```
/*
 * 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.BorrowedEquipments;
import model.Equipments;
import model.Students;
import utility.DBConnection;

/**
 *
 * @author Adarsha
 */
class EquipmentPersister {

    private Connection connection;
    private PreparedStatement insertEquipment;
    private PreparedStatement deleteEquipment;
    private PreparedStatement getAllEquipments;
    private PreparedStatement getAllEquipmentsForStudents;
    private PreparedStatement findAllEquipmentsByName;
    private PreparedStatement findAllEquipmentsById;
    private PreparedStatement insertBorrowedEquipments;
    private PreparedStatement updateEquipmentStockAfterBooking;
    private PreparedStatement updateEquipmentStockAfterReturning;

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

                    insertEquipment = connection.prepareStatement("INSERT
INTO equipments (equipment_name, brand, type, availableQuantity, campus)
"
                            + "VALUES(?, ?, ?, ?, ?)");
                    getAllEquipments =
connection.prepareStatement("SELECT * FROM equipments");
```

```
                    getAllEquipmentsForStudents =
connection.prepareStatement("SELECT * FROM equipments WHERE campus = ?");
                    deleteEquipment = connection.prepareStatement("DELETE
FROM equipments WHERE equipment_id = ?");
                    findAllEquipmentsById =
connection.prepareStatement("SELECT * FROM equipments WHERE equipment_id
= ? ");
                    findAllEquipmentsByName =
connection.prepareStatement("SELECT * FROM equipments WHERE
equipment_name LIKE ? ");
                    insertBorrowedEquipments =
connection.prepareStatement("INSERT INTO borrowedequipments(equipment_id,
equipment_name, brand, type,campus, borrower_name, status, borrow_date,
return_date) "
                            + "VALUES(?,?,?,?,?,?,?,?,?)");
                    updateEquipmentStockAfterBooking =
connection.prepareStatement("UPDATE equipments SET availableQuantity =
availableQuantity - 1 WHERE equipment_id = ? ");

                    }
            } catch (SQLException e) {
                System.out.println("Connection Failed!");
                System.out.println("SQLException : " + e.getMessage());
            }

    }

    List<Equipments> getAllEquipmentList() {
        List<Equipments> equipmentsList = new ArrayList<>();
        try {
            ResultSet equipmentResult = getAllEquipments.executeQuery();

            System.out.println("Equipments details reading from the
database.");
            while (equipmentResult.next()) {
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                int equipmentAvailable =
equipmentResult.getInt("availableQuantity");
                String equipmentCampus =
equipmentResult.getString("campus");


                Equipments newEquipment = new
Equipments(equipmentName,equipmentBrand,
equipmentType,equipmentAvailable,equipmentCampus);
                newEquipment.setEquipment_id(equipmentId);
                System.out.println("New added Equipment is :
"+newEquipment);
                equipmentsList.add(newEquipment);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
```

```
        System.out.println("Final Equipment list to be sent from
persister is :"+ equipmentsList);
        return equipmentsList;
    }

    List<Equipments> getAllEquipmentListForStudents() {
        List<Equipments> equipmentsList = new ArrayList<>();
        ActiveUserPresenter ac = new ActiveUserPresenter();
        String activeUser = ac.getMyUsername().get(0).getUsername();
        StudentPresenter s = new StudentPresenter();
        Students std = s.findStudentsByName(activeUser).get(0);

        String stdCampus = std.getCampus();

        try {
            getAllEquipmentsForStudents.setString(1, stdCampus);
            ResultSet equipmentResult =
getAllEquipmentsForStudents.executeQuery();

            System.out.println("Equipments details reading from the
database.");
            while (equipmentResult.next()) {
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                int equipmentAvailable =
equipmentResult.getInt("availableQuantity");
                String equipmentCampus =
equipmentResult.getString("campus");


                Equipments newEquipment = new
Equipments(equipmentName,equipmentBrand,
equipmentType,equipmentAvailable,equipmentCampus);
                newEquipment.setEquipment_id(equipmentId);
                System.out.println("New added Equipment is :
"+newEquipment);
                equipmentsList.add(newEquipment);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        System.out.println("Final Equipment list to be sent from
persister is :"+ equipmentsList);
        return equipmentsList;
    }

    boolean registerEquipment(Equipments equipment) {
        try {
          // System.out.println("At the persister");
            insertEquipment.setString(1, equipment.getEquipment_name());
            insertEquipment.setString(2, equipment.getEquipment_brand());
            insertEquipment.setString(3, equipment.getEquipment_type());
            insertEquipment.setInt(4,
equipment.getAvailableQuantities());
```

```
            insertEquipment.setString(5, equipment.getCampus());

            insertEquipment.executeUpdate();  // execute the prepared
statement insert
                    //   System.out.println("Returning from the
persister");
            return true;

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

    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();
        }
    }

    String deleteEquipment(int equipment_id) {
        String equipmentStatus = "";
        try {

            deleteEquipment.setInt(1, equipment_id);
            int equipmentResult = deleteEquipment.executeUpdate();

            if (equipmentResult > 0) {
                equipmentStatus = "Equipment deleted successfully.";
```

```
            } else {
                equipmentStatus = "Cannot delete the Equipment.";
            }

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

   public List<Equipments> findEquipmentsByName(String keyword) {
       Equipments equipment = new Equipments();
       System.out.println("Here1");
       List<Equipments> searchedEquipments = new ArrayList();
        try {
            findAllEquipmentsByName.setString(1, "%"+keyword+"%");
            System.out.println("Here2");
            ResultSet equipmentResult =
findAllEquipmentsByName.executeQuery();
System.out.println("Here3");
            System.out.println("Equipments details reading from the
database.");
            while (equipmentResult.next()) {
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentCampus =
equipmentResult.getString("campus");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                int equipmentAvailableQuantities =
equipmentResult.getInt("availableQuantity");

System.out.println("Here4");
                equipment = new Equipments(equipmentName,
equipmentBrand,equipmentType, equipmentAvailableQuantities,
equipmentCampus );
                equipment.setEquipment_id(equipmentId);
                System.out.println("Here5");
                searchedEquipments.add(equipment);
                System.out.println("Here6");
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        return searchedEquipments;
    }

   public List<Equipments> findEquipmentsById(int id) {
        Equipments equipment = new Equipments();
       List<Equipments> searchedEquipments = new ArrayList();
        try {
            findAllEquipmentsById.setInt(1, id);
            ResultSet equipmentResult =
findAllEquipmentsById.executeQuery();
```

```
            System.out.println("Equipments details reading from the
database.");
            while (equipmentResult.next()) {
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentCampus =
equipmentResult.getString("campus");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                int equipmentAvailableQuantities =
equipmentResult.getInt("availableQuantity");


                equipment = new Equipments(equipmentName, equipmentBrand,
equipmentType, equipmentAvailableQuantities, equipmentCampus );
                equipment.setEquipment_id(equipmentId);

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

    public Boolean addBorrowedEquipment(BorrowedEquipments equipments) {
        try{
            insertBorrowedEquipments.setInt(1,
equipments.getBorrowedEquipments().getEquipment_id());
            insertBorrowedEquipments.setString(2,
equipments.getBorrowedEquipments().getEquipment_name());
            insertBorrowedEquipments.setString(3,
equipments.getBorrowedEquipments().getEquipment_brand());
            insertBorrowedEquipments.setString(4,
equipments.getBorrowedEquipments().getEquipment_type());
            insertBorrowedEquipments.setString(5,
equipments.getBorrowedEquipments().getCampus());
            insertBorrowedEquipments.setString(6,
equipments.getActiveUser().getUsername());
            insertBorrowedEquipments.setString(7,
equipments.getStatus());
            insertBorrowedEquipments.setDate(8,
getSQLDate(equipments.getBorrowDate()));
            insertBorrowedEquipments.setDate(9,
getSQLDate(equipments.getReturnDate()));

            insertBorrowedEquipments.executeUpdate();


updateEquipmentStockAfterBooking.setInt(1,equipments.getBorrowedEquipment
s().getEquipment_id());
            updateEquipmentStockAfterBooking.executeUpdate();

            return true;
        }catch(SQLException e) {
```

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

        private java.sql.Date getSQLDate(java.util.Date date) {
        java.sql.Date date1 = null;
        if (date != null) {
            date1 = new java.sql.Date(date.getTime());
        }
        return date1;
    }

}
```