A CallableStatement with ResultSet object

A `CallableStatement` in Java is used to execute SQL stored procedures in a database. It can return multiple result sets, update counts, or output parameters. When a stored procedure returns a `ResultSet`, the `CallableStatement` can retrieve it. Below is an example program demonstrating how a `CallableStatement` can return one `ResultSet` object.

Example Program

In this example, assume you have a stored procedure in your database named `getEmployees` that returns a list of employees. We'll show how to call this stored procedure from Java using `CallableStatement` and retrieve the result set.

Step 1: Stored Procedure (Database Side)

You would first need to create a stored procedure in your database. Here's an example for MySQL:

```
DELIMITER $$

CREATE PROCEDURE getEmployees()
BEGIN
    SELECT employee_id, first_name, last_name FROM employees;
END$$

DELIMITER ;
```

This stored procedure retrieves the `employee_id`, `first_name`, and `last_name` from the `employees` table.

Step 2: Java Program

Now, let's write the Java program to call this stored procedure using `CallableStatement` and retrieve the `ResultSet`:

```
import java.sql.*;

public class CallableStatementExample {
    public static void main(String[] args) {
        // Database connection variables
        String url = "jdbc:mysql://localhost:3306/your_database_name";
```

```
        String username = "your_username";
        String password = "your_password";

        // Connection and CallableStatement objects
        Connection conn = null;
        CallableStatement stmt = null;
        ResultSet rs = null;

        try {
            // Establish the connection to the database
            conn = DriverManager.getConnection(url, username, password);

            // Call the stored procedure
            String sql = "{CALL getEmployees()}";
            stmt = conn.prepareCall(sql);

            // Execute the stored procedure
            rs = stmt.executeQuery();

            // Process the result set
            while (rs.next()) {
                int employeeId = rs.getInt("employee_id");
                String firstName = rs.getString("first_name");
                String lastName = rs.getString("last_name");

                // Print employee details
                  System.out.println("Employee ID: " + employeeId + ", Name: " + firstName + " "
+ lastName);
            }

        } catch (SQLException e) {
            e.printStackTrace();
        } finally {
            // Close the ResultSet, CallableStatement, and Connection
            try {
                if (rs != null) rs.close();
                if (stmt != null) stmt.close();
                if (conn != null) conn.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Explanation of the Code:

1. **Connection Setup:**
    - The connection to the MySQL database is established using `DriverManager.getConnection()`, providing the database URL, username, and password.
2. **CallableStatement Creation:**
    - The SQL query `"{CALL getEmployees()}"` is used to call the stored procedure `getEmployees`.
    - The `{CALL}` syntax is used in JDBC to call a stored procedure.
3. **Executing the Stored Procedure:**
    - `stmt.executeQuery()` is used to execute the stored procedure. Since `getEmployees` returns a `ResultSet`, the `executeQuery` method is appropriate for retrieving the result set.
4. **Processing the ResultSet:**
    - The `ResultSet` object (`rs`) contains the result of the stored procedure. We iterate over the result set using `rs.next()` and retrieve the employee details using the appropriate `get` methods (e.g., `getInt()` for integers, `getString()` for strings).
5. **Resource Cleanup:**
    - It is important to close the `ResultSet`, `CallableStatement`, and `Connection` objects to free up resources once the operation is complete. This is done in the `finally` block to ensure it runs regardless of whether an exception occurs.

Key Points:

- **CallableStatement:** It is specifically used for calling stored procedures.
- **ResultSet:** The `ResultSet` returned by `executeQuery()` can be processed just like any other result set.
- **SQL Procedure Call:** The syntax `{CALL procedure_name()}` is used to execute a stored procedure. If the procedure returns a `ResultSet`, it can be retrieved via the `executeQuery()` method.
-

Output Example:

Assuming the `employees` table has the following data:

employee_id	first_name	last_name
1	John	Doe
2	Jane	Smith

The output of the program would be:

Employee ID: 1, Name: John Doe
Employee ID: 2, Name: Jane Smith

Conclusion:

This example demonstrates how a `CallableStatement` can be used to call a stored procedure in the database that returns a `ResultSet`. It is a powerful tool when working with stored procedures, allowing you to execute complex database operations from Java while handling the result set efficiently.