Task 6: Create a Database Connection Code

Submitted by: HARSH BHATURKAR
Date: January 11 2026
GitHub Repository: https://github.com/harshbhaturkar1404/OracleJDBCConnection

1. Goal
To establish a connection between a Java application and a MySQL database using JDBC, verify the connection is successful, and handle any potential exceptions.
2. Steps Followed
I followed these steps to complete the task using IntelliJ IDE and MySQL:
1. Project Setup: Created a new Java project in IntelliJ IDE named Task1

2. Library Configuration: Downloaded the MySQL Connector/J JAR file and added it to the project's Build Path (Classpath) to allow Java to communicate with MySQL.

3. Database Preparation: Created a database named rstask in MySQL to serve as the connection target.

4. Driver Loading: Used Class.forName("com.mysql.cj.jdbc.Driver") to dynamically load the MySQL driver class.

5. Establishing Connection: Used DriverManager.getConnection() with the URL jdbc:mysql://localhost:3306/rstask, along with my username and password.

6. Verification: Added a conditional check to print "Connection Successful!" if the connection object was not null.

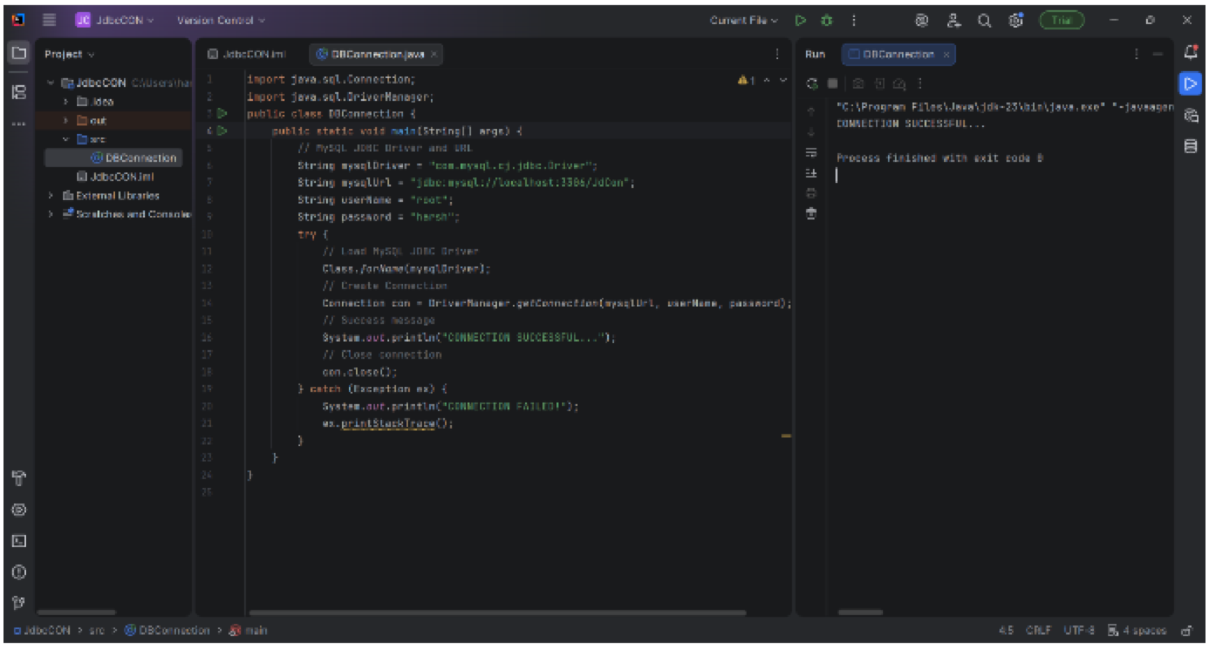
7. Resource Management: Implemented con.close() to properly close the connection and free system resources after execution.

8. Exception Handling: Wrapped the code in a try-catch block to handle ClassNotFoundException (for missing drivers) and SQLException (for connection errors

Source Code:

```
import java.sql.Connection;

import java.sql.DriverManager;

public class DBConnection {
  public static void main(String[] args) {


    String mysqlDriver = "com.mysql.cj.jdbc.Driver";//DataBase Connection Detail

    String mysqlUrl = "jdbc:mysql://localhost:3306/JdCon";

    String userName = "root";

    String password = "harsh";

    try {


      Class.forName(mysqlDriver);


      Connection con = DriverManager.getConnection(mysqlUrl, userName,
password);

      //Connection with database

      System.out.println("CONNECTION SUCCESSFUL...");


      con.close();

    } catch (Exception ex) {

      System.out.println("CONNECTION FAILED!");

      ex.printStackTrace();
      }

    }

}
```

output:



```
import java.sql.Connection;
import java.sql.DriverManager;
public class DBConnection {
    public static void main(String[] args) {
        // MySQL JDBC Driver and URL
        String mysqlDriver = "com.mysql.cj.jdbc.Driver";
        String mysqlUrl = "jdbc:mysql://localhost:3306/JdCon";
        String userName = "root";
        String password = "harsh";
        try {
            // Load MySQL JDBC Driver
            Class.forName(mysqlDriver);
            // Create Connection
            Connection con = DriverManager.getConnection(mysqlUrl, userName, password);
            // Success message
            System.out.println("CONNECTION SUCCESSFUL...");
            // Close connection
            con.close();
        } catch (Exception ex) {
            System.out.println("CONNECTION FAILED!");
            ex.printStackTrace();
        }
    }
}
```

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagen
CONNECTION SUCCESSFUL...

Process finished with exit code 0
```

Code Explanation

- Class.forName("com.mysql.cj.jdbc.Driver"): This line explicitly loads the MySQL JDBC driver class into memory, ensuring the Java application can communicate with the MySQL server.

- DriverManager.getConnection(url, user, pass): This method establishes the actual connection to the database using the specified connection string (URL), username, and password.

- jdbc:mysql://localhost:3306/rstask: This is the connection URL. It specifies the protocol (jdbc:mysql), the server address (localhost), the port (3306), and the specific database name (rstask).

- con.close(): This is a critical step that closes the database connection to release resources and prevent memory leaks.

- try-catch Block: The code is wrapped in a try-catch block to gracefully handle potential runtime errors, such as a missing driver (ClassNotFoundException) or invalid credentials (SQLException).