Islamic University of Technology

Database Management Systems Lab

CSE 4308 / CSE 4174

Lab 7 (Pre-cursor)

Author:
Ishmam Tashdeed
Md. Tariquzzaman
Zannatun Naim Sristy
CSE, IUT

Contents

1 Introduction to JDBC

Class and Object Example

The code takes user input and prints the name and ID.

```java
import java.util.Scanner;

public class HelloWorld {
    private static class MyClass {
        public int id;
        public String name;
    }

    public static void main(String[] args) {
        MyClass myObj = new MyClass();
        Scanner input = new Scanner(System.in);

        System.out.println("What's your ID?");
        myObj.id = input.nextInt();
        myObj.name = "ABC";

        System.out.println("My name is " + myObj.name + ", ID
    " + myObj.id);
    }
}
```

Try and Catch Example

The following code demonstrates basic exception handling in Java. It attempts division by zero and catches the exception, handling it properly.

```java
public class TryCatchExample {
    public static void main(String[] args) {
        int number, result;
        number = 40;

        try {
            result = number / 0;
            System.out.println(result);
        } catch (ArithmeticException e) {
            System.out.println("Error in try block. Exception
   : " + e);
            result = number / 10;
            System.out.println(result);
        } catch (Exception e) {
            System.out.println("Error in try block. Exception
   : " + e);
            result = number / 0;
        }

        System.out.println("Code completed.");
    }
}
```

Sample JDBC Code Snippet

This code demonstrates connecting to a database using JDBC, executing a query, and handling SQL exceptions.

```
import java.sql.*;

public class Main {
    public static void main(String[] args) {
        String username = "senpai";
        String password = "senpai";
        String url = "jdbc:oracle:thin:@localhost:1521/XE";
        String sqlQuery = "SELECT AVG(AGE) FROM CUSTOMER";
        double avgAge = 0;

        try {
            // 1) Register the driver class
            Class.forName("oracle.jdbc.driver.OracleDriver");

            // 2) Create the connection object
            Connection con = DriverManager.getConnection(url,
    username, password);
            System.out.println("Connection to database
   successful");

            // 3) Create the statement object
            Statement statement = con.createStatement();

            // 4) Execute the query
            ResultSet result = statement.executeQuery(
   sqlQuery);
            while (result.next()) {
                avgAge = result.getDouble(1);
            }
            System.out.println("Average age: " + avgAge);
            // 5) Close the connection object
            con.close();
            statement.close();
            result.close();

        } catch (SQLException e) {
            System.out.println("Error while connecting to
    database. Exception code: " + e);
        } catch (ClassNotFoundException e) {
            System.out.println("Failed to register driver.
    Exception code: " + e);
        }
    }
}
```

This code establishes a JDBC connection to an Oracle database using the provided credentials. It then executes an SQL query to find the average age from the `CUSTOMER` table and prints the result. The connection, statement, and result set are closed afterward. Exception handling for `SQLException` and `ClassNotFoundException` ensures proper handling of database errors.

Prepared Statements

`PreparedStatements` use "?" as values, specifying that the actual values will be provided later. This has a number of very important advantages. Each time the query is executed (with new values to replace the "?"s), the database system can reuse the previously compiled form of the query and apply the new values as parameters. The following code fragment shows how `PreparedStatements` can be used:

```
    PreparedStatement pStmt = conn.prepareStatement(
        "insert into instructor values(?, ?, ?, ?)"
        );
    pStmt.setString(1 , " 88877 ");
    pStmt.setString(2 , " Perry ");
    pStmt.setString(3 , " Finance ");
    pStmt.setInt(4 , 125000);
    pStmt.executeUpdate();
```

Protection against SQL injection attacks. The `setString()` method automatically inserts any escape characters needed for syntactic correctness, and so, malicious users cannot manipulate SQL code to steal data or damage the database.

Metadata

Recall that when we submit a query using the `executeQuery()` method, the result of the query is contained in a `ResultSet` object. The interface `ResultSet` has a method, `getMetaData()`, that returns a `ResultSetMetaData` object that contains metadata about the result set.

In turn, `ResultSetMetaData`, has methods to find metadata information, such as the number of columns in the result, the name of a specified column, or the type of a specified column. The following Java code segment uses JDBC to print out the names and types of all columns of a result set:

```
    ResultSetMetaData rsmd = rs.getMetaData() ;
    for (int i = 1; i <= rsmd.getColumnCount() ; i ++) {
        System.out.println( rsmd.getColumnName(i)) ;
        System.out.println( rsmd.getColumnTypeName(i)) ;
    }

// The getColumnCount () method returns the arity ( number of
    attributes ) of the result relation.
// For each attribute , we retrieve its name and data type
   using the methods getColumnName () and getColumnTypeName
   () respectively.
```

2 Environment Setup

All the files mentioned below have been provided in Google Classroom.

1. Install Java Development Kit (JDK).

2. If you want to use VSCode, install the extension pack for Java.

3. Create a new Java Project and add `ojdbc14.jar` or `ojdbc6.jar` file as an external JAR file.

   (a) For IntelliJ IDE, in the video, the whole process has been demonstrated.

   (b) For VSCode IDE, the process of adding an external JAR file to a project: from the Explorer Bar, go to `Java Projects` → `<projectname>` → `Referenced Libraries`. Then click on the plus sign and select the `ojdbc14.jar` or `ojdbc6.jar` and hit the select `jar Libraries` button.

   (c) For Eclipse IDE, the process of adding an external JAR file to a project: from the Menu Bar, go to `Project` → `Properties`. Then on the opened window, click on `Java Build Path` → `Classpath` → `Add External JARs`. This will open a File Explorer where you can navigate to the path where `ojdbc14.jar` or `ojdbc6.jar` is located to select and add it to your project.

4. Now add a new file named `jdbc_practice.java` file (given) and go through the code to get an idea about what is going on.