Lab4: Java Database Connectivity (JDBC)

Subject

Preamble

Follow these steps to create a JDBC project :

1. Create a plain project on your java IDE
2. Installing the MySQL driver: put `mysql-connector-java-x.y.z.jar` in the lib directory
3. Setting the name of the driver : `System.setProperty("jdbc.drivers", "com.mysql.jdbc.Driver");`
4. Setting the database's URL : `jdbc:mysql://[host][:port]/[database]`

Please refer to the subject of this lab for any further information.

Exercice 1

Constructor of the class :

```
public DataAccess(String url, String login, String password) throws
    SQLException {
        try{
            conn = DriverManager.getConnection(url, login, password);
            System.out.println("connected to " + url);
        }
        catch(SQLException e) {
            System.err.println("Connexion problem");
        }
    }
```

Exercice 2

Write the method `List<EmployeeInfo> getEmployees()` that returns the number, name and salary of all the employee in the EMP table.

```
public List<EmployeeInfo> getEmployee()
{
    List<EmployeeInfo> returnList = new ArrayList<>();
    try {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * FROM emp");
        while(rs.next()) {
            returnList.add(new EmployeeInfo(rs.getInt(1),rs.getString(1),
rs.getFloat(2)));
        }
        }catch(SQLException ex) {
```

```
            System.err.println("The query isn't possible");
        }
    return returnList;
}
```

Exercice 3 : SQL injection attack

Write the method boolean `raiseSalary(String job, float amount)` that raises the salary of the employees with the specified job by the specified amount.

```
boolean raiseSalary(String job, float amount)
{
    boolean returnRaise = false;
    try {
        Statement stmt = conn.createStatement();
        stmt.executeQuery("UPDATE emp SET salary = salary + "+amount+" WHERE job =
"+job);
    }catch(SQLException ex) {
    System.err.println("The query isn't possible");
}
    return returnRaise;
}
```

In order to perform an attack to increase the salary of every employees the user can replace the String `job` by `*`. It will increase the salary of every employees.

Exercice 4

Write a second version of the `getEmployees` and `raiseSalary`, named `getEmployeesPS` and `raiseSalaryPS`, that uses prepared statements instead of statements.

```
public List<EmployeeInfo> getEmployeePS()
{
    List<EmployeeInfo> returnList = new ArrayList<>();
    try {
        PreparedStatement prep = conn.prepareStatement("SELECT * FROM emp");
        ResultSet rs = prep.executeQuery();
        while(rs.next()) {
            returnList.add(new EmployeeInfo(rs.getInt(1),rs.getString(1),
rs.getFloat(2)));
        }
        }catch(SQLException ex) {
            System.err.println("The query isn't possible");
        }
    return returnList;
}
```

```
boolean raiseSalaryPS(String job, float amount)
{
    boolean returnRaise = false;
    try {
        PreparedStatement prep = conn.prepareStatement("UPDATE emp SET salary = ?
+ WHERE job = ?");
        prep.setString(1, job);
        prep.setFloat(1,amount);
        returnRaise = true;
    }catch(SQLException ex) {
    System.err.println("The query isn't possible");
    returnRaise = false;
}
    return returnRaise;
}
```

Prepared statements helps performance because it does pre-processing to speed-up queries, it also reduce load on the DataBase Sources. To take advantage of them it's necessary to use the right types to parametrized SQL queries.

Exercice 5

Write the method `List<DepartmentInfo> getDepartments(Integer id, String name, String location)` that retrieves the departments matching the specified criteria. A criterion may be omitted by specifying the Java null value.

```
public List<DepartmentInfo> getDepartments(Integer id, String name, String
location)
{
    List<DepartmentInfo> returnList = new ArrayList<>();
    try {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT *"
                + " FROM    dept"
                + " WHERE   (DID ="+id+" OR "+id+" IS NULL)"
                + " AND     (DNAME ="+name+" OR "+name+" IS NULL)"
                + " AND     (DLOC ="+location+" OR "+location+" IS NULL)");
        while(rs.next()) {
            returnList.add(new DepartmentInfo(rs.getInt(1),rs.getString(1),
rs.getString(2)));
        }
        }catch(SQLException ex) {
            System.err.println("The query isn't possible");
        }
    return returnList;
}
```

```
public List<DepartmentInfo> getDepartmentsPS(Integer id, String name, String
location)
{
    List<DepartmentInfo> returnList = new ArrayList<>();
    try {
        PreparedStatement prep = conn.prepareStatement("SELECT *"
            + " FROM    dept"
            + " WHERE   (DID = ? OR ? IS NULL)"
            + " AND     (DNAME = ? OR ? IS NULL)"
            + " AND     (DLOC = ? OR ? IS NULL)");
        ResultSet rs = prep.executeQuery();
        prep.setInt(1, id);
        prep.setInt(2, id);
        prep.setString(1, name);
        prep.setString(2, name);
        prep.setString(3, location);
        prep.setString(4, location);
        while(rs.next()) {
            returnList.add(new DepartmentInfo(rs.getInt(1),rs.getString(1),
rs.getString(2)));
        }
        }catch(SQLException ex) {
            System.err.println("The query isn't possible");
        }
    return returnList;
}
```

Exercice 6

Write the method `List<String> executeQuery(String query)` that executes the specified query (i.e. select statement) on the database.

```
List<String> executeQuery(String query){
    List<String> returnList = new ArrayList<>();
    try {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(query);
        while(rs.next()) {
                returnList.add(rs.getString(1));
            }
    }catch(SQLException ex) {
        System.err.println("The query isn't possible");
    }
    return returnList;
}
```

Write the method `List<String> executeStatement(String statement)` that executes any statement (e.g. select, insert, update, etc.) on the database.

```
List<String> executeStatement(String statement){
    List<String> returnList = new ArrayList<>();
      try {
          Statement stmt = conn.createStatement();
          ResultSet rs = stmt.executeQuery(statement);
          while(rs.next()) {
                returnList.add(rs.getString(1));
            }
      }catch(SQLException ex) {
          System.err.println("The query isn't possible");
      }
      return returnList;
}
```