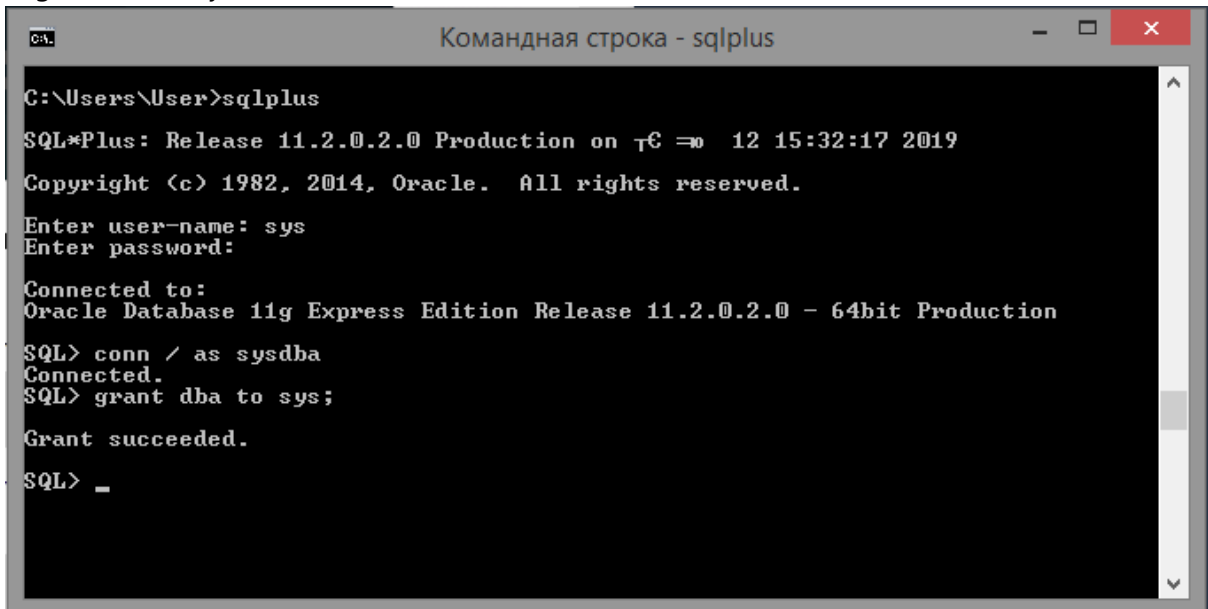
After having all of the prerequisites installed startup the Oracle database. This manual shows the way of doing it via cmd.

Enter the following command into the command line one by one:

->sqlplus (cmd prompts to enter the valid username and password)

->conn / as sysdba

->grant dba to sys;



```
C:\Users\User>sqlplus
SQL*Plus: Release 11.2.0.2.0 Production on 12 15:32:17 2019
Copyright (c) 1982, 2014, Oracle. All rights reserved.
Enter user-name: sys
Enter password:
Connected to:
Oracle Database 11g Express Edition Release 11.2.0.2.0 - 64bit Production
SQL> conn / as sysdba
Connected.
SQL> grant dba to sys;
Grant succeeded.
SQL> _
```

Figure 36. Startup the Oracle database

->Create two table named EMP and DEPT with appropriate column attributes.

After connecting to the Oracle database, move to the Eclipse IDE. In case, if 'Database debug' perspective is not installed to the IDE, install it via Navigation tab in the menu bar. This manual shows one of the possible ways how user can do that.

->Go to the Navigation-> Perspective -> Open Perspective -> Other

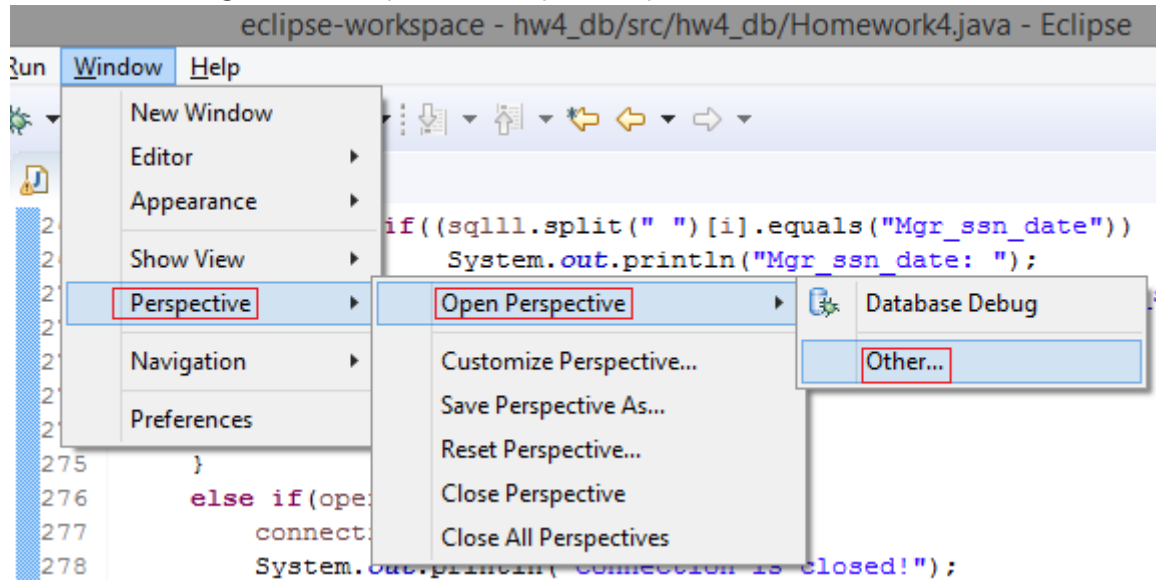


Figure 37. Installing the Database Debug

->New window 'Open perspective' will show up. Choose the 'Database Debug' from the list and click 'OK'.

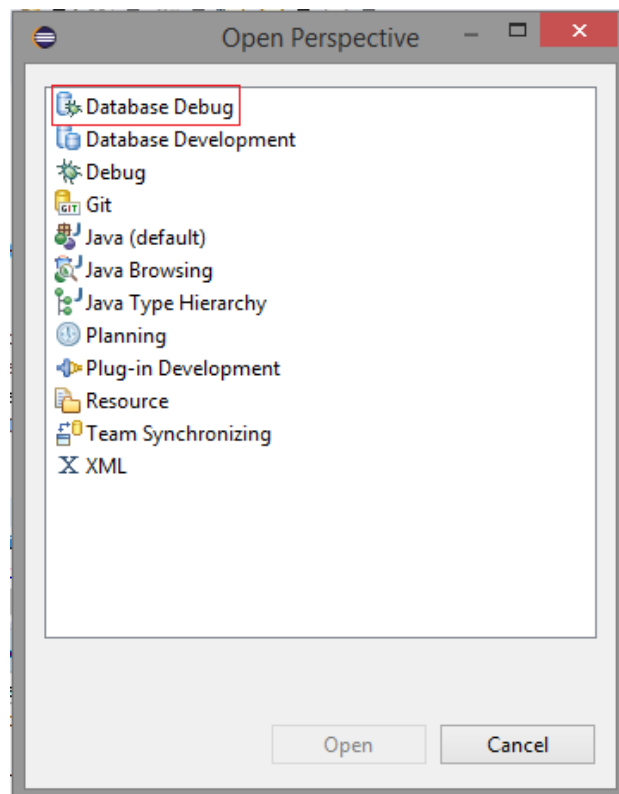


Figure 38. Open Perspective window

->After installation 'Data Source Explorer' tab will appear. Now go to the 'Database Connection', right click on it and choose 'New'.

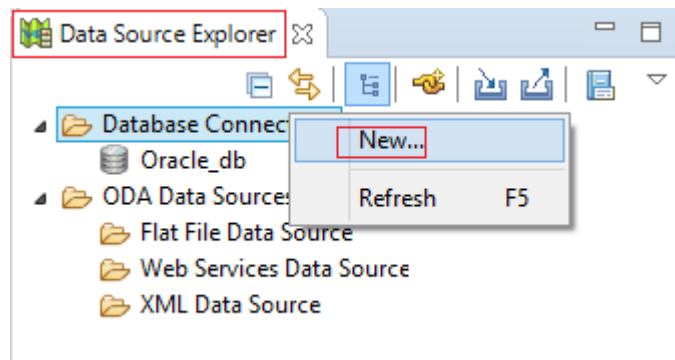


Figure 39. Database Connection

->'New connection profile' window will show up. Type the 'Oracle' into 'Connection Profile Types' and give the valid name. Click 'Next'.

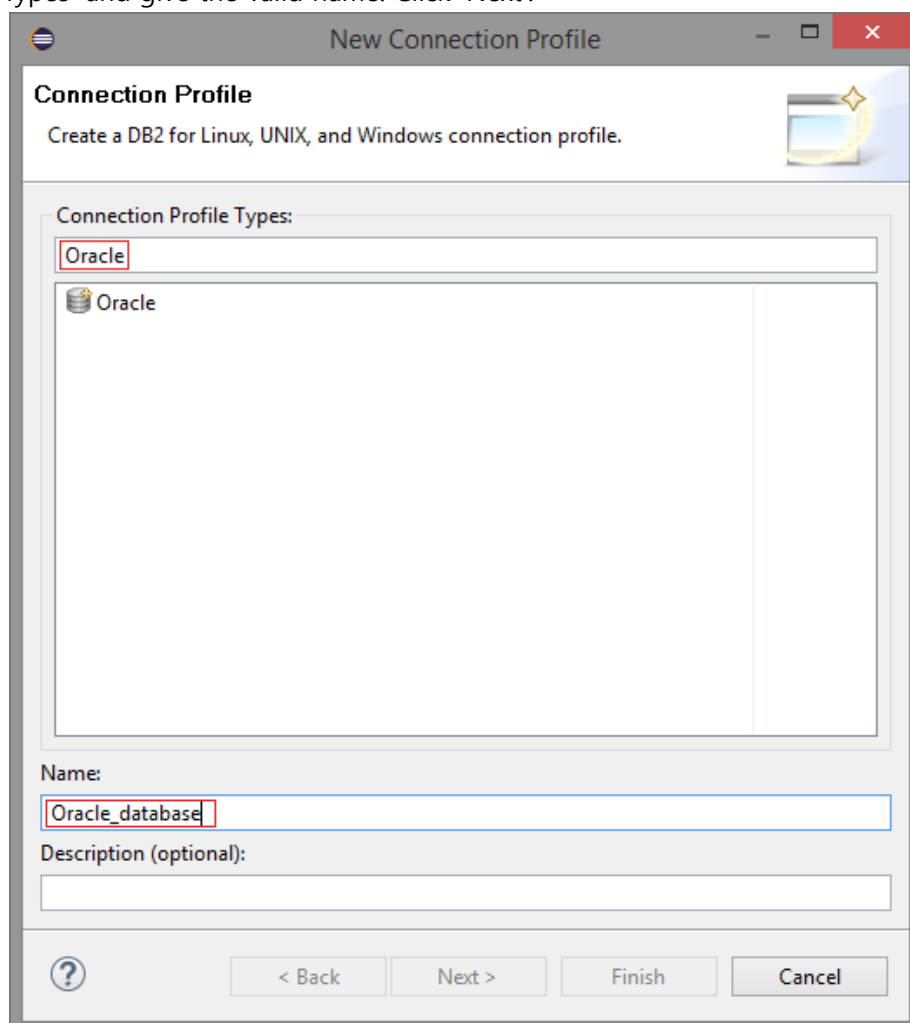


Figure 40. New Connection-1

->Fill out the blanks according to the user's database configuration. Information about database configuration can be found in the tnsnames.ora in ADMIN folder(oraclexe folder location). Click 'Test Connection' and after click 'Finish'.

New Connection Profile

Specify a Driver and Connection Details
Select a driver from the drop-down and provide login details for the connection.

Drivers: Oracle Thin Driver

Properties

General Optional

Database instance: ☒ Service Name XE
☐ SID

Connection URL: jdbc:oracle:thin:@Qwerty:1521:XE

Host: Qwerty

Port number: 1521

User name: sys

Password: ●●●●●●●●
☐ Save password

Catalog: User

☒ Connect when the wizard completes
☐ Connect every time the workbench is started

Test Connection

? < Back Next > Finish Cancel

Figure 41. New Connection-2

->Environment setting is done.

Running the program.

->Place the source code of the program and be sure that all libraries is included.

->Run the program.



Figure 42. Run the program

->Check connection status.

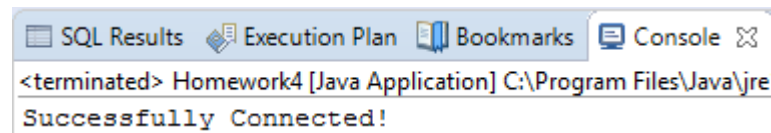


Figure 43. Connection status

->Carefully read the displayed message.

Choose the number of operation that you want to prefer:

1. Inserting the data into table.
2. Displaying the data from table.
3. Displaying the result of JOIN SQL processing.
4. No operation left, close the connection.

Figure 44. Operation list

->Choose the desired operation's number. ONLY NUMBER! (ex. 2)

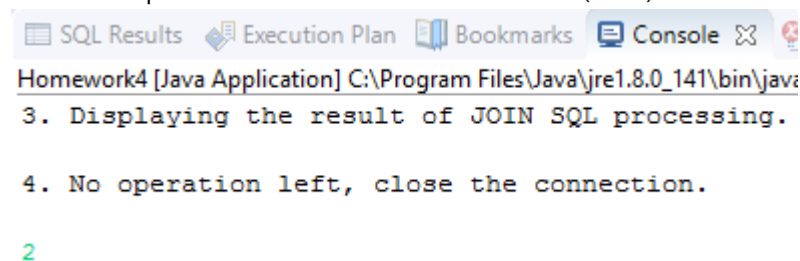
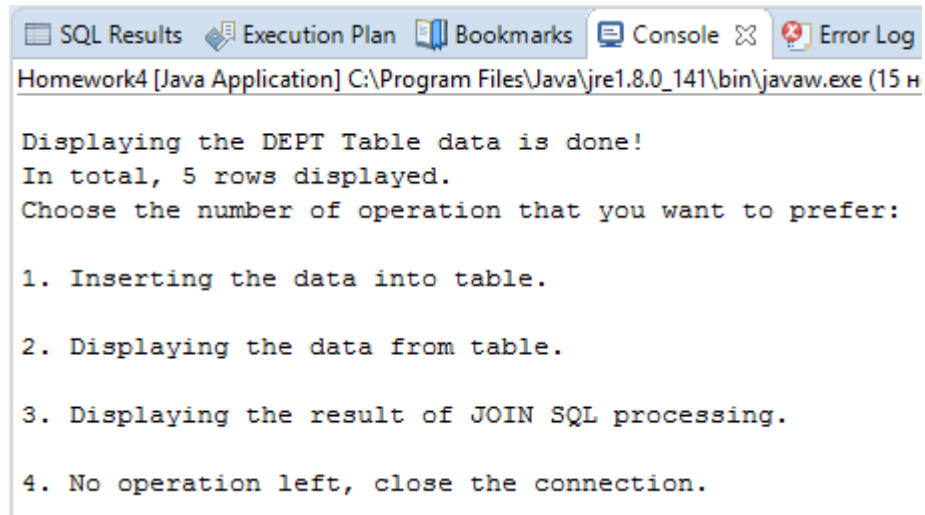


Figure 45. 2nd operation is chosen

->Carefully read the further messages that are displayed and give a valid input. (ex. 2nd operation, 2nd table is chosen)

->After compilation of the operation, can choose another operation. (All of the remaining operations are executed in the same way).

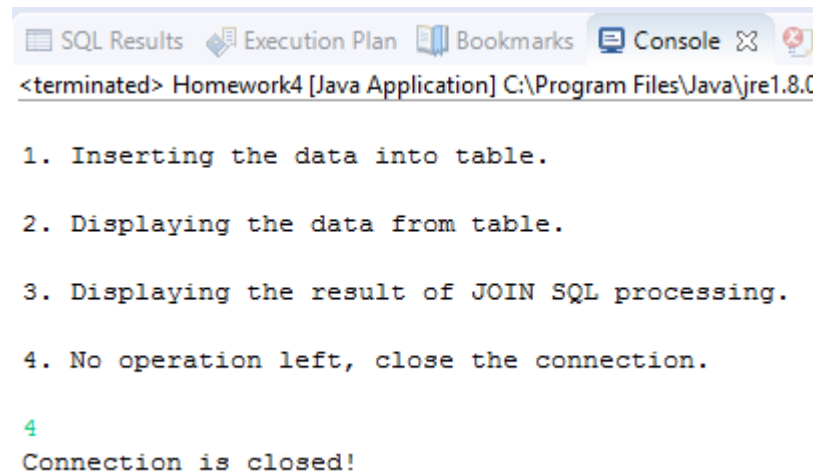


The screenshot shows a Java application window titled "Homework4 [Java Application] C:\Program Files\Java\jre1.8.0_141\bin\javaw.exe (15 H". The console displays the following text:

```
Displaying the DEPT Table data is done!  
In total, 5 rows displayed.  
Choose the number of operation that you want to prefer:  
  
1. Inserting the data into table.  
  
2. Displaying the data from table.  
  
3. Displaying the result of JOIN SQL processing.  
  
4. No operation left, close the connection.
```

Figure 46. Compilation of operation

->Once all desired operations are done, close the connection by preferring the operation number 4.



The screenshot shows the same Java application window. The console displays the same menu as in Figure 46, but with the number '4' highlighted in green, indicating it has been selected. Below the menu, the text "Connection is closed!" is displayed.

```
1. Inserting the data into table.  
  
2. Displaying the data from table.  
  
3. Displaying the result of JOIN SQL processing.  
  
4. No operation left, close the connection.  
  
4  
Connection is closed!
```

Figure 47. Closing the connection

->Close the program and exit from the Eclipse IDE.

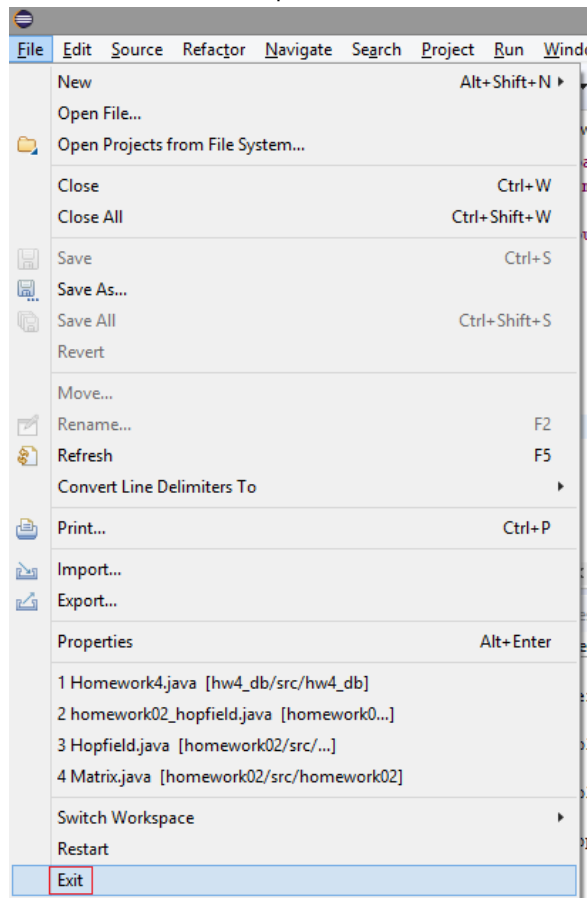


Figure 48. Exit the program

->Shutdown the Oracle connection from cmd. Type 'shutdown immediate' in the command line.

3. 논의 (10점)

-Describe new things you learned during your homework (outside of class), what you would like to cover in class, and what you had trouble doing during your homework.

I found out this homework really helpful for myself since I learned how to do some basic database connections and how to deal with the JDBC. To be honest, for me it took more time to set up all of the programs before doing the homework itself. Especially, I had the program with NLS_LANG since conversion from Russian character set into English is not performed as it is done with other languages. But, eventually I found the solution and I was able to connect the database. What about the program itself, it was not so difficult but at the same time I learned even more about the SQL Query execution and how it done with the JDBC. The only thing that I want to highlight is that even though I translate the homework's requirements via the translator, it can be

wrong in some cases, so my answer may answer may differ. Hope, you will understand my situation.