Insecure SQL

What is wrong with the following code?

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Objects;
import java.util.Scanner;

public class SQLCode {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter admin username:");
    String username = scanner.nextLine();
    System.out.println("Enter admin password:");
    String password = scanner.nextLine();
    scanner.close();

    if (hasAdminAccess(username, password)) {
      // Admin only operations are now allowed!!
    }
  }

  private static boolean hasAdminAccess(final String username, final String password) {
    String query = "SELECT * FROM users WHERE username = '" + username + "' AND password = '" + password + "';";

    try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/myDatabase", "user", "password");
         Statement statement = connection.createStatement();
         ResultSet resultSet = statement.executeQuery(query)) {
      return Objects.nonNull(resultSet);
    } catch (Exception e) {
      e.printStackTrace();
      return false;
    }
  }
}
```

Can you come up with a way to exploit the vulnerability?

Click here to expand...

Vulnerability: SQL Injection

Example exploit 1: `admin'--` which makes the query

```sql
SELECT * FROM users WHERE username = 'admin'--' AND password = 'any_password';
```

which is equivalent to

```sql
SELECT * FROM users WHERE username = 'admin';
```

since `--` is used for single-line comments in SQl, and everything after `--` will be ignored.

Example exploit 2: `' OR 1=1--` which makes the query

```sql
SELECT * FROM users WHERE username = '' OR 1=1--' AND password = 'any_password';
```

which is equivalent to

```sql
SELECT * FROM users WHERE username = '' OR 1=1
```

since 1=1 is always true, the resulting query retrieves all user info.

Fix: use PreparedStatements to handle user input safely. PreparedStatements automatically escape user input and prevent SQL injection attacks:

```java
private static boolean hasAdminAccess(final String username, final String password) {
    String query = "SELECT * FROM users WHERE username = ? AND password = ?;";

    try (Connection connection = DriverManager.getConnection("jdbc:mysql://localhost:3306/myDatabase", "user", "password");
        PreparedStatement preparedStatement = connection.prepareStatement(query)) {

        preparedStatement.setString(1, username);
        preparedStatement.setString(2, password);

        try (ResultSet resultSet = preparedStatement.executeQuery()) {
            return resultSet.next();
        }
    } catch (Exception e) {
        e.printStackTrace();
```

```
    }
    return false;
}
```