

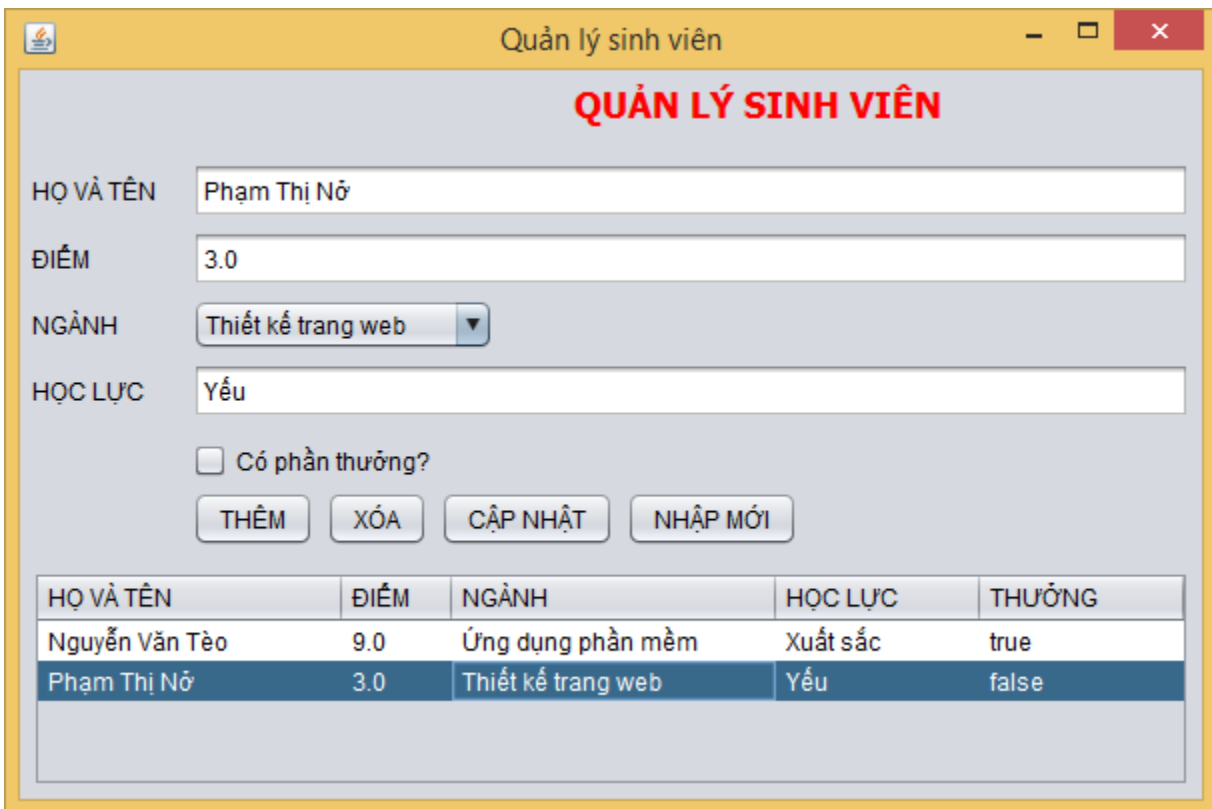
LAB 3: COLLECTION & MAP

MỤC TIÊU:

Kết thúc bài thực hành này bạn có khả năng

- ✓ Sử dụng List và ArrayList
- ✓ Sử dụng JTable, JComboBox
- ✓ Sử dụng lớp tiện ích Collections
- ✓ Sử dụng Map và HashMap

Cụ thể sau bài lab này bạn có thể xây dựng một ứng dụng nhỏ để quản lý sinh viên với giao diện như sau:

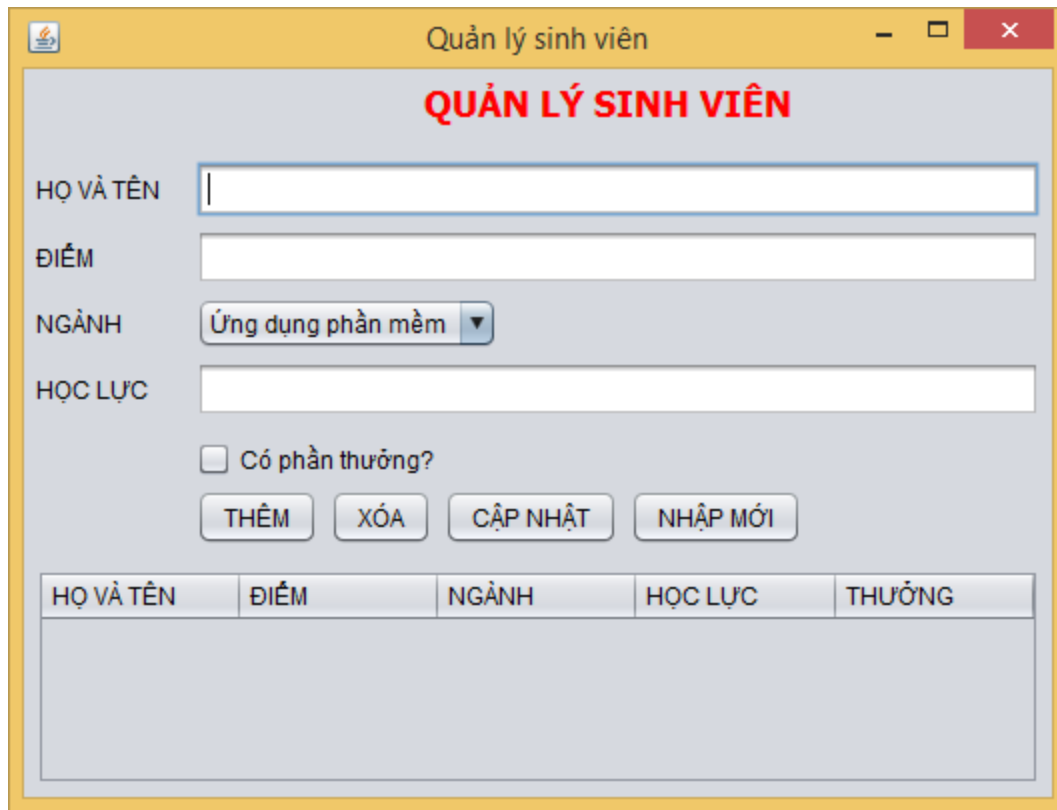


HỌ VÀ TÊN	ĐIỂM	NGÀNH	HỌC LỰC	THƯỞNG
Nguyễn Văn Tèo	9.0	Ứng dụng phần mềm	Xuất sắc	true
Phạm Thị Nở	3.0	Thiết kế trang web	Yếu	false

PHẦN I

BÀI 1 (2 ĐIỂM)

Thiết kế giao diện như hình sau



- ✓ Đặt tên theo qui ước cho các thành phần giao diện trên form
- ✓ Không cho phép nhập vào ô học lực
- ✓ Viết mã để
 - Đưa cửa sổ hiển thị giữa màn hình
 - Click nút [NHẬP MỚI] sẽ xóa trắng các ô nhập trên form và bỏ chọn CheckBox [Có phần thưởng]

BÀI 2 (4 ĐIỂM)

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

```
public class Student {
    public String name;
    public double marks;
    public String major;
    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 của bài 1 và khai báo vào lớp JFrame các trường và phương thức sau

- ✓ **trường list** để chứa danh sách sinh viên nhập vào.

```

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

```

- ✓ Các phương thức xử lý theo 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 sinh 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 sinh viên với thông tin nhập từ form
 */
Student sv = new Student();
sv.name = txtHoTen.getText();
sv.marks = Double.parseDouble(txtDiem.getText());
sv.major = (String) cboNganh.getSelectedItem();
// Bổ sung sinh viên vào List<Student>
list.add(sv);
/*
 * Hiển thị học lực và thưởng
 */
txtHocLuc.setText(sv.getGrade());
chkThuong.setSelected(sv.isBonus());
```

- ✓ Viết mã cho phương thức fillToTable() như sau

```

/*
 * Lấy mô hình dữ liệu của bảng và xóa sách các hàng
 */
DefaultTableModel model = (DefaultTableModel) tblStudents.getModel();
model.setRowCount(0);
/*
 * Duyệt List<Student> và bổ sung các sinh viên vào bảng
 */
for(Student sv : list){
    Object[] row = new Object[]{sv.name, sv.marks,
        sv.major, sv.getGrade(), sv.isBonus()};
    model.addRow(row);
}

```

4. Viết mã xử lý sự kiện click vào 1 hàng 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 tblStudentsMouseClicked(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 = tblStudents.getSelectedRow();
Student sv = list.get(index);
/*
 * Cập nhật lại thông tin mới từ form
 */
sv.name = txtHoTen.getText();
sv.marks = Double.parseDouble(txtDiem.getText());
sv.major = (String) cboNganh.getSelectedItem();
txtHocLuc.setText(sv.getGrade());
chkThuong.setSelected(sv.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 = tblStudents.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
}
```

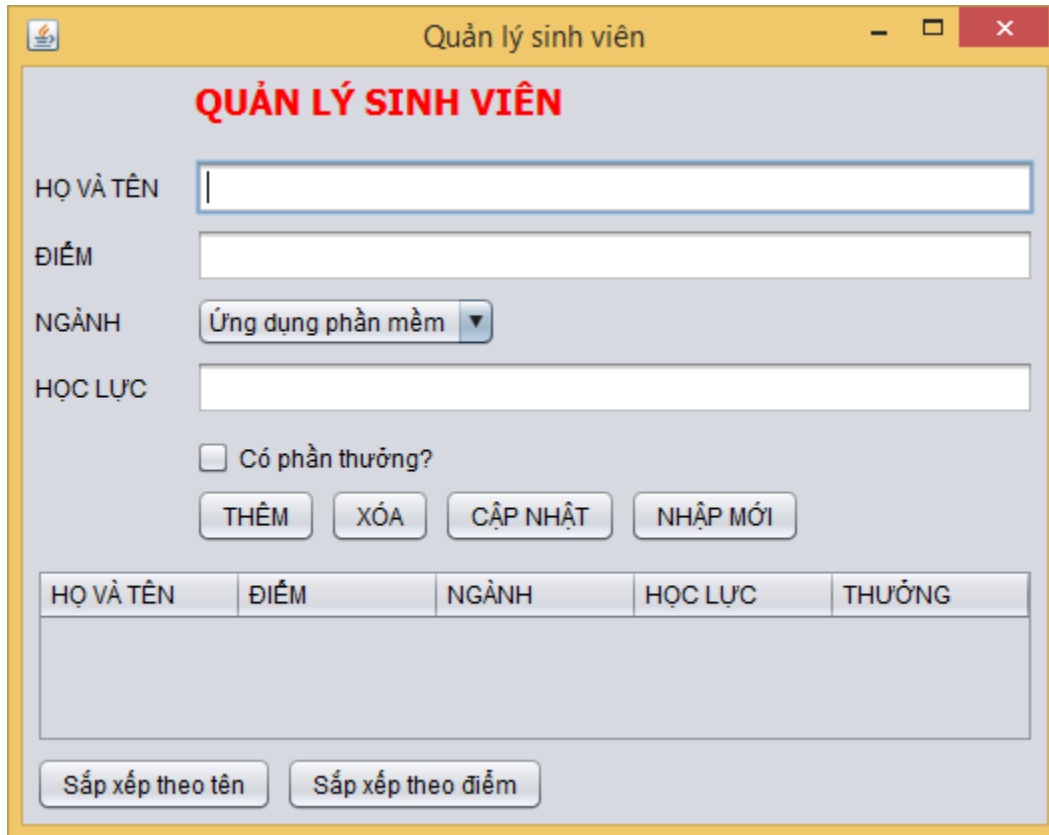
- ✓ Mã của phương thức updateStudent()

```
/*
 * Lấy sinh viên tại vị trí được chọn trên bảng
 */
int index = tblStudents.getSelectedRow();
Student sv = list.get(index);
/*
 * Cập nhật lại thông tin mới từ form
 */
sv.name = txtHoTen.getText();
sv.marks = Double.parseDouble(txtDiem.getText());
sv.major = (String) cboNganh.getSelectedItem();
txtHocLuc.setText(sv.getGrade());
chkThuong.setSelected(sv.isBonus());
```

PHẦN II

BÀI 3 (2 ĐIỂM)

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



- ✓ Đặt tên nút hợp lệ theo qui 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 btnSXTTheoTenActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    this.orderByName(); // gọi phương thức sắp xếp theo tên
    this.fillToTable();
}

```

- ✓ Mã cho phương thức orderByName(){}


```
public void orderByName(){}

```

```

Comparator<Student> com = new Comparator<Student>() {
    @Override
    public int compare(Student o1, Student o2) {
        return o1.name.compareTo(o2.name);
    }
};
Collections.sort(list, com);

```

3. Viết mã cho nút [Sắp xếp theo tên]

- ✓ Mã xử lý 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();
}

```

- ✓ Mã cho phương thức orderByMarks(){}

```

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

```

BÀI 4 (2 ĐIỂM)

Tạo lớp MapDemo chứa phương thức main(). Viết mã cho phương thức main() theo hướng dẫn sau:

- ✓ Tạo Map<String, Student> để chứa các cặp tên và đối tượng Student.
 Map<String, Student> map = new HashMap<>();
- ✓ Bổ sung vào map 3 sinh viên. Mã sau bổ sung một sinh viên vào map
 Student sv1 = new Student();
 sv1.name = "Tuấn";


```
sv1.major = "CNTT";
```

```
sv1.marks = 7.0;
```

```
map.put(sv1.name, sv1);
```

- ✓ Duyệt Map và xuất thông tin của sinh viên ra màn hình console

```
Set<String> keys = map.keySet();
```

```
for(String name : keys){
```

```
    Student sv = map.get(name);
```

```
    System.out.println(">Họ và tên: " + sv.name);
```

```
    System.out.println(">Học lực: " + sv.getGrade());
```

```
}
```