```
/*
 * 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.sql.Connection;
import java.sql.Date;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import model.ActiveUser;
import model.BorrowedEquipments;
import model.Equipments;
import model.Students;
import utility.DBConnection;

/**
 *
 * @author Adarsha
 */
public class BorrowedEquipmentPersister {

    private Connection connection;
    private DBConnection utility;
    private PreparedStatement getAllBorrowedEquipmentList;
    private EquipmentPresenter equipmentPresenter;
    private StudentPresenter studentPresenter;
    private Equipments borrowedEquipment;
    private Students borrower;

    private PreparedStatement findBorrowedEquipmentsByName;
    private PreparedStatement findBorrowedEquipmentsByUsername;
    private PreparedStatement returnEquipment;
    private PreparedStatement updateEquipmentStockAfterReturning;
    private PreparedStatement checkOverDue;

    public BorrowedEquipmentPersister() {
        utility = new DBConnection();
        equipmentPresenter = new EquipmentPresenter();
        borrowedEquipment = new Equipments();
        borrower = new Students();
        studentPresenter = new StudentPresenter();


        try {
            this.connection = DBConnection.getConnection(); // database
connection
            if (connection != null) {
                getAllBorrowedEquipmentList =
connection.prepareStatement("SELECT * FROM borrowedequipments WHERE
(status = ? OR status = ?) ORDER BY borrow_date DESC");
```

```
                findBorrowedEquipmentsByName =
connection.prepareStatement("SELECT * FROM borrowedequipments WHERE
equipment_name LIKE ?");
                findBorrowedEquipmentsByUsername =
connection.prepareStatement("SELECT * FROM borrowedequipments WHERE
borrower_name LIKE ?");
                returnEquipment = connection.prepareStatement("UPDATE
borrowedequipments SET status = ?, returned_on_date = ? WHERE borrow_id =
?");
                updateEquipmentStockAfterReturning =
connection.prepareStatement("UPDATE equipments SET availableQuantity =
availableQuantity + 1 WHERE equipment_id = ? ");
                checkOverDue = connection.prepareStatement("UPDATE
borrowedequipments SET status = ? WHERE return_date < NOW() AND (status
=? OR status = ?)");

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


    List<BorrowedEquipments> getAllBorrowedEquipments() {
        List<BorrowedEquipments> equipmentsList = new ArrayList<>();
        try {
            checkOverDue.setString(1, "OverDue");
            checkOverDue.setString(2, "Borrowed");
            checkOverDue.setString(3, "OverDue");
            checkOverDue.executeUpdate();
            getAllBorrowedEquipmentList.setString(1, "Borrowed");
            getAllBorrowedEquipmentList.setString(2, "Overdue");

            ResultSet equipmentResult =
getAllBorrowedEquipmentList.executeQuery();

            System.out.println("Borrowed Equipments details reading from
the database.");
            while (equipmentResult.next()) {
                int borrowId = equipmentResult.getInt("borrow_id");
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                String equipmentCampus =
equipmentResult.getString("campus");
                String equipmentBorrower =
equipmentResult.getString("borrower_name");
                String equipmentStatus =
equipmentResult.getString("status");
                Date borrowDate = equipmentResult.getDate("borrow_date");
                Date returnDate = equipmentResult.getDate("return_date");
```

```
                borrowedEquipment =
equipmentPresenter.findEquipmentsById(equipmentId).get(0);


                //borrower =
studentPresenter.findStudentsByName(equipmentBorrower).get(0);

                ActiveUser actU = new ActiveUser(equipmentBorrower);

                BorrowedEquipments newBorrowedEquipment = new
BorrowedEquipments(borrowedEquipment, actU, borrowDate, returnDate,
equipmentStatus);
                newBorrowedEquipment.setBorrowID(borrowId);
                equipmentsList.add(newBorrowedEquipment);
            }
        } 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<BorrowedEquipments> findEquipmentsByName(String keyword) {
       BorrowedEquipments equipment = new BorrowedEquipments();
       List<BorrowedEquipments> searchedEquipments = new ArrayList();
        try {
            findBorrowedEquipmentsByName.setString(1, "%"+keyword+"%");
            ResultSet equipmentResult =
findBorrowedEquipmentsByName.executeQuery();
            System.out.println("Borrowed Equipments details reading from
the database.");
            while (equipmentResult.next()) {
                 int borrowId = equipmentResult.getInt("borrow_id");
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                Date borrowDate = equipmentResult.getDate("borrow_date");
                Date returnDate = equipmentResult.getDate("return_date");
                String status = equipmentResult.getString("status");
                String borrowerUsername =
equipmentResult.getString("borrower_name");


                borrowedEquipment =
equipmentPresenter.findEquipmentsById(equipmentId).get(0);

                ActiveUser actU = new ActiveUser(borrowerUsername);
                BorrowedEquipments newBorrowedEquipment = new
BorrowedEquipments(borrowedEquipment, actU, borrowDate, returnDate,
status);
                newBorrowedEquipment.setBorrowID(borrowId);
                searchedEquipments.add(newBorrowedEquipment);
            }
        } catch (SQLException e) {
```

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

    List<BorrowedEquipments> findEquipmentsbyBorrowerUsername(String
keyword) {
                BorrowedEquipments equipment = new BorrowedEquipments();
       List<BorrowedEquipments> searchedEquipments = new ArrayList();
        try {
            findBorrowedEquipmentsByUsername.setString(1,
"%"+keyword+"%");
            ResultSet equipmentResult =
findBorrowedEquipmentsByUsername.executeQuery();
            System.out.println("Borrowed Equipments details reading from
the database.");
            while (equipmentResult.next()) {
                 int borrowId = equipmentResult.getInt("borrow_id");
                int equipmentId = equipmentResult.getInt("equipment_id");
                String equipmentName =
equipmentResult.getString("equipment_name");
                String equipmentBrand =
equipmentResult.getString("brand");
                String equipmentType = equipmentResult.getString("type");
                Date borrowDate = equipmentResult.getDate("borrow_date");
                Date returnDate = equipmentResult.getDate("return_date");
                String status = equipmentResult.getString("status");
                String borrowerUsername =
equipmentResult.getString("borrower_name");


                borrowedEquipment =
equipmentPresenter.findEquipmentsById(equipmentId).get(0);

                ActiveUser actU = new ActiveUser(borrowerUsername);
                BorrowedEquipments newBorrowedEquipment = new
BorrowedEquipments(borrowedEquipment, actU, borrowDate, returnDate,
status);
                newBorrowedEquipment.setBorrowID(borrowId);
                searchedEquipments.add(newBorrowedEquipment);
            }
        } catch (SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
        }
        return searchedEquipments;
    }

     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;
     }

    boolean returnEquipment(BorrowedEquipments selectedEquipment) {
        try{
            int borrower_id = selectedEquipment.getBorrowID();
```

```
            int equipment_id = selectedEquipment.getEquipmentID();
            String status = selectedEquipment.getStatus();

            System.out.println("borrowID is :"+ borrower_id+" equipment
ID is "+equipment_id+" status is : "+status);

            returnEquipment.setString(1, status);
            returnEquipment.setDate(2,
getSQLDate(selectedEquipment.getReturned_onDate()));
            returnEquipment.setInt(3, borrower_id);
            int result = returnEquipment.executeUpdate();

            updateEquipmentStockAfterReturning.setInt(1, equipment_id);
            updateEquipmentStockAfterReturning.executeUpdate();

            if(result>0){
                System.out.println("The equipment has been returned
successfully");
                return true;
            }
        }catch(SQLException e) {
            System.out.println("SQL Exception: " + e.getMessage());
            return false;
        }

      return true;


    }
}
```