Day 29 Assignment

Name: Mehul Anjikhane Email: mehulanjikhane13@gmail.com

Task 1: Establishing Database Connections
Write a Java program that connects to a SQLite database and prints out the connectionobject to confirm successful connection.

```
package jdbc.assignments;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class DatabaseConnection {
	public static void main(String[] args) {
		String url = "jdbc:mysql://localhost:3306/database1";
		String user = "root";
		String password = "root";

		try (Connection conn = DriverManager.getConnection(url, user, password))
{
			if (conn != null) {
				System.out.println("Connection to MySQL has been
established.");
				System.out.println(conn);
			}
		} catch (SQLException e) {
			System.out.println(e.getMessage());
		}
	}
}
```

Output:

```
Connection to MySQL has been established.
com.mysql.cj.jdbc.ConnectionImpl@c0c2f8d
```

Task 2: SQL Queries using JDBC

Create a table 'User' with a following schema 'User ID' and 'Password' stored as hash format (note you have research on how to generate hash from a string), accept ""User ID"" and ""Password"" as input and check in the table if they match to confirm whether user access is allowed or not.

```
package jdbc.assignments;
import java.sql.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;
```

```java
public class UserAuthentication {
        private static Connection connect() {
                String url = "jdbc:mysql://localhost:3306/database1";
                String user = "root";
                String password = "root";
                Connection conn = null;
                try {
                        conn = DriverManager.getConnection(url, user, password);
                } catch (SQLException e) {
                        System.out.println(e.getMessage());
                }
                return conn;
        }
        private static String hashPassword(String password) {
                try {
                        MessageDigest md = MessageDigest.getInstance("SHA-256");
                        byte[] hash = md.digest(password.getBytes());
                        StringBuilder hexString = new StringBuilder();
                        for (byte b : hash) {
                                hexString.append(String.format("%02x", b));
                        }
                        return hexString.toString();
                } catch (NoSuchAlgorithmException e) {
                        throw new RuntimeException(e);
                }
        }
        private static void createNewTable() {
                String sql = "CREATE TABLE IF NOT EXISTS User (\n" + " UserID
VARCHAR(50) PRIMARY KEY,\n"
                                + " Password VARCHAR(64) NOT NULL\n" + ");";

                try (Connection conn = connect(); Statement stmt =
conn.createStatement()) {
                        stmt.execute(sql);
                } catch (SQLException e) {
                        System.out.println(e.getMessage());
                }
        }

        private static void insertUser(String userID, String password) {
                String sql = "INSERT INTO User(UserID, Password) VALUES(?, ?)";

                try (Connection conn = connect(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                        pstmt.setString(1, userID);
                        pstmt.setString(2, hashPassword(password));
                        pstmt.executeUpdate();
                } catch (SQLException e) {
                        System.out.println(e.getMessage());
                }
        }

        private static boolean authenticateUser(String userID, String password) {
                String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";
```

```java
		try (Connection conn = connect(); PreparedStatement pstmt = 
conn.prepareStatement(sql)) {
			pstmt.setString(1, userID);
			pstmt.setString(2, hashPassword(password));
			ResultSet rs  =  pstmt.executeQuery();
			return rs.next();
		} catch (SQLException e) {
			System.out.println(e.getMessage());
			return false;
		}
	}

	public static void main(String[] args) {
		createNewTable();

		Scanner scanner = new Scanner(System.in);
		System.out.println("Enter UserID:");
		String userID = scanner.nextLine();
		System.out.println("Enter Password:");
		String password = scanner.nextLine();

		insertUser(userID, password);

		System.out.println("Enter UserID to authenticate:");
		String authUserID = scanner.nextLine();
		System.out.println("Enter Password to authenticate:");
		String authPassword = scanner.nextLine();

		if (authenticateUser(authUserID, authPassword)) {
			System.out.println("User authenticated successfully.");
		} else {
			System.out.println("Authentication failed.");
		}
		scanner.close();
	}
}
```

Output:

```
Enter UserID:
101
Enter Password:
King
Enter UserID to authenticate:
101
Enter Password to authenticate:
King
User authenticated successfully.
```

Task 3: PreparedStatement

Modify the SELECT query program to use PreparedStatement to parameterize the query and prevent SQL injection.

```java
package jdbc.assignments;
```

```java
import java.sql.*;
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
import java.util.Scanner;

public class UserAuthentication1 {
	private static Connection connect() {
		String url = "jdbc:mysql://localhost:3306/database1";
		String user = "root";
		String password = "root";
		Connection conn = null;
		try {
			conn = DriverManager.getConnection(url, user, password);
		} catch (SQLException e) {
			System.out.println(e.getMessage());
		}
		return conn;
	}

	private static String hashPassword(String password) {
		try {
			MessageDigest md = MessageDigest.getInstance("SHA-256");
			byte[] hash = md.digest(password.getBytes());
			StringBuilder hexString = new StringBuilder();
			for (byte b : hash) {
				hexString.append(String.format("%02x", b));
			}
			return hexString.toString();
		} catch (NoSuchAlgorithmException e) {
			throw new RuntimeException(e);
		}
	}

	private static void createNewTable() {
		String sql = "CREATE TABLE IF NOT EXISTS User (\n" + " UserID 
VARCHAR(50) PRIMARY KEY,\n"
				+ " Password VARCHAR(64) NOT NULL\n" + ");";

		try (Connection conn = connect(); Statement stmt = 
conn.createStatement()) {
			stmt.execute(sql);
		} catch (SQLException e) {
			System.out.println(e.getMessage());
		}
	}

	private static void insertUser(String userID, String password) {
		String sql = "INSERT INTO User(UserID, Password) VALUES(?, ?)";

		try (Connection conn = connect(); PreparedStatement pstmt = 
conn.prepareStatement(sql)) {
			pstmt.setString(1, userID);
			pstmt.setString(2, hashPassword(password));
			pstmt.executeUpdate();
		} catch (SQLException e) {
```

```
                System.out.println(e.getMessage());
            }
    }

    private static boolean authenticateUser(String userID, String password) {
            String sql = "SELECT * FROM User WHERE UserID = ? AND Password = ?";

            try (Connection conn = connect(); PreparedStatement pstmt =
conn.prepareStatement(sql)) {
                pstmt.setString(1, userID);
                pstmt.setString(2, hashPassword(password));
                ResultSet rs  =  pstmt.executeQuery();
                return rs.next();
            } catch (SQLException e) {
                System.out.println(e.getMessage());
                return false;
            }
    }

    public static void main(String[] args) {
            createNewTable();

            Scanner scanner = new Scanner(System.in);
            System.out.println("Enter UserID:");
            String userID = scanner.nextLine();
            System.out.println("Enter Password:");
            String password = scanner.nextLine();

            insertUser(userID, password);

            System.out.println("Enter UserID to authenticate:");
            String authUserID = scanner.nextLine();
            System.out.println("Enter Password to authenticate:");
            String authPassword = scanner.nextLine();

            if (authenticateUser(authUserID, authPassword)) {
                System.out.println("User authenticated successfully.");
            } else {
                System.out.println("Authentication failed.");
            }
            scanner.close();
    }
}
```

Output:

```
Enter UserID:
102
Enter Password:
king
Enter UserID to authenticate:
102
Enter Password to authenticate:
King
Authentication failed.
```