```
/*
 * 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 utility.DBConnection;

/**
 *
 * @author Adarsha
 */
class CampusPersister {

    private Connection connection; // Connection object creation
    private PreparedStatement insertCampus;
    private PreparedStatement getAllCampuses;
    private PreparedStatement findAllCampusesByName;
    private PreparedStatement findAllCampusesById;
    private PreparedStatement deleteCampus;

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

                    insertCampus = connection.prepareStatement("INSERT
INTO campuses (campus_name, location, phone, address) "
                            + "VALUES(?, ?, ?, ?)");

                    getAllCampuses = connection.prepareStatement("SELECT
* FROM campuses");
                    findAllCampusesById =
connection.prepareStatement("SELECT * FROM campuses WHERE campus_id = ?
");
                    findAllCampusesByName =
connection.prepareStatement("SELECT * FROM campuses WHERE campus_name
LIKE ? ");
```

```
                    deleteCampus = connection.prepareStatement("DELETE
FROM campuses WHERE campus_id = ?");
                }
            } catch (SQLException e) {
                System.out.println("Connection Failed!");
                System.out.println("SQLException : " + e.getMessage());
            }
    }

    public boolean registerCampus(Campuses campus) {
        try {
            insertCampus.setString(1, campus.getCampusName());
            insertCampus.setString(2, campus.getCampusLocation());
            insertCampus.setString(3, campus.getCampusPhone());
            insertCampus.setString(4, campus.getCampusAddress());
            insertCampus.executeUpdate();  // execute the prepared
statement insert
            return true;
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            return false;
        }
    }

    List<Campuses> getAllCampuses() {
        List<Campuses> campusList = new ArrayList<>();
        try {
            ResultSet campusResult = getAllCampuses.executeQuery();

            System.out.println("Campus details reading from the
database.");
            while (campusResult.next()) {
                int campusId = campusResult.getInt("campus_id");
                String campusName =
campusResult.getString("campus_name");
                String campusLocation =
campusResult.getString("location");
                String campusPhone = campusResult.getString("phone");
                String campusAddress = campusResult.getString("address");

                Campuses newCampus = new Campuses(campusName,
campusLocation, campusPhone, campusAddress);
                newCampus.setCampus_id(campusId);
                System.out.println("New added campus is : "+newCampus);
                campusList.add(newCampus);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        System.out.println("Final campus list to be sent from persister
is :"+ campusList);
        return campusList;
    }

    List<Campuses> findCampusesByName(String keyword) {
       Campuses campus = new Campuses();
       List<Campuses> searchedCampuses = new ArrayList();
        try {
```

```
            findAllCampusesByName.setString(1, "%"+keyword+"%");
            ResultSet campusResult =
findAllCampusesByName.executeQuery();

            System.out.println("Campus details reading from the
database.");
            while (campusResult.next()) {
                int campusId = campusResult.getInt("campus_id");
                String campusName =
campusResult.getString("campus_name");
                String campusLocation =
campusResult.getString("location");
                String campusPhone = campusResult.getString("phone");
                String campusAddress = campusResult.getString("address");


                campus = new
Campuses(campusName,campusLocation,campusPhone,campusAddress);
                campus.setCampus_id(campusId);

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

    List<Campuses> findCampusesById(int id) {
       Campuses campus = new Campuses();
       List<Campuses> searchedCampuses = new ArrayList();
        try {
            findAllCampusesById.setInt(1, id);
            ResultSet campusResult = findAllCampusesById.executeQuery();

            System.out.println("Campus details reading from the
database.");
            while (campusResult.next()) {
                int campusId = campusResult.getInt("campus_id");
                String campusName =
campusResult.getString("campus_name");
                String campusLocation =
campusResult.getString("location");
                String campusPhone = campusResult.getString("phone");
                String campusAddress = campusResult.getString("address");


                System.out.println("Campus id : "+ campusId);
                System.out.println("CampusName : "+ campusName);

                campus = new
Campuses(campusName,campusLocation,campusPhone,campusAddress);
                campus.setCampus_id(campusId);

                searchedCampuses.add(campus);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
```

```
        }
        System.out.println("The selected campus is : "+
searchedCampuses);
        return searchedCampuses;


    }

    public String deleteCampus(int campus_id) {
       String campusStatus = "";
        try {

            deleteCampus.setInt(1, campus_id);
            int campusResult = deleteCampus.executeUpdate();

            if (campusResult > 0) {
                campusStatus = "Campus deleted successfully.";
            } else {
                campusStatus = "Cannot delete the Campus.";
            }

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

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

    void writeToFile(String notification) {
         try {

          File myObj = new File("AdminLog.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("AdminLog.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();
        }
    }

}
```