Key differences between JDBC Statement and PreparedStatement objects

Now, `Statement` objects are fine when you have a single, static SQL statement to execute.

But if you are executing several just-slightly-different SQL statements, or "building" a dynamic SQL statement based on user input (when SQL injection becomes a danger), a `PreparedStatement` object is a better choice, when it can be used. A `PreparedStatement` is precompiled, so executing it multiple times is more efficient than executing a `Statement` the same number of times, and because it limits exactly where user input is included, it can reduce (although not eliminate) some SQL injection attacks.

So, here are the pertinent differences you need to know, so you can try out a `PreparedStatement`:

- For each part or parts in a SQL statement string that you would like to replace with user input or update before the next of several repeated executions, put a `?`. For example:

```
String insertPrepString = "insert into log_table "
                          + "values "
                          + "(?, sysdate)";
String emplQuery = "select * "
                   + "from empl "
                   + "where empl_last_name = ? "
                   + "     or salary > ?";
```

- Use the `Connection` object's method `prepareStatement`, which expects as its argument the SQL string containing those `?` instances, to create your `PreparedStatement`:

```
PreparedStatement insertPrepStmt =
    con.prepareStatement(insertPrepString);
PreparedStatement emplPrepStmt =
    con.prepareStatement(emplQuery);
```

- Use one of the `PreparedStatement`'s `set` methods -- whose type is appropriate for the type of the value that should appear where the `?` appears -- to set each of the `?` instances in the `PreparedStatement` object to the value to be used in its next execution.

- Note that these expect the 1-based position of the `?`, and the value the `?` is to be set to, as their arguments.
- Some examples of these `set` methods: `setDouble`, `setInt`, `setString`, and quite a few more!
- For example:

```
insertPrepStmt.setString(1, nextName);  // nextName is a String
emplPrepStmt.setString(1, chosenName);  // chosenName is a String
emplPrepStmt.setInt(2, chosenMinSal);   // chosenMinSal is an int
```

- Finally, call the `PreparedStatement` object's `executeQuery` or `executeUpdate` method, depending on the kind of SQL statement being executed, but with NO arguments:

```
insertPrepStmt.executeUpdate();          // executing the insert
ResultSet desiredEmpls = emplPrepStmt.executeQuery(); // for a select
```

- Before each execution of that prepared statement, make sure you call the `PreparedStatement`'s `set` methods to reset the values of its `?` instances for that next execution.
- And, when you are done with all of the desired executions of that prepared statement, call the `close` method of that `PreparedStatement` object (just as you would for a `Statement` object):

```
insertPrepStmt.close();
emplPrepStmt.close();
```