```
/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
package presenter;

import java.io.BufferedWriter;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardOpenOption;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.ZoneId;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
import model.Campuses;
import model.Students;
import utility.DBConnection;

/**
 *
 * @author Adarsha
 */



class StudentPersister {

    private Connection connection;
    private PreparedStatement insertStudent;
    private PreparedStatement deleteStudent;
    private PreparedStatement getAllStudents;
    private PreparedStatement findAllStudentsByName;
    private PreparedStatement findAllStudentsById;
    private PreparedStatement loginStudentByUsernameAndPassword;
    private PreparedStatement updateStudent;


    public StudentPersister(){
        try {
                this.connection = DBConnection.getConnection(); //
database connection
                if (connection != null) {

                    insertStudent = connection.prepareStatement("INSERT
INTO students (student_name, username, password, email, phone, address,
campus) "
                            + "VALUES(?, ?, ?, ?, ?, ?, ?)");
```

```
                    getAllStudents = connection.prepareStatement("SELECT
* FROM students");
                    deleteStudent = connection.prepareStatement("DELETE
FROM students WHERE student_id = ?");
                    findAllStudentsById =
connection.prepareStatement("SELECT * FROM students WHERE student_id = ?
");
                    findAllStudentsByName =
connection.prepareStatement("SELECT * FROM students WHERE username LIKE ?
");
                    loginStudentByUsernameAndPassword =
connection.prepareStatement("SELECT * FROM students WHERE username = ?
AND password = ? ");
                    updateStudent = connection.prepareStatement("UPDATE
students SET student_name = ?, username = ?, password = ? ,email = ?,
phone = ?, address = ?, campus = ? WHERE student_id = ?");
                    }
            } catch (SQLException e) {
                System.out.println("Connection Failed!");
                System.out.println("SQLException : " + e.getMessage());
            }

    }



    public boolean registerStudent(Students student) {
       try {

            insertStudent.setString(1, student.getStudentName());
            insertStudent.setString(2, student.getUsername());
            insertStudent.setString(3, student.getPassword());
            insertStudent.setString(4, student.getEmail());
            insertStudent.setString(5, student.getPhone());
            insertStudent.setString(6, student.getAddress());
            insertStudent.setString(7, student.getCampus());
            insertStudent.executeUpdate();  // execute the prepared
statement insert
            return true;

        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            return false;
        }
    }

    static Date getDateFromLocalDate(LocalDate date) {
        Date newDate = null;
        if (date != null) {
            newDate =
Date.from(date.atStartOfDay(ZoneId.systemDefault()).toInstant());
        }
        return newDate;
    }

    static void writeToFile(String notification) {
        try {
```

```
          File myObj = new File("StudentLog.txt");
          if (myObj.createNewFile()) {
            System.out.println("File created: " + myObj.getName());
          } else {
            System.out.println("File already exists.");
          }
        } catch (IOException e) {
          System.out.println("An error occurred.");
          e.printStackTrace();
        }

        Path p = Paths.get("StudentLog.txt");
        try (BufferedWriter writer = Files.newBufferedWriter(p,
StandardOpenOption.APPEND)) {
            writer.write(notification+"\n");

            System.out.println("Successfully wrote to the file.");
        } catch (IOException e) {
          System.out.println("An error occurred.");
          e.printStackTrace();
        }
    }

    public List<Students> getAllStudentList() {
      List<Students> studentList = new ArrayList<>();
        try {
            ResultSet studentResult = getAllStudents.executeQuery();

            System.out.println("Students details reading from the
database.");
            while (studentResult.next()) {
                int studentId = studentResult.getInt("student_id");
                String studentName =
studentResult.getString("student_name");
                String studentCampus = studentResult.getString("campus");
                String studentPhone = studentResult.getString("phone");
                String studentAddress =
studentResult.getString("address");
                String studentUsername =
studentResult.getString("username");
                String studentPassword =
studentResult.getString("password");
                String studentEmail = studentResult.getString("email");


                Students newStudent = new
Students(studentName,studentPhone, studentCampus, studentUsername,
studentPassword,studentAddress,studentEmail);
                newStudent.setStudent_id(studentId);
                System.out.println("New added Student is : "+newStudent);
                studentList.add(newStudent);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        System.out.println("Final Student list to be sent from persister
is :"+ studentList);
        return studentList;
```

```
    }

    String deleteStudent(int student_id) {
        String studentStatus = "";
        try {

            deleteStudent.setInt(1, student_id);
            int studentResult = deleteStudent.executeUpdate();

            if (studentResult > 0) {
                studentStatus = "Student deleted successfully.";
            } else {
                studentStatus = "Cannot delete the Student.";
            }

        } catch (SQLException e) {
            studentStatus = "The Student cannot be deleted.";
            System.out.println("The student cannot be deleted : " +
e.getMessage());
        }
        return studentStatus;
    }

    List<Students> findStudentsByName(String keyword) {
       Students student = new Students();
       List<Students> searchedStudents = new ArrayList();
        try {
            findAllStudentsByName.setString(1, "%"+keyword+"%");
            ResultSet studentResult =
findAllStudentsByName.executeQuery();

            System.out.println("Student details reading from the
database.");
            while (studentResult.next()) {
                int studentId = studentResult.getInt("student_id");
                String studentName =
studentResult.getString("student_name");
                String studentCampus = studentResult.getString("campus");
                String studentPhone = studentResult.getString("phone");
                String studentAddress =
studentResult.getString("address");
                String studentEmail = studentResult.getString("email");
                String studentUsername =
studentResult.getString("username");
                String studentPassword =
studentResult.getString("password");




                student = new Students(studentName,studentPhone,
studentCampus, studentUsername,studentPassword, studentAddress,
studentEmail );
                student.setStudent_id(studentId);

                searchedStudents.add(student);
            }
        } catch (SQLException e) {
```

```
            System.out.println("SQL Exception: " + e.getMessage());
        }
        return searchedStudents;
    }

    List<Students> findStudentsById(int id) {
        Students student = new Students();
       List<Students> searchedStudents = new ArrayList();
        try {
            findAllStudentsById.setInt(1,id);
            ResultSet studentResult = findAllStudentsById.executeQuery();

            System.out.println("Student details reading from the
database.");
            while (studentResult.next()) {
                int studentId = studentResult.getInt("student_id");
                String studentName =
studentResult.getString("student_name");
                String studentCampus = studentResult.getString("campus");
                String studentPhone = studentResult.getString("phone");
                String studentAddress =
studentResult.getString("address");
                String studentEmail = studentResult.getString("email");
                String studentUsername =
studentResult.getString("username");
                String studentPassword =
studentResult.getString("password");




                student = new Students(studentName,studentPhone,
studentCampus, studentUsername,studentPassword, studentAddress,
studentEmail );
                student.setStudent_id(studentId);

                searchedStudents.add(student);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        return searchedStudents;
    }


    boolean isStudentExist(String username, String password) {
        try {
            loginStudentByUsernameAndPassword.setString(1, username);
            loginStudentByUsernameAndPassword.setString(2, password);
            ResultSet loginResult =
loginStudentByUsernameAndPassword.executeQuery();
            if (loginResult.next()) {
                return true;
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        return false;
```

```
    }

    boolean updateStudent(Students student) {
         try {
            int myID = student.getStudent_id();
            updateStudent.setInt(8, myID);

            updateStudent.setString(1, student.getStudentName());
            updateStudent.setString(2, student.getUsername());
            updateStudent.setString(3, student.getPassword());
            updateStudent.setString(4, student.getEmail());
            updateStudent.setString(5, student.getPhone());
            updateStudent.setString(6, student.getAddress());
            updateStudent.setString(7, student.getCampus());
            updateStudent.executeUpdate();  // execute the prepared
statement insert
            return true;

        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            return false;
        }
    }

}
```