NOTE: SCENARIO IS WHAT PUZZLE TAKER SEES.

SCENARIO:

See the J57-sql-injection puzzle.

```
01   // OMITTED: Import whatever is needed.
02   public class DbUtils {
03     public static ResultSet getUser (String username)
04         throws SQLException {
05       String sql = "SELECT * FROM users WHERE username = ?";
06
07       // Get a connection from the connection pool:
08       Connection conn = DbUtils.getConnectionPool().getConnection();
09       PreparedStatement stmt = conn.prepareStatement(sql);
10       stmt.setString(1, username);
11       return stmt.executeQuery();
12     }
13   }
```

Questions:
See the J57-sql-injection puzzle.

NOTE: ANSWER IS TO BE SHOWN TO PUZZLE TAKER AT THE END OF SESSION.

ANSWER:
a
See the J57-sql-injection puzzle.

NOTE: THE REST OF DOCUMENT CONTAINS EXTRA INFORMATION FOR THE PROJECT RESEARCHERS. IT IS NOT TO BE SHOWN TO PUZZLE TAKERS.

TAGS:
java, preparedstatement-class, sql-injection, input-verification, input-sanitization

CATEGORIES:
Blindspot - NO
Type - SQL
Number of distinct functions - 5
Number of total functions - 5
Blindspot function - NA
Function call omitted - NA
Blindspot type - NA

Number of Parameters in the blindspot function - NA
Cyclomatic complexity - 2

NAME:
PreparedStatement class - An object that represents a precompiled SQL statement.

MORE INFORMATION:
To see the incorrect way of use of the API look at the J57-sql-injection puzzle.

REFERENCES:
1. https://www.owasp.org/index.php/SQL_Injection