

## PHIẾU BÀI TẬP SỐ 4

### A. MỤC ĐÍCH

- Sinh viên biết triển khai một dự án thực tế đơn giản với Java
- Sinh viên biết kết hợp các kiến thức khác nhau của Java
- Mở rộng thêm kiến thức.

### B. NỘI DUNG

#### Bài 1: Quản lý sản phẩm

Trong chương trình sinh viên biết kết hợp một số chức năng:

- Các control : JMenuBar, JSplitPane, JList, JTable, JCombobox
- Các collections: ArrayList, Vector
- JFileChooser
- Cho phép lưu đối tượng xuống ổ cứng và đọc đối tượng lên giao diện

#### Mô tả:

Cho phép nhập xuất danh sách các danh mục sản phẩm, các sản phẩm của từng danh mục, các chức năng thêm sửa xóa, lưu tập tin

#### Cấu trúc file chương trình gồm có:

Product.java : dùng để lưu thông tin của từng sản phẩm

Category.java: dùng để lưu danh mục sản phẩm và lưu danh sách các sản phẩm của từng danh mục

ListCategory.java: dùng để lưu danh sách các danh mục

MyProcessFile.java: dùng để xử lý tập tin: lưu và đọc đối tượng trên ổ cứng

MainManagerUI.java: lớp giao diện chính của chương trình

CategoryManagerUI.java: lớp giao diện phụ để cập nhật thông tin của danh mục

TestMain.java: dùng để chạy chương trình chính

#### Giao diện chính của chương trình như sau:

Quản lý sản phẩm!

## Quản lý sản phẩm

Danh mục sản phẩm

- Mặt hàng điện tử
- Mặt hàng hóa chất
- Mặt hàng gia dụng

Thông tin chi tiết

Product ID	Product Name	UnitPrice	Quantity	Description
p1	đèn huỳnh quang	888.0	1	đèn huỳnh qua...
p343	Máy nổ 113	134.0	4	Máy nổ nổ nổ

Category : Mặt hàng điện tử

Product ID:

Product Name:

Unit Price:

Quantity:

Description:

New Update Remove

New Save Remove

Khi bấm vào nút New của danh mục sẽ hiển thị màn hình cập nhật danh mục:

Cate information

Cate Id:

Cate Name:

OK

### Coding mẫu:

#### Product.java

```
import java.io.Serializable; public class Product implements Serializable {
    private static final long serialVersionUID = 1L;
    private String productId;
    private String productName;
    private String description;
```

```
private double unitPrice;

private int quantity;

public String getProductId() {
    return productId;
}

public void setProductId(String productId) {
    this.productId = productId;
}

public String getProductName() {
    return productName;
}

public void setProductName(String productName) {
    this.productName = productName;
}

public String getDescription() {
    return description;
}

public void setDescription(String description) {
    this.description = description;
}

public double getUnitPrice() {
    return unitPrice;
}

public void setUnitPrice(double unitPrice) {
    this.unitPrice = unitPrice;
}

public int getQuantity() {
    return quantity;
}
```

```

}

public void setQuantity(int quantity) {
    this.quantity = quantity;
}

public Product(String productId, String productName,
String description, double unitPrice, int quantity) {
    super();
    this.productId = productId;
    this.productName = productName;
    this.description = description;
    this.unitPrice = unitPrice;
    this.quantity = quantity;
}

public Product() {
    super();
}
}

```

#### Category.java:

```

import java.io.Serializable;
import java.util.ArrayList;

public class Category implements Serializable {
    private static final long serialVersionUID = 1L;
    private String catId;
    private String cateName;
    private ArrayList<Product> listPro = new ArrayList<Product>();
    public String getCatId() {
        return catId;
    }
}

```

```
public void setCateId(String cateId) {  
    this.cateId = cateId;  
}  
  
public String getCateName(){  
    return cateName;  
}  
  
public void setCateName(String cateName) {  
    this.cateName = cateName;  
}  
  
public Category(String cateId, String cateName) {  
    super();  
    this.cateId = cateId;  
    this.cateName = cateName;  
}  
  
public Category() {  
    super();  
}  
  
public Product findProductById(String id)  
{  
    for(Product p: listPro)  
    if(p.getProductId().equalsIgnoreCase(id))  
        return p;  
    return null;  
}  
  
public boolean addProduct(Product p)  
{  
    Product pFind=findProductById(p.getId());  
    if(pFind!=null)
```

```
{  
    System.err.println("Duplicate product ID!");  
    return false;  
}  
listPro.add(p);  
return true;  
}  
public ArrayList<Product>getListPro() {  
    return listPro;  
}  
public void setListPro(ArrayList<Product> listPro) {  
    this.listPro = listPro;  
}  
public void removeProductById(String id)  
{  
    Product pFind=findProductById(id);  
    if(pFind!=null)  
        listPro.remove(pFind);  
}  
public void removeProductByIndex(String index)  
{  
    listPro.remove(index);  
}  
public int numberOfProduct()  
{  
    return listPro.size();  
}  
public String toString() {
```

```
    return this.cateName;
}
}
```

ListCategory.java:

```
import java.io.Serializable;
import java.util.ArrayList;

public class ListCategory implements Serializable {
    private static final long serialVersionUID = 1L;
    private ArrayList<Category> listCate=new ArrayList<Category>();

    public Category findCateById(String id)
    {
        for(Category cate: listCate)
        {
            if(cate.getCateId().equalsIgnoreCase(id))
            return cate;
        }
        return null;
    }

    public void addCate(Category cate)
    {
        Category cateFind= findCateById(cate.getCateId());
        if(cateFind!=null)
            cateFind=cate;
        else
            listCate.add(cate);
    }

    public void removeCateById(String id)
    {

```

```
Category cateFind= findCateById(id);  
if(cateFind!=null)  
listCate.remove(cateFind);  
  
}  
public ArrayList<Category>getList()  
{  
return listCate;  
}  
}
```

#### **MyProcessFile.java:**

```
import java.io.FileInputStream;  
import java.io.FileOutputStream;  
import java.io.ObjectInputStream;  
import java.io.ObjectOutputStream;  
public class MyProcessFile {  
public static void saveData(Object list,String fileName)  
{  
try  
{  
FileOutputStream fOut=new FileOutputStream(fileName);  
ObjectOutputStream oOut=new ObjectOutputStream(fOut);  
oOut.writeObject(list);  
}  
catch(Exception ex)  
{  
ex.printStackTrace();  
}  
}
```



```
public static Object openData(String fileName)
{
try
{
    FileInputStream fln=new FileInputStream(fileName);
    ObjectInputStream oIn=new ObjectInputStream(fln);
    return oIn.readObject();
}
catch(Exception ex)
{
    ex.printStackTrace();
}
return null;
}
}
```

MainManagerUI.java:

```
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.Vector;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComboBox;
```

```
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.JTable;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.border.TitledBorder;
import javax.swing.event.ListSelectionEvent;
import javax.swing.event.ListSelectionListener;
import javax.swing.table.DefaultTableModel;
public class MainManagerUI extends JFrame {
    private static final long serialVersionUID = 1L;
    private static JList lstCate;
    private JTable tblProduct;
    private DefaultTableModel dtmProduct;
    private JButton btnCateRemove, btnCateNew, btnCateUpdate, btnNew, btnSave, btnRemove;
    private JTextField txtId, txtName, txtUnitprice, txtQuantity;
    private JTextArea txtDescription;
    private static JComboBox cboCateList;
    JMenuBar menubar;
```

```
JMenu mnuFile;

JMenuItem mnuFileOpenDataFromDisk,mnuFileWritetodisk,mnuFileExit;

public static ListCategory listData;

public static Category selectedCate;

public MainManagerUI(String title)
{
    super(title);
    listData=new ListCategory();
}

public void doShow()
{
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setSize(800, 550);
    addControl();
    setLocationRelativeTo(null);
    setVisible(true);
}

public void addMenu()
{
    menubar=new JMenuBar();
    mnuFile=new JMenu("File");
    mnuFileWritetodisk=new JMenuItem("Write Data to disk");
    mnuFileOpenDataFromDisk=new JMenuItem("Open Data from disk");
    mnuFileExit=new JMenuItem("Exit");
    menubar.add(mnuFile);
    mnuFile.add(mnuFileWritetodisk);
    mnuFile.add(mnuFileOpenDataFromDisk);
    mnuFile.addSeparator();
```

```
mnuFile.add(mnuFileExit);  
setJMenuBar(menuBar);  
}  
public void addControl()  
{  
    addMenu();  
    JPanel pnBorder=new JPanel();  
    pnBorder.setLayout(new BorderLayout());  
    JPanel pnNorth=new JPanel();  
    JLabel lblTitle=new JLabel("Quản lý sản phẩm");  
    Font ftTitle=new Font("arial", Font.BOLD, 32);  
    lblTitle.setFont(ftTitle);  
    lblTitle.setForeground(Color.BLUE);  
    pnNorth.add(lblTitle);  
    pnBorder.add(pnNorth, BorderLayout.NORTH);  
    JPanel pnListCate=new JPanel();  
    JPanel pnListProduct=new JPanel();  
    JSplitPane slitPane=new JSplitPane(JSplitPane.HORIZONTAL_SPLIT, pnListCate, pnListProduct);  
    pnBorder.add(slitPane, BorderLayout.CENTER);  
    pnListCate.setLayout(new BorderLayout());  
    lstCate=new JList();  
    TitledBorder cateborder=new TitledBorder(BorderFactory.createLineBorder(Color.RED), "Danh  
    mục sản phẩm");  
    lstCate.setBorder(cateborder);  
    pnListCate.setPreferredSize(new Dimension(300, 0));  
    pnListCate.add(lstCate, BorderLayout.CENTER);  
    JPanel pnListCateSouth=new JPanel();  
    btnCateNew =new JButton("New");
```

```
pnListCateSouth.add(btnCateNew);
btnCateUpdate =new JButton("Update");
pnListCateSouth.add(btnCateUpdate);
btnCateRemove =new JButton("Remove");
pnListCateSouth.add(btnCateRemove);
pnListCate.add(pnListCateSouth,BorderLayout.SOUTH);
pnListProduct.setLayout(new BorderLayout());
JPanel pnProductTitle=new JPanel();
JLabel lblProductTitle=new JLabel("Thông tin chi tiết");
pnProductTitle.add(lblProductTitle);
pnListProduct.add(pnProductTitle,BorderLayout.NORTH);
JPanel pnProductTable=new JPanel();
pnProductTable.setLayout(new BorderLayout());
pnListProduct.add(pnProductTable,BorderLayout.CENTER);
dtmProduct =new DefaultTableModel();
dtmProduct.addColumn("Product ID");
dtmProduct.addColumn("Product Name");
dtmProduct.addColumn("UnitPrice");
dtmProduct.addColumn("Quantity");
dtmProduct.addColumn("Description");
tblProduct=new JTable(dtmProduct);
JScrollPane sctblproduct=new JScrollPane(tblProduct);
pnProductTable.add(sctblproduct,BorderLayout.CENTER);
JPanel pnProductDetail=new JPanel();
pnListProduct.add(pnProductDetail,BorderLayout.SOUTH);
pnProductDetail.setLayout(new BoxLayout(pnProductDetail, BoxLayout.Y_AXIS));
JPanel pnCateList=new JPanel();
JLabel lblCateId=new JLabel("Category :");
```

```
cboCateList=new JComboBox();
pnCateList.add(lblCateId);
pnCateList.add(cboCateList);
pnProductDetail.add(pnCateList);
JPanel pnProductId=new JPanel();
JLabel lblProId=new JLabel("Product ID:");
txtId=new JTextField(20);
pnProductId.add(lblProId);
pnProductId.add(txtId);
pnProductDetail.add(pnProductId);
JPanel pnProductName=new JPanel();
JLabel lblProName=new JLabel("Product Name:");
txtName=new JTextField(20);
pnProductName.add(lblProName);
pnProductName.add(txtName);
pnProductDetail.add(pnProductName);
JPanel pnProductUnitPrice=new JPanel();
JLabel lblUnitPrice=new JLabel("Unit Price:");
txtUnitprice=new JTextField(20);
pnProductUnitPrice.add(lblUnitPrice);
pnProductUnitPrice.add(txtUnitprice);
pnProductDetail.add(pnProductUnitPrice);
JPanel pnProductQuantity=new JPanel();
JLabel lblQuantity=new JLabel("Quantity:");
txtQuantity=new JTextField(20);
pnProductQuantity.add(lblQuantity);
pnProductQuantity.add(txtQuantity);
pnProductDetail.add(pnProductQuantity);
```

```
JPanel pnProductDescription=new JPanel();
JLabel lblDescription=new JLabel("Description:");
txtDescription=new JTextArea(4, 20);
JScrollPane scare=new JScrollPane(txtDescription);
pnProductDescription.add(lblDescription);
pnProductDescription.add(scare);
pnProductDetail.add(pnProductDescription);

JPanel pnButton=new JPanel();
btnNew=new JButton("New");
btnSave=new JButton("Save");
btnRemove=new JButton("Remove");
pnButton.add(btnNew);
pnButton.add(btnSave);
pnButton.add(btnRemove);
pnProductDetail.add(pnButton);

cboCateList.setPreferredSize(txtId.getPreferredSize());
lblCatId.setPreferredSize(lblProName.getPreferredSize());
lblDescription.setPreferredSize(lblProName.getPreferredSize());
lblQuantity.setPreferredSize(lblProName.getPreferredSize());
lblUnitPrice.setPreferredSize(lblProName.getPreferredSize());
lblProid.setPreferredSize(lblProName.getPreferredSize());

Container con=getContentPane();
con.add(pnBorder);

btnCateNew.addActionListener(new processButtonEvent());
btnNew.addActionListener(new processButtonEvent());
btnSave.addActionListener(new processButtonEvent());
btnRemove.addActionListener(new processButtonEvent());
cboCateList.addActionListener(new processButtonEvent());
```

```

mnuFileWritetodisk.addActionListener(new processButtonEvent());
mnuFileOpenDataFromDisk.addActionListener(new processButtonEvent());
lstCate.addListSelectionListener(new ListSelectionListener() {

@Override

public void valueChanged(ListSelectionEvent arg0) {
    selectedCate=(Category) lstCate.getSelectedValue();
    showListProductIntoTable();
}

});
}

private void showListProductIntoTable()
{
    dtmProduct.setRowCount(0);
    for(Product p: selectedCate.getListPro())
    {
        Vector<String>vec=new Vector<String>();
        vec.add(p.getProductId());
        vec.add(p.getProductName());
        vec.add(p.getUnitPrice()+"");
        vec.add(p.getQuantity()+"");
        vec.add(p.getDescription());
        dtmProduct.addRow(vec);
    }
}

public static void updateCateList()
{
    lstCate.removeAll();
    lstCate.setListData(listData.getList().toArray());
}

```



```
lstCate.updateUI();
cboCateList.removeAllItems();
for(Category cate : listData.getList())
{
    cboCateList.addItem(cate);
}
}

private void doCreateNewCate()
{
    CategoryManagerUI cateUI=new CategoryManagerUI("Cate information");
    cateUI.doShow();
}

private void doSaveProduct()
{
    if(selectedCate!=null)
    {
        Product p=new Product();
        p.setProductId(txtId.getText());
        p.setProductName(txtName.getText());
        p.setQuantity(Integer.parseInt(txtQuantity.getText()));
        p.setUnitPrice(Double.parseDouble(txtUnitprice.getText()));
        p.setDescription(txtDescription.getText());
        selectedCate.addProduct(p);
    }
}

private void doComboboxSelected()
{
    selectedCate=(Category) cboCateList.getSelectedItem();
```

```

}

private void doWriteDataToDisk()
{
    JFileChooser fc=new JFileChooser(".");
    if(fc.showSaveDialog(null)==JFileChooser.APPROVE_OPTION)
    {
        MyProcessFile.saveData(listData, fc.getSelectedFile().getAbsolutePath());
    }
}

private void doReadDataFromDisk()
{
    JFileChooser fc=new JFileChooser(".");
    if(fc.showOpenDialog(null)==JFileChooser.APPROVE_OPTION)
    {
        listData=(ListCategory) MyProcessFile.openData(fc.getSelectedFile().getAbsolutePath());
        updateCateList();
    }
}

private class processButtonEvent implements ActionListener
{
    @Override
    public void actionPerformed(ActionEvent e) {
        Object o=e.getSource();
        if(o.equals(btnCateNew))
        {
            doCreateNewCate();
        }
        else if(o.equals(btnNew))

```

```
{
txtDescription.setText("");
txtId.setText("");
txtName.setText("");
txtUnitprice.setText("");
txtQuantity.setText("");
txtId.requestFocus();
}
else if(o.equals(btnSave))
{
doSaveProduct();
}
else if(o.equals(cboCateList))
{
doComboboxSelected();
}
else if(o.equals(mnuFileWritetodisk))
{
doWriteDataToDisk();
}
else if(o.equals(mnuFileOpenDataFromDisk))
{
doReadDataFromDisk();
}
}
}
}
```

CategoryManagerUI.java:

```
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;
public class CategoryManagerUI extends JFrame{
    private static final long serialVersionUID = 1L;
    private JTextField txtId,txtName;
    private JButton btnOk;
    public CategoryManagerUI(String title)
    {
        setTitle(title);
    }
    public void doShow()
    {
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setSize(300, 150);
        addControl();
        setLocationRelativeTo(null);
        setAlwaysOnTop(true);
        setVisible(true);
    }
    public void addControl()
    {
```

```
JPanel pnBox=new JPanel();
pnBox.setLayout(new BoxLayout(pnBox, BoxLayout.Y_AXIS));
JPanel pnId=new JPanel();
txtId=new JTextField(15);
txtName=new JTextField(15);
JLabel lblId=new JLabel("Cate Id:");
JLabel lblName=new JLabel("Cate Name:");
pnId.add(lblId);
pnId.add(txtId);
pnBox.add(pnId);
JPanel pnName=new JPanel();
pnName.add(lblName);
pnName.add(txtName);
pnBox.add(pnName);
JPanel pnButton=new JPanel();
btnOk=new JButton("OK");
pnButton.add(btnOk);
pnBox.add(pnButton);
lblId.setPreferredSize(lblName.getPreferredSize());
Container con=getContentPane();
con.add(pnBox);
btnOk.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent arg0) {
        Category cate=new Category(txtId.getText(), txtName.getText());
        MainManagerUI.listData.addCate(cate);
        MainManagerUI.updateCateList();
        dispose();
    }
}
```

```
});  
  
}  
  
}
```

**TestMain.java:**

```
public class TestMain {
```

```
    public static void main(String[] args) {
```

```
        MainManagerUI uiProduct=new MainManagerUI("Quản lý sản phẩm");
```

```
        uiProduct.doShow();
```

```
    }
```

```
}
```

## **Bài 2: Quản lý sinh viên với mô hình MVC**

**Đề bài:** Viết chương trình **quản lý sinh viên trong Java** , sử dụng Swing để tạo giao diện và áp dụng mô hình MVC. Mỗi đối tượng sinh viên có các thuộc tính sau: id, name, age, address và gpa (điểm trung bình). Với các chức năng sau:

1. Sử dụng mô hình MVC.
2. Tạo màn hình đăng nhập.
3. Add student.
4. Edit student.
5. Delete student.
6. Sắp xếp student theo GPA.
7. Sắp xếp student theo Name.
8. Hiện thị danh sách student.
9. Lưu danh sách sinh viên vào file "student.xml".

### **Hướng dẫn:**

#### **1) Mô tả**

Bài này sẽ áp dụng mô hình MVC và Java Swing để tạo chương trình quản lý sinh viên. ( Sử dụng maven để quản lý project trong môi trường NetBeans)

- Tầng M (model) bao gồm package ***my.com.qlsv.dao*** và ***my.com.qlsv.entity***
  - Lớp **User.java** để lưu thông tin người dùng.
  - Lớp **UserDao.java** chứa phương thức `checkUser()` để kiểm tra thông tin đăng nhập.
  - Lớp **Student.java** để lưu thông tin cho mỗi sinh viên.
  - Lớp **StudentXML.java** để lưu thông tin danh sách sinh viên với định dạng XML vào file `student.xml`.
  - Lớp **StudentDao.java** chứa các phương thức quản lý sinh viên như thêm, sửa, xóa, sắp xếp, đọc, ghi sinh viên.
- Tầng V (view) bao gồm package ***my.com.qlsv.view***
  - Lớp **LoginView.java** tạo màn hình login.
  - Lớp **StudentView.java** tạo màn hình quản lý sinh viên.

Tầng C (controller) bao gồm package ***my.com.qlsv.controller***

- Lớp **LoginController.java** xử lý các sự kiện từ `LoginView.java`.
- Lớp **StudentController.java** xử lý các sự kiện từ `StudentView.java`.

Các file khác:

- Lớp **FileUtils.java** được sử dụng để đọc ghi file.
- Lớp **App.java** chứa hàm `main` để khởi chạy ứng dụng.
- File **student.xml** được sử dụng để lưu danh sách sinh viên.

Thêm các thư viện sau vào file `pom.xml`

- **jaxb-api-2.3.0.jar** : chuyển đối tượng thành xml và lưu vào file, đọc file và chuyển xml thành đối tượng.
- **junit-3.8.1.jar**: Test

**Đặt 2 thư viện vào thư mục theo đường dẫn sau:** (nếu chưa có thì tạo ra)

C:\Users\Admin\.m2\repository\junit\junit\3.8.1\junit-3.8.1.jar

C:\Users\Admin\.m2\repository\javax\xml\bind\jaxb-api\2.3.0\jaxb-api-2.3.0.jar

## 2) Thực hiện

### 2.1. Tạo chức năng login

1. Tạo lớp User.java
2. Tạo lớp UserDao.java
3. Tạo lớp LoginView.java
4. Tạo lớp LoginController.java

### 2.2. Tạo chức năng quản lý sinh viên

Tạo màn hình quản lý sinh viên chứa các thông tin sau:

- Các trường tương ứng với các thuộc tính của sinh viên.
- Button Add.
- Button Edit.
- Button Delete.
- Button Clear.
- Bảng hiển thị danh sách sinh viên.
- Button "Sort By Name"
- Button "Sort By GPA"

#### 1. Tạo lớp Student.java

2. Tạo lớp StudentXML.java
3. Tạo lớp StudentDao.java
4. Tạo lớp StudentView.java
5. Tạo lớp StudentController.java
6. Tạo lớp FileUtils.java

### 2.3. Tạo lớp App.java

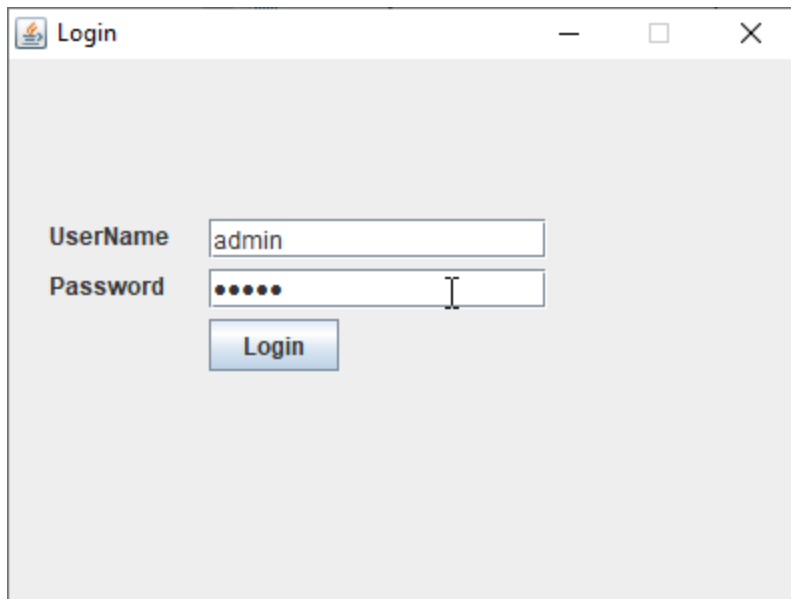
File: App.java

Lớp này chứa phương thức main() để chạy ứng dụng.

## 3. Hình ảnh giao diện

Login với username/password: admin/admin:





A screenshot of a login window titled "Login". The window has a standard Windows-style title bar with a minimize button, a maximize button, and a close button. The main area is light gray. It contains two labels, "UserName" and "Password", each followed by a text input field. The "UserName" field contains the text "admin". The "Password" field contains six dots, indicating a masked password. A blue "Login" button is positioned below the password field. A cursor is visible at the end of the password field.

Login

UserName admin

Password ●●●●●●

Login

Màn hình quản lý sinh viên:

Student Information

Id

Name

Age

Address

GPA

ID	Name	Age	Address	GPA
1	Mai Thanh Toan	22	Ha Noi	8.0
3	Dai Bang	24	Ha Nam	9.0
2	Vinh The Mac	23	Vinh Phuc	9.5

Nhập thông tin sinh viên:

Student Information

Id

Name

Age

Address

GPA

ID	Name	Age	Address	GPA
1	Mai Thanh Toan	22	Ha Noi	8.0
3	Dai Bang	24	Ha Nam	9.0
2	Vinh The Mac	23	Vinh Phuc	9.5

Thêm sinh viên:

Student Information

Id: 4

Name: Đỗ Đại Học

Age: 18

Address: Ha Noi

GPA: 9.0

Add Edit Delete Clear

ID	Name	Age	Address	GPA
1	Mai Thanh Toan	22	Ha Noi	8.0
3	Dai Bang	24	Ha Nam	9.0
2	Vinh The Mac	23	Vinh Phuc	9.5
4	Đỗ Đại Học	18	Ha Noi	9.0

Sort By GPA Sort By Name

Message

Thêm thành công!

OK

Bài 3: Sửa lại bài 1, 2 để chương trình kết nối với CSDL MySQL hoặc SQLServer.