JDBC

CallableStatement

Stored Procedures

- relational database itself can contain pre-defined stored procedures
- `CallableStatement` allows you to make use of the stored procedure
- for the exam, you don't need to know how to create stored procedures
  - you only need to know how to execute them in Java code
- stored procedures differ regarding to the parameters they take:
  - no parameters
  - IN - an input parameter
  - OUT - an output parameter
  - INOUT - a parameter that serves for both input and output

```
// using stored procedure read_phone_by_name(first_name) (IN parameter)
String url = "jdbc:postgresql://localhost/phonebook_db";
String procedureCall = "{call read_phone_by_name(?)}";
try (Connection conn = DriverManager.getConnection(url, "luka", "luka123");
     CallableStatement cs = conn.prepareCall(procedureCall)) {
    cs.setString(1, "John");
    boolean hasResults = cs.execute();
    if (hasResults) {
      ResultSet rs = cs.getResultSet();
      while (rs.next()) {
        System.out.println(rs.getString("first_name") + ": " + rs.getString("phone"));
      }
    } else { System.out.println("No results."); }
}
// catch exception
```

parameter is entered as "?"

setting input (IN) parameter

executing stored procedure

getting the ResultSet

processing the ResultSet

```
// using stored procedure read_phone_by_name(first_name, phone) (IN, OUT)

String url = "jdbc:postgresql://localhost/phonebook_db";

String procedureCall = "{call read_phone_by_name(?, ?)}";

try (Connection conn = DriverManager.getConnection(url);
     CallableStatement cs = conn.prepareCall(procedureCall)) {
    cs.setString(1, "John");
    cs.registerOutParameter(2, Types.VARCHAR);
    cs.execute();
    String phone = cs.getString(2);
    System.out.println("Phone number for John: " + phone);
}

// catch exception
```

procedure has two parameters

setting input (IN) parameter

registering the output (OUT) parameter

executing stored procedure

retrieving the output (OUT) parameter value