Day 21:

Task 1: Establishing Database Connections

Write a Java program that connects to a SQLite database and prints out the connection object to confirm successful connection.

```
import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;


public class SQLiteConnectionExample {


   public static void main(String[] args) {

      Connection connection = null;


      try {

         Class.forName("org.sqlite.JDBC");

         String url = "jdbc:sqlite:/path/to/your/database.db"; // Replace this with the path to your SQLite database file

         connection = DriverManager.getConnection(url);


         if (connection != null) {

            System.out.println("Connected to the SQLite database.");

            System.out.println("Connection object: " + connection);

         } else {

            System.out.println("Failed to connect to the SQLite database.");

         }

      } catch (ClassNotFoundException e) {

         System.out.println("SQLite JDBC driver not found.");

         e.printStackTrace();

      } catch (SQLException e) {
```

```
            System.out.println("Failed to connect to the SQLite database.");
            e.printStackTrace();
        } finally {
            // Close the connection
            try {
                if (connection != null) {
                    connection.close();
                }
            } catch (SQLException e) {
                e.printStackTrace();
            }
        }
    }
}
```

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.

let's create the SQLite table 'User':

```
CREATE TABLE User (
    UserID TEXT PRIMARY KEY,
    PasswordHash TEXT
);
```

let's write the Java code to perform the authentication:

```
import java.sql.*;

public class UserAuthentication {

    public static void main(String[] args) {
        String url = "jdbc:sqlite:/path/to/your/database.db"; // Replace this with the path to your SQLite database file

        try (Connection connection = DriverManager.getConnection(url)) {
            String userId = "user123"; // Example user ID
            String password = "password123"; // Example password
            String hashedPassword = hashPassword(password);
            String sql = "SELECT * FROM User WHERE UserID = ? AND PasswordHash = ?";
            try (PreparedStatement statement = connection.prepareStatement(sql)) {
                statement.setString(1, userId);
                statement.setString(2, hashedPassword);

                ResultSet resultSet = statement.executeQuery();

                if (resultSet.next()) {
                    System.out.println("User authentication successful. Access granted.");
                } else {
                    System.out.println("Invalid credentials. Access denied.");
                }
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

```
    private static String hashPassword(String password) {
        return "hashed_" + password; // Replace this with your actual hashing logic
    }
}
```

Task 3: PreparedStatement

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

```
import java.sql.*;

public class UserAuthentication {

    public static void main(String[] args) {
        String url = "jdbc:sqlite:/path/to/your/database.db"; // Replace this with the path to your SQLite database file

        try (Connection connection = DriverManager.getConnection(url)) {
            String userId = "user123"; // Example user ID
            String password = "password123"; // Example password
            String hashedPassword = hashPassword(password);
            String sql = "SELECT * FROM User WHERE UserID = ? AND PasswordHash = ?";
            try (PreparedStatement statement = connection.prepareStatement(sql)) {
                statement.setString(1, userId);
                statement.setString(2, hashedPassword);

                ResultSet resultSet = statement.executeQuery();

                if (resultSet.next()) {
                    System.out.println("User authentication successful. Access granted.");
```

```
            } else {
                System.out.println("Invalid credentials. Access denied.");
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

private static String hashPassword(String password) {

    return "hashed_" + password; // Replace this with your actual hashing logic
}
}
```