Java Programming 6-1: JDBC Introduction

Practice Activities

JDBC Introduction and Setup

JDBC (Java Database Connectivity) is an API that allows Java applications to interact with databases. It provides a standard interface for connecting to databases, executing SQL statements, and retrieving results.

Vocabulary Definitions

1.

The JDBC-ODBC Bridge plus ODBC Driver

2.

- **JDBC-ODBC Bridge**: A JDBC driver that translates JDBC calls into ODBC calls, allowing Java applications to interact with databases using the ODBC driver. (Note: This bridge was deprecated and removed in JDK 8.)

3.

The interface used for executing a static SQL statement and returning the results it produces

4.

- **Statement Interface**: A JDBC interface used to execute SQL queries and update statements.

5.

The class which manages a set of JDBC Driver

6.

- **DriverManager**: A JDBC class that manages a list of database drivers and establishes a connection to the database.

7.

The interface which is able to provide information for the tables, stored procedures and so on

8.

- **DatabaseMetaData Interface**: Provides information about the database, including tables, stored procedures, and other database objects.

1. Set the JDBC Runtime Environment

To use JDBC in your Java application, ensure the following JAR files are included in your project's classpath:

- `ojdbc8.jar` (Oracle JDBC Driver)
- `orai18n.jar` (Oracle Internationalization Support)

You can set the classpath in your IDE or build tool (e.g., Maven or Gradle) to include these JAR files.

2. Identify the Structure of Database Tables

To view the structure of the tables in an Oracle database using SQLPlus, follow these steps:

1.

Log in to SQLPlus:

2.

```
sqlplus orcluser/password

DESCRIBE job_grades;
DESCRIBE job_history;
DESCRIBE employees;
DESCRIBE jobs;
DESCRIBE departments;
DESCRIBE locations;
DESCRIBE countries;
DESCRIBE regions;
```

3. Write a Java Application to Display the JOBS Table Information

Here's a basic Java application that connects to an Oracle database using JDBC, retrieves data from the `JOBS` table, and displays the `JOB_ID`, `JOB_TITLE`, `MIN_SALARY`, and `MAX_SALARY` columns.

Java Code Example

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
```

```
public class JobsTableDisplay {
   // JDBC URL, username, and password of MySQL server
   private static final String URL = "jdbc:oracle:thin:@localhost:1521:orcl";
   private static final String USER = "orcluser";
   private static final String PASSWORD = "password";

   // JDBC variables for opening, managing, and closing connection
   private static Connection connection;
   private static Statement statement;

   public static void main(String[] args) {
      try {
         // Load JDBC driver
         Class.forName("oracle.jdbc.driver.OracleDriver");

         // Establish a connection
         connection = DriverManager.getConnection(URL, USER, PASSWORD);

         // Create a statement
         statement = connection.createStatement();

         // Execute a query
         String query = "SELECT JOB_ID, JOB_TITLE, MIN_SALARY, MAX_SALARY FROM JOBS";
         ResultSet resultSet = statement.executeQuery(query);

         // Process the result set
         System.out.println("JOB_ID\tJOB_TITLE\tMIN_SALARY\tMAX_SALARY");
         while (resultSet.next()) {
            String jobId = resultSet.getString("JOB_ID");
            String jobTitle = resultSet.getString("JOB_TITLE");
            double minSalary = resultSet.getDouble("MIN_SALARY");
            double maxSalary = resultSet.getDouble("MAX_SALARY");
            System.out.printf("%s\t%s\t%.2f\t%.2f%n", jobId, jobTitle, minSalary, maxSalary);
         }

         // Close the resources
         resultSet.close();
         statement.close();
         connection.close();
      } catch (Exception e) {
         e.printStackTrace();
      }
   }
}
```

Explanation of the Java Code

1.

Loading the Driver:

2.

3.

```
java
```

```
Copy code
```

```
"oracle.jdbc.driver.OracleDriver"
```

This line loads the Oracle JDBC driver.

Establishing a Connection:

```
java
```

```
Copy code
```

This line creates a connection to the Oracle database using the provided URL, username, and password.

Creating a Statement:

```
java
```

```
Copy code
```

This line creates a `Statement` object for sending SQL statements to the database.

Executing a Query:

java

Copy code

```
ResultSet resultSet
```

This line executes the SQL query and retrieves the results into a `ResultSet`.

Processing the Result Set:

java

Copy code

```
while
```

49.

This loop iterates through the `ResultSet` to fetch and print the data.

50.

51.

Closing Resources:

52.

These lines close the `ResultSet`, `Statement`, and `Connection` to free up database resources.

53.

This example demonstrates how to use JDBC to connect to an Oracle database, execute a query, and display results in Java. Make sure to adjust the `URL`, `USER`, and `PASSWORD` to match your database configuration.