MỤC TIÊU:

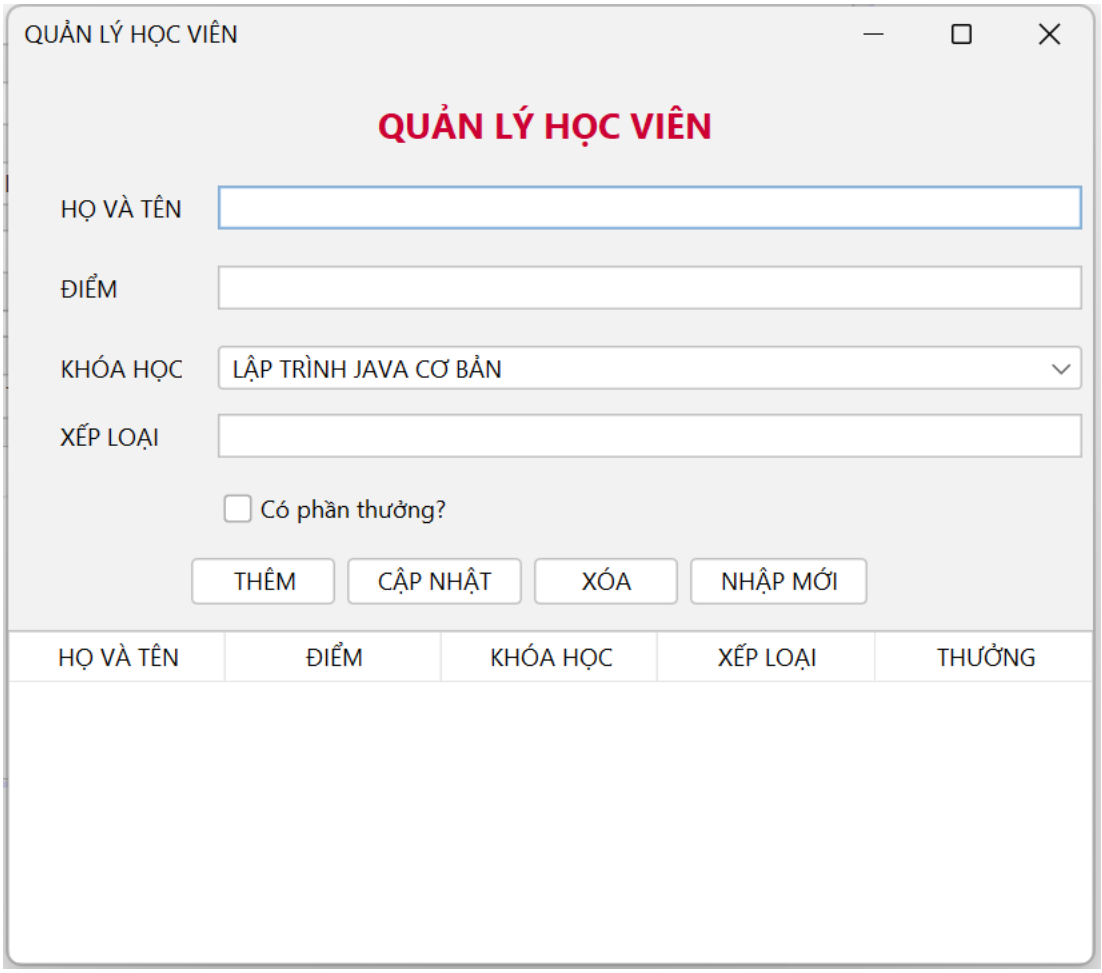
Kết thúc bài thực hành này bạn có khả năng:

- ✓ Hiểu được các component: JCombobox, JTable, JprogressBar, JSlider
- ✓ Xử lý sự kiện

PHẦN I

BÀI 1 (2 ĐIỂM)

Thiết kế giao diện như hình sau:



- ✓ Đặt tên theo quy ước cho các thành phần giao diện trên form

- ✓ Không cho phép nhập vào ô "Xếp loại"
- ✓ Viết mã để:
  - Đưa cửa sổ hiển thị giữa màn hình
  - Click nút "NHẬP MỚI" xóa trắng các ô trên form và bỏ chọn check có phần thưởng

BÀI 2 (4 ĐIỂM)

1. Tạo lớp Student để quản lý thông tin học viên như sau:

```
public class Student {
    public String name;
    public double marks;
    public String course;

    public Student() {
    }

    public String getGrade() {
        if (this.marks < 3) {
            return "Kém";
        }
        if (this.marks < 5) {
            return "Yếu";
        }
        if (this.marks < 6.5) {
            return "Trung bình";
        }
        if (this.marks < 7.5) {
            return "Khá";
        }
        if (this.marks < 9) {
            return "Giỏi";
        }
        return "Xuất sắc";
    }
    public boolean isBonus() {
        return this.marks >= 7.5;
    }
}
```

2. Sử dụng giao diện bài 1 và khai báo vào lớp Jframe các thuộc tính và phương thức sau

- ✓ Trường list để chứa danh sách nhập vào

```
//Nắm giữ danh sách học viên nhập từ người dùng
List<Student> list = new ArrayList<>();
```

- ✓ Các phương thức xử lý theo thao tác người dùng

public void **addStudent**(){}

public void **removeStudent**(){}

public void **updateStudent**(){}

public void **fillToTable**(){}

public void **showDetail**(){}

3. Viết mã cho nút "THÊM" cho phép tạo học viên và bổ sung vào List<Student> theo hướng dẫn sau:

- ✓ Xử lý sự kiện click nút "**THÊM**"

```
private void btnThemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.addStudent(); //Thêm sinh viên vào List<Student>
    this.fillToTable(); //Hiển thị List<Student> lên bảng
}
```

- ✓ Viết mã cho phương thức addStudent()

```
/*
* Tạo học viên với thông tin nhập từ form
*/
Student sv = new Student();
sv.setName(name: txtHoTen.getText());
sv.setMarks(marks: Double.parseDouble(s: txtDiem.getText()));
sv.setCourse((String) cboKhoaHoc.getSelectedItem());
//bổ sung học viên vào List<Student>
list.add(e: sv);
/*
* Hiển thị xếp loại và thưởng
*/
txtXepLoai.setText(t: sv.getGrade());
chkThuong.setSelected(b: sv.isBonus());
```

✓ Viết mã cho phương thức fillToTable()

```
/*
* Tạo mới DefaultTableModel
*/
String[]headers = {"HỌ VÀ TÊN","ĐIỂM", "KHÓA HỌC", "XẾP LOẠI", "THƯỞNG"};
DefaultTableModel model = new DefaultTableModel(columnNames: headers, rowCount: 0);
/*
* Duyệt List<Student> và bổ sung vào bảng
*/
for(Student st:list){
    Object[]row = new Object[]{st.getName(), st.getMarks(),st.getCourse(),
        st.getGrade(), st.isBonus()?"THƯỞNG":"KHÔNG"};
    model.addRow(rowData:row);
}
tblHocViens.setModel(dataModel: model);
```

4. Viết mã xử lý sự kiện click vào 1 dòng trên bảng thì hiển thị chi tiết thông tin của sinh viên được chọn lên form

✓ Xử lý sự kiện click chuột vào bảng

```
private void tblHocViensMouseClicked(java.awt.event.MouseEvent evt)
    // TODO add your handling code here:
    this.showDetail();
}
```

✓ Viết mã cho phương thức showDetail()

```
/*
* Lấy sinh viên tại vị trí được chọn trên bảng
*/
int index = tblHocViens.getSelectedRow();
Student st = list.get(index);
/*
* Cập nhật lại thông tin trên form
*/
txtHoTen.setText(t: st.getName());
txtDiem.setText(t: String.valueOf(d: st.getMarks()));
txtXepLoai.setText(t: st.getGrade());
cboKhoaHoc.setSelectedItem(anObject: st.getCourse());
chkThuong.setSelected(b: st.isBonus());
```

5. Xử lý sự kiện click "XÓA"

✓ Mã xử lý sự kiện xóa

```
private void btnXoaActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.removeStudent();//Xóa sinh viên khỏi List<Student>
    this.fillToTable();//Hiển thị lại bảng
}
```

✓ Mã của phương thức removeStudent()

```
/*
* Xóa sinh viên tại vị trí được chọn trên bảng
*/
int index = tblHocViens.getSelectedRow();
list.remove(index);
```

6. Xử lý sự kiện click "CẬP NHẬT"

✓ Xử lý sự kiện cập nhật

```
private void btnCapNhatActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.updateStudent();//Cập nhật thông tin sinh viên
    this.fillToTable();//Hiển thị lại bảng
}
```

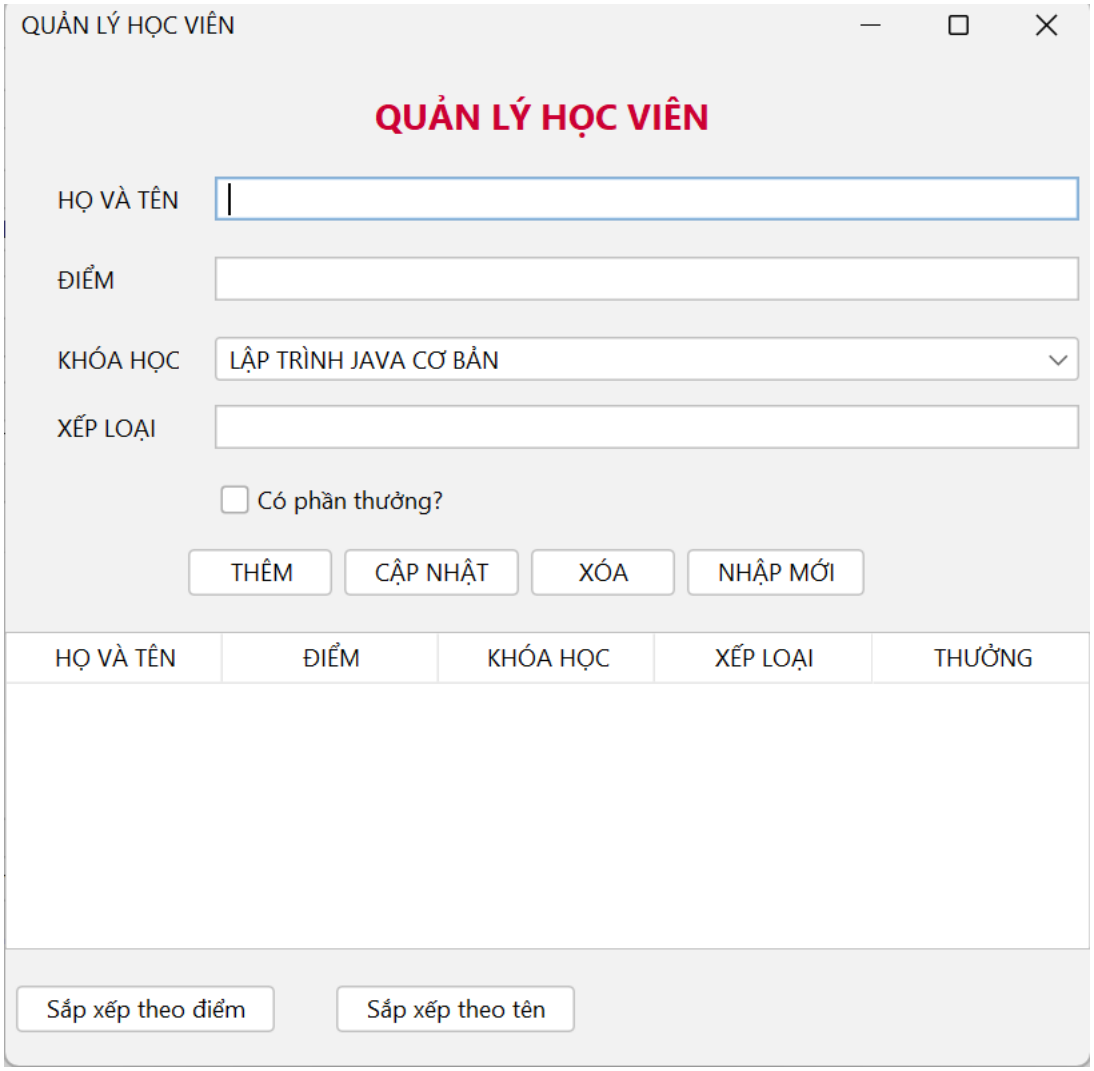
✓ Mã của phương thức updateStudent()

```
int index = tblHocViens.getSelectedRow();
Student st = list.get(index);
st.setName(name: txtHoTen.getText());
st.setCourse(course:cboKhoaHoc.getSelectedItem().toString());
st.setMarks(marks: Double.parseDouble(s: txtDiem.getText()));
txtXepLoai.setText(t: st.getGrade());
chkThuong.setSelected(b: st.isBonus());
```

PHẦN II

BÀI 3 (2 ĐIỂM)

1. Bổ sung 2 nút để sắp xếp danh sách có giao diện như sau:



   - ✓ Đặt tên nút hợp lệ theo quy ước
   - ✓ Bổ sung 2 phương thức sau vào JFrame
     public void orderByName(){}
     public void orderByMarks(){}

2. Viết mã cho nút "Sắp xếp theo tên"

- ✓ Mã xử lý sự kiện click "Sắp xếp theo tên"

```
private void btnSXTheoTenActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.orderByName();//gọi phương thức sắp xếp theo tên
    this.fillToTable();//Hiển thị lại bảng
}
```

- ✓ Mã cho phương thức orderByName()

```
Comparator<Student> com = new Comparator<Student>() {
    @Override
    public int compare(Student o1, Student o2) {

        return o1.getName().compareTo(anotherString:o2.getName());
    }
};
Collections.sort(list,c:com);
```

3. Viết mã cho nút "Sắp xếp theo điểm"

- ✓ Mã sự kiện click "Sắp xếp theo điểm"

```
private void btnSXTheoDiemActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.orderByMarks();//gọi phương thức sắp xếp theo điểm
    this.fillToTable();//Hiển thị lại bảng
}
```

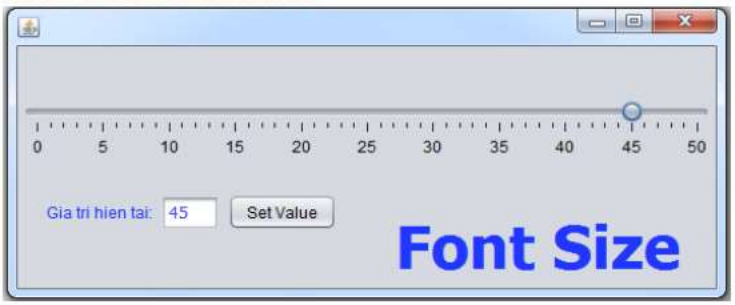
- ✓ Mã cho phương thức orderByMarks ()

```
Comparator<Student> com = new Comparator<Student>() {
    @Override
    public int compare(Student o1, Student o2) {
        Double d1 = o1.getMarks();
        Double d2 = o2.getMarks();
        return d1.compareTo(anotherDouble:d2);
    }
};
Collections.sort(list,c:com);
```

BÀI 4: XÂY DỰNG ỨNG DỤNG NHƯ HÌNH (1 ĐIỂM)

Yêu cầu:

- Khi kéo JSlider tăng hoặc giảm, giá trị của textbox và "Font Size" thay đổi theo
- Người dùng chọn nút "Set Value" thì giá trị của JSlider và "Font Size" thay đổi theo giá trị trong textbox.




```
private void jSlider1StateChanged(javax.swing.event.ChangeEvent evt) {
    // TODO add your handling code here:
    int value = jSlider1.getValue();
    //lblValue.setText(String.valueOf(value));
    txtValue.setText(String.valueOf(value));
    lblFont.setFont(new java.awt.Font("Tahoma", 1, value));
}

private void formWindowActivated(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
    int value = jSlider1.getValue();
    //lblValue.setText(String.valueOf(value));
    txtValue.setText(String.valueOf(value));
    lblFont.setFont(new java.awt.Font("Tahoma", 1, value));
}

private void btnSetValueActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    int value = Integer.parseInt(txtValue.getText());
    jSlider1.setValue(value);
}
```

BÀI 5: GIẢNG VIÊN CHO THÊM (1 ĐIỂM)

***** YÊU CẦU NỘP BÀI:**

Sv nén file bao gồm các yêu cầu đã thực hiện trên, nộp lms đúng thời gian quy định của giảng viên. Không nộp bài coi như không có điểm.

--- Hết ---