Bài tập chương 7 Graphic User Interface

Nội dung kiến thức thực hành:

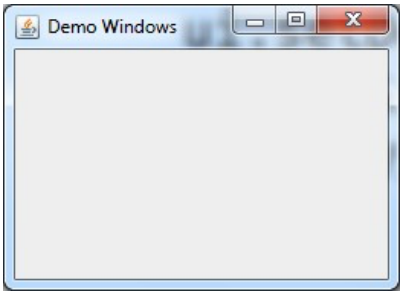
- Mục đích của Module này giúp các sinh viên hiểu được LayoutManager, Common Control, Event, DialogBox, Advanced Control

Bài tập 62.

Mục đích:

- Thực hành cách hiển thị cửa sổ Windows trong Java

Yêu cầu:



Hãy hiển thị cửa sổ trên, yêu cầu viết class kế thừa từ JFrame

Hướng dẫn:

```
public class MyWindow extends JFrame{
  public MyWindow(){
   super("Demo Windows");
   setDefaultCloseOperation(EXIT_ON_CLOSE);
  }
  public static void main(String[] args) {
   MyWindow ui=new MyWindow();
   ui.setSize(400, 300);
   ui.setLocationRelativeTo(null);
   ui.setVisible(true);
  }
}
```

Giải thích:

```
super("Demo Windows");
  Use to set title for this window
setDefaultCloseOperation(EXIT_ON_CLOSE);
  Allow click 'x' Top right corner to
close the window
ui.setSize(400, 300);
  set Width =400 and Height =300
ui.setLocationRelativeTo(null);
  Display window on desktop center screen
ui.setVisible(true);
  Show the window
```

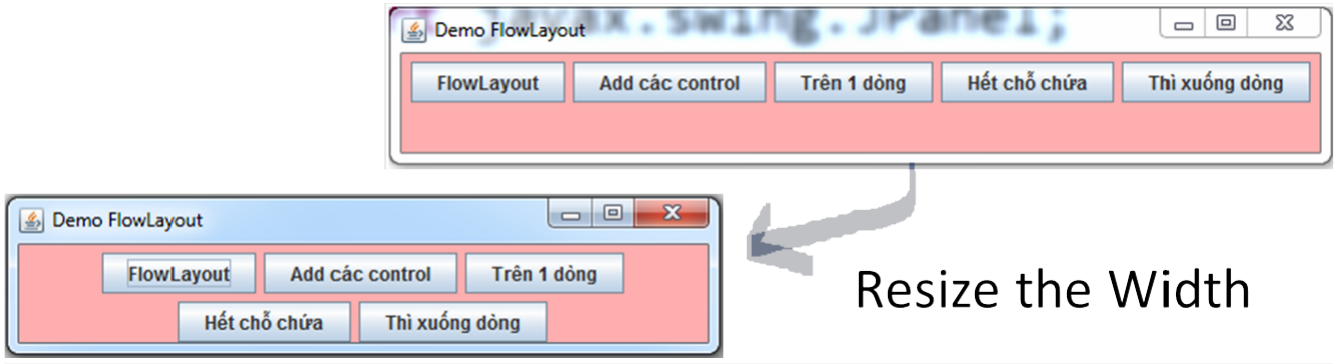
Bài tập 63.

Mục đích:

- Thực hành về FlowLayout

Yêu cầu:

FlowLayout cho phép add các control trên cùng một dòng, khi nào hết chỗ chứa nó sẽ tự động xuống dòng, ta cũng có thể điều chỉnh hướng xuất hiện của control. Mặc định khi một JPanel được khởi tạo thì bản thân lớp chứa này sẽ có kiểu Layout là FlowLayout.



Hướng dẫn:

```
JPanel pnFlow=new JPanel();
pnFlow.setLayout(new FlowLayout());
pnFlow.setBackground(Color.PINK);
JButton btn1=new JButton("FlowLayout");
JButton btn2=new JButton("Add các control");
JButton btn3=new JButton("Trên 1 dòng");
JButton btn4=new JButton("Hết chỗ chứa");
JButton btn5=new JButton("Thì xuống dòng");
pnFlow.add(btn1);pnFlow.add(btn2);
pnFlow.add(btn3);pnFlow.add(btn4);
pnFlow.add(btn5);
   Container con=getContentPane();
   con.add(pnFlow);
```

Bài tập 64.

Mục đích:

- Thực hành về **BoxLayout**

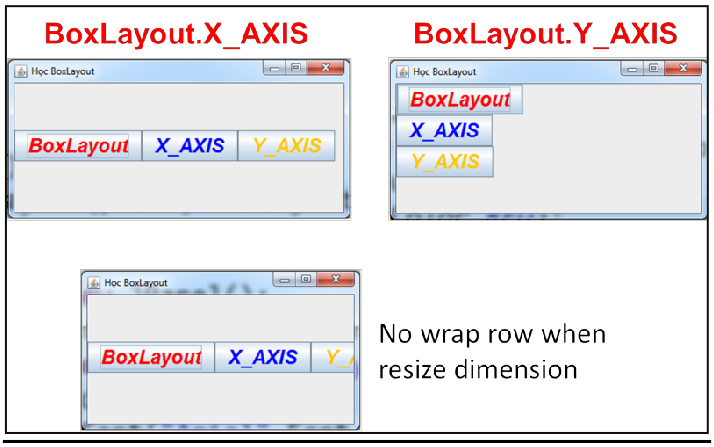
Yêu cầu:

BoxLayout cho phép add các control theo dòng hoặc cột, tại mỗi vị trí add nó chỉ chấp nhận 1 control, do đó muốn xuất hiện nhiều control tại một vị trí thì bạn nên add vị trí đó là 1 JPanel rồi sau đó add các control khác vào JPanel này.

BoxLayout.X_AXIS : Cho phép add các control theo hướng từ trái qua phải.

BoxLayout.Y_AXIS : Cho phép add các control theo hướng từ trên xuống dưới.

BoxLayout sẽ không tự động xuống dòng khi hết chỗ chứa, tức là các control sẽ bị che khuất nếu như thiếu không gian chứa nó.



Hướng dẫn:

```
JPanel pnBox=new JPanel();
pnBox.setLayout(new BoxLayout(pnBox, BoxLayout.X_AXIS));
JButton btn1=new JButton("BoxLayout");
btn1.setForeground(Color.RED);
Font font=new Font("Arial",Font.BOLD | Font.ITALIC,25);
btn1.setFont(font);pnBox.add(btn1);
JButton btn2=new JButton("X_AXIS");
btn2.setForeground(Color.BLUE);
btn2.setFont(font);pnBox.add(btn2);
JButton btn3=new JButton("Y_AXIS");
btn3.setForeground(Color.ORANGE);
btn3.setFont(font);pnBox.add(btn3);

Container con=getContentPane();
con.add(pnBox);
```

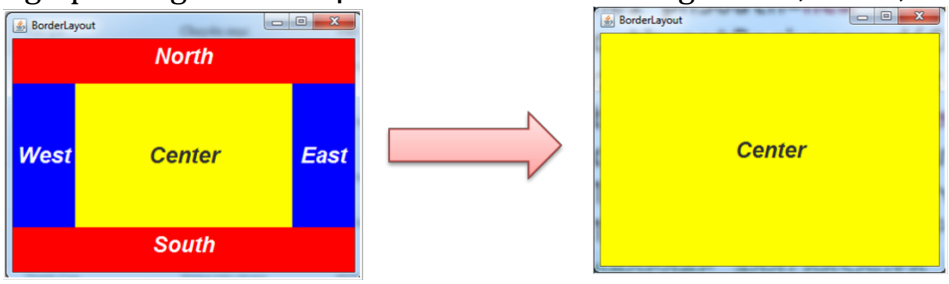
Bài tập 65.

Mục đích:

- Thực hành về **BoxLayout**

Yêu cầu:

BorderLayout giúp chúng ta hiển thị các control theo 5 vùng: North, South, West, East, Center

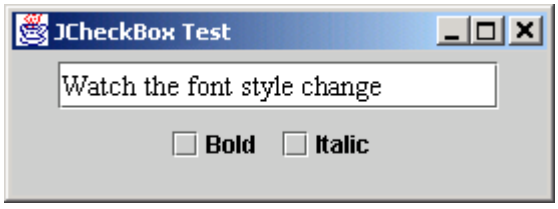


Nếu như không có 4 vùng : North, West, South, East. Thì vùng Center sẽ tràn đầy cửa sổ, thông thường khi đưa các control JTable, JTree, ListView, JScrollpane… ta thường đưa vào vùng Center để nó có thể tự co giãn theo kích thước cửa sổ giúp giao diện đẹp hơn.

```
JPanel pnBorder=new JPanel();
    pnBorder.setLayout(new BorderLayout());
JPanel pnNorth=new JPanel();
pnNorth.setBackground(Color.RED);
    pnBorder.add(pnNorth,BorderLayout.NORTH);
JPanel pnSouth=new JPanel();
pnSouth.setBackground(Color.RED);
    pnBorder.add(pnSouth,BorderLayout.SOUTH);
JPanel pnWest=new JPanel();
pnWest.setBackground(Color.BLUE);
    pnBorder.add(pnWest,BorderLayout.WEST);
JPanel pnEast=new JPanel();
pnEast.setBackground(Color.BLUE);
    pnBorder.add(pnEast,BorderLayout.EAST);
JPanel pnCenter=new JPanel();
pnCenter.setBackground(Color.YELLOW);
    pnBorder.add(pnCenter,BorderLayout.CENTER);
getContentPane().add(pnBorder);
```

Bài tập 66.

Thiết kế giao diện như sau:

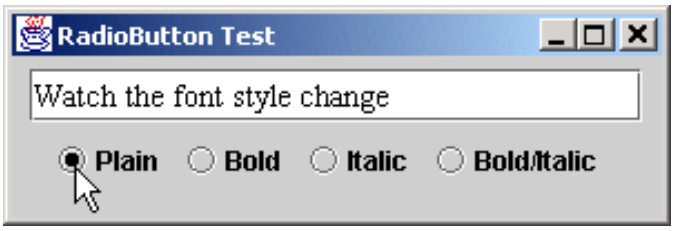


Hoạt động: Chương trình cho phép thay đổi định dạng chữ trong ô JTextField khi nhấn

chọn checkbox tương ứng.

Bài tập 67.

Thiết kế giao diện như sau:



Hoạt động: Chương trình cho phép thay đổi định dạng của chữ trong ô JTextField khi nhấn chọn radiobutton tương ứng.

Bài tập 68.

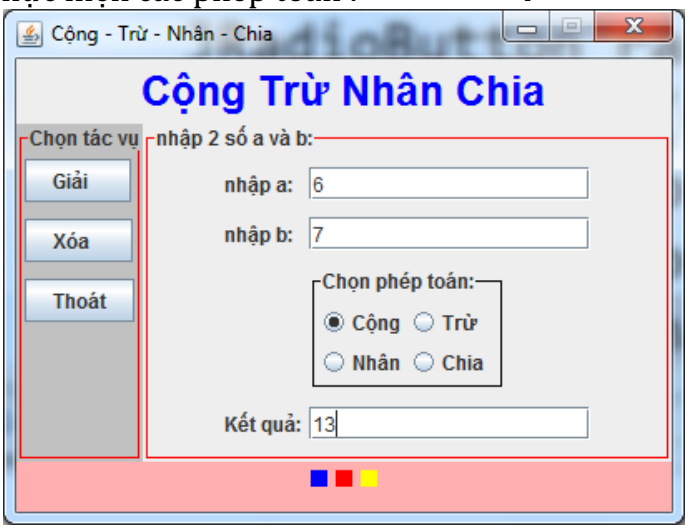
Thiết kế giao diện để giải phương trình bậc 2:

Hướng dẫn: Sinh viên phải xác định Layout Manager trước, ta cũng có thể kế hợp các Layout để thiết kế giao diện, đặt tên control theo yêu cầu bên dưới

Tên Control	Tên Biến Control	Mô tả
JTextField	txtSoa	Dùng để nhập giá trị cho a
JTextField	txtSob	Dùng để nhập giá trị cho b
JTextField	txtSoc	Dùng để nhập giá trị cho c
JTextField	txtKetqua	Dùng để hiển thị kết quả
JButton	btnGiai	Viết lệnh để giải phương
JButton	btnXoaTrang	Xóa toàn bộ dữ liệu trong ô dl
JButton	btnThoat	Viết lệnh thoát chương trình
JLabel	lblTieuDe	Giải Phương Trình Bậc 2

Bài tập 69.

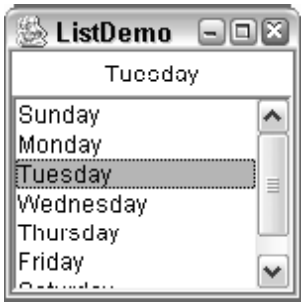
Thiết kế giao diện để thực hiện các phép toán : **'+' '-' '*' ':'**



Khi bấm nút Giải thì tùy thuộc vào phép toán được chọn mà kết quả thực hiện khác nhau.

Bài tập 70.

Thiết kế giao diện như sau:



Hoạt động: Khi người dùng chọn một dòng trên JList thì dòng đó sẽ hiện ra trên JLabel bên trên. Yêu cầu thiết lập:

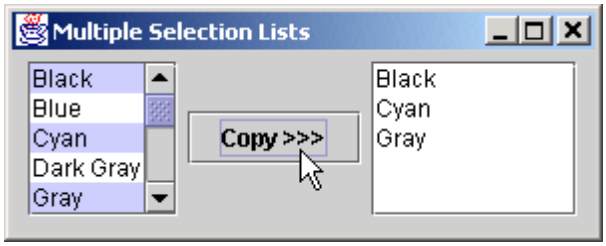
+ Nội dung trong JLabel được canh giữa, tạo đường viền, đổi màu nền, đổi màu chữ cho JLabel.

+ Không cho phép chọn nhiều dòng trên JList.

+ Khi chương trình hiện lên thì dòng đầu tiên phải được chọn.

Bài tập 71.

Thiết kế giao diện như sau:

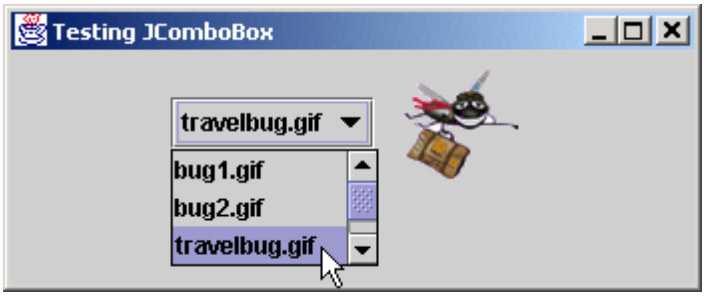


Chương trình cho phép người dùng sao chép các mục chọn trong JList bên trái qua JList bên phải khi nhấn nút "Copy>>>".

Dữ liệu trong các JList phải được đọc từ file ra và khi nhấn nút đóng chương trình thì chương trình sẽ cho phép lưu dữ liệu thay đổi trong JList bên phải vào file. Thêm vào chương trình menu để lưu, đóng, sao chép.

Bài tập 72.

Thiết kế giao diện như sau:

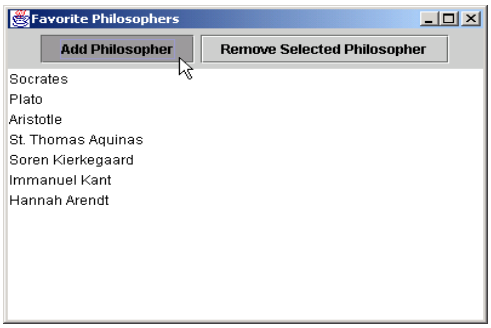


JComboBox chứa sẵn một số tên file hình, khi chương trình hiện lên thì cho xuất hiện hình đầu
tiên lên JLabel.

Hoạt động: Khi người dùng chọn tên hình nào trong JComboBox thì chương trình sẽ hiện hình đó ra label.

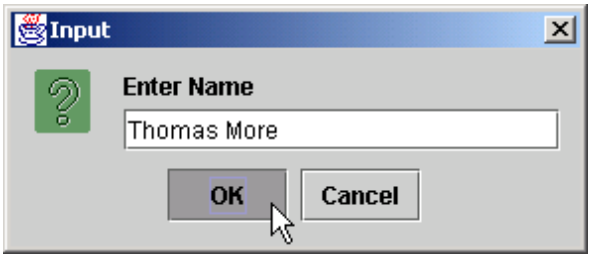
Bài tập 73.

Thiết kế giao diện như sau: giao diện gồm 2 JButton và 1 JList có thanh cuộn.



Hoạt động: Chương trình cho phép thêm và xóa một mục trong JList như sau:

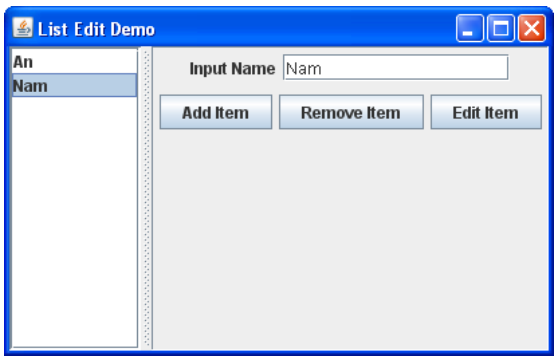
+ Khi người dùng nhấn nút "**Add Philosopher**" thì chương trình hiện ra một cửa sổ cho nhập



+ Khi người dùng nhấn nút "**Remove Selected Philosopher**" thì chương trình sẽ xóa mục đang chọn trong JList, trước khi xóa phải hỏi xác nhận lại, nếu không chọn mục nào để xóa thì phải thông báo.

Bài tập 74.

Thiết kế giao diện như hình sau:



Yêu cầu xử lý:

+ Khi nhấn nút "**Add Item**" thì thêm nội dung ô nhập vào JList, cảnh báo người dùng trường hợp để trống ô nhập hoặc nhập trùng dữ liệu đã có.

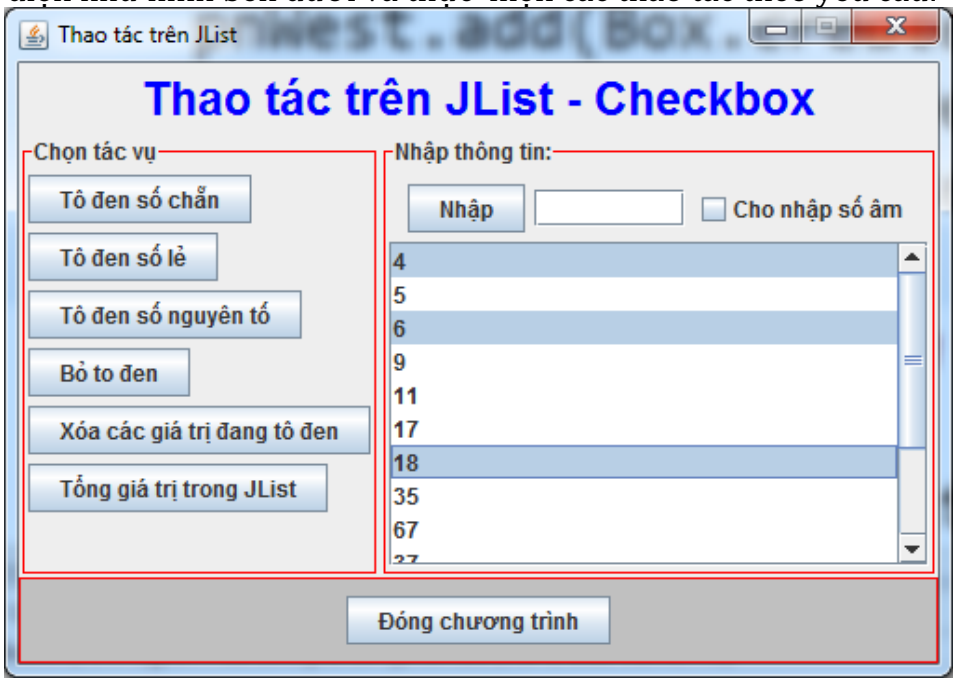
+ Khi nhấn nút "**Remove Item**" thì cho phép xóa các dòng đang chọn trong JList, trước khi xóa phải hỏi xác nhận lại, cảnh báo người dùng trường hợp không chọn mà xóa.

+ Khi nhấn nút "**Edit Item**" thì cho phép sửa nội dung dòng đang chọn thành nội dung mới trong ô nhập liệu.

+ Khi nhấn chọn một dòng trên JList thì hiện nội dung dòng đó lên ô nhập liệu.

Bài tập 75.

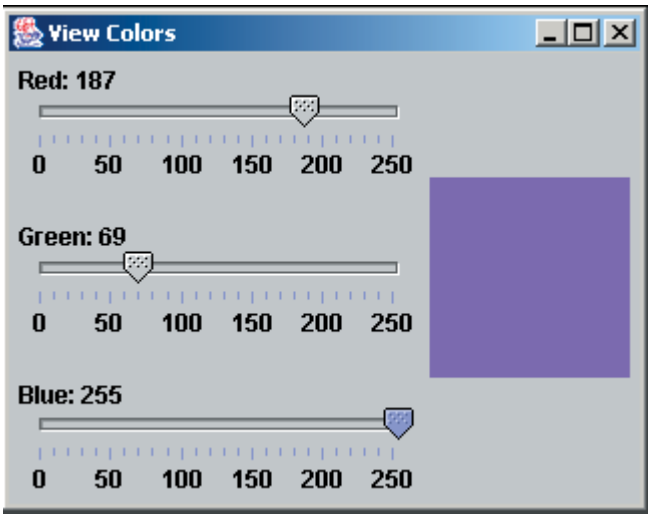
Thiết kế giao diện như hình bên dưới và thực hiện các thao tác theo yêu cầu:



- Chương trình cho phép nhập vào các số nguyên từ giao diện trong phần nhập thông tin, Khi người sử nhập giá trị vào JTextField và click nút "Nhập" thì sẽ cập nhập dữ liệu xuống JList, Nếu checked vào "Cho nhập số âm" thì các số âm mới được phép đưa vào JList còn không thì thông báo lỗi.
- Ô Chọn tác vụ, sinh viên phải thực hiện toàn bộ các yêu cầu
- Nút Đóng chương trình: sẽ hiển thị thông báo hỏi người sử dụng có muốn đóng hay không.

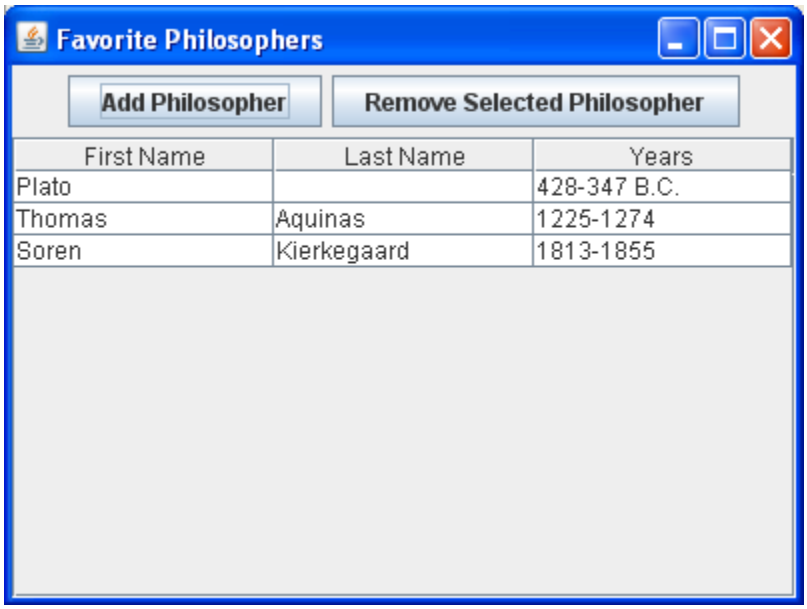
Bài tập 76.

Viết chương trình đổi màu panel dùng JSlider như hình:



Bài tập 77.

Thiết kế giao diện như hình bên dưới:



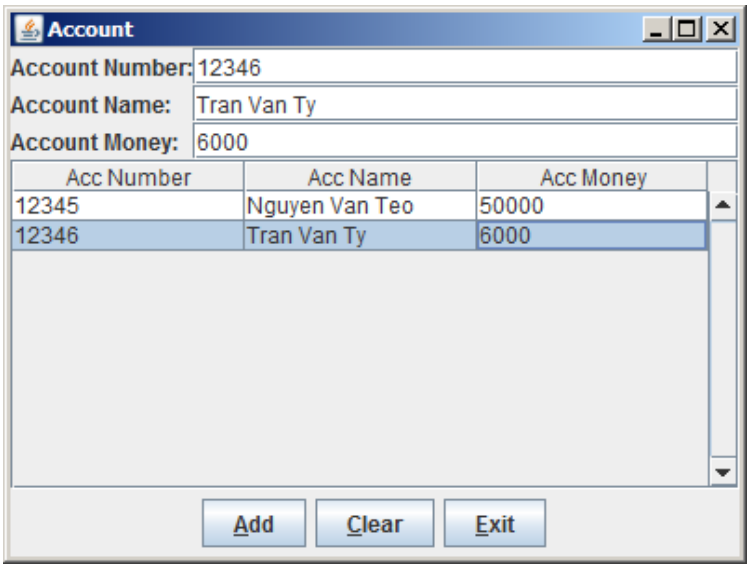
+ Khi người dùng nhấn nút "**Add Philosopher**" thì chương trình lần lượt hiện ra các

cửa sổ cho nhập vào First Name, Last Name và Years, sau đó đưa các thông tin này lên JTable

+ Khi người dùng nhấn nút "**Remove Selected Philosopher**" thì chương trình sẽ xóa mục đang chọn trong JTable, nếu không chọn mục nào để xóa thì phải thông báo, trước khi xóa phải hỏi xác

Bài tập 78.

Viết chương trình quản lý Account



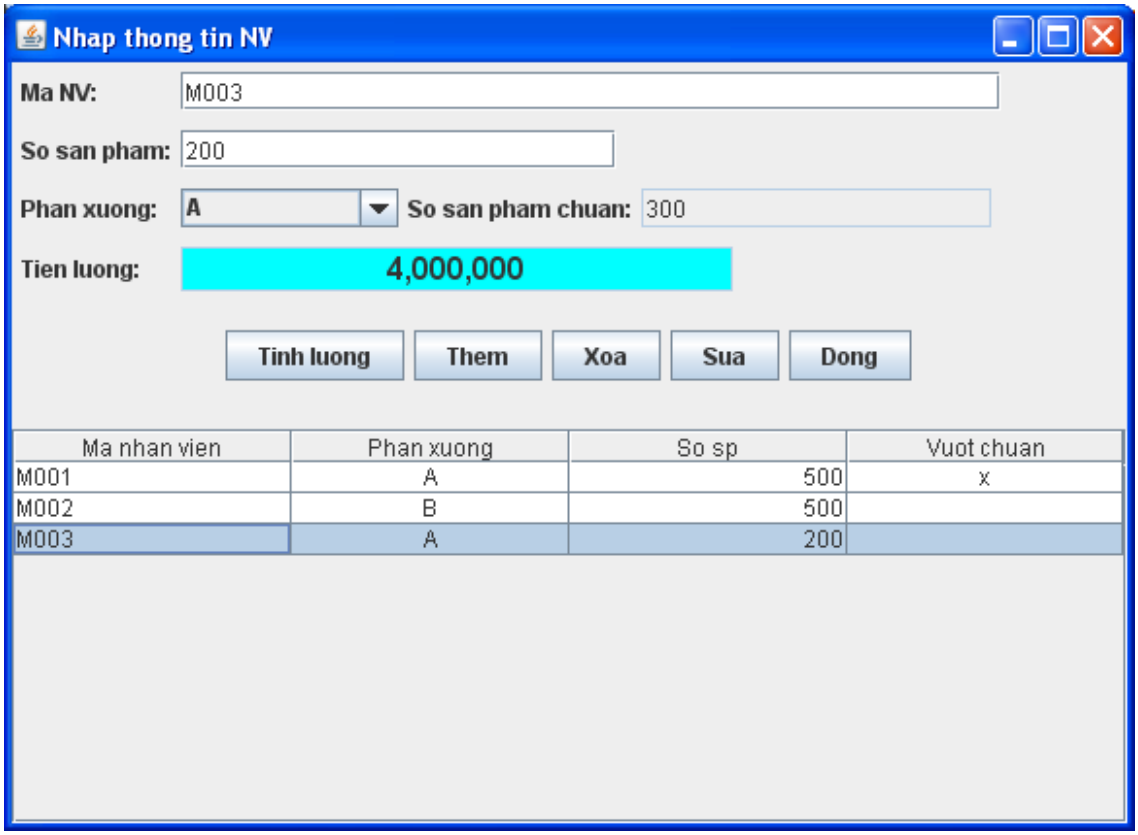
Bài tập 79.

i. Viết lớp NhanVien với các yêu cầu sau:

- Thuộc tính: **mã nhân viên** (String), **phân xưởng** (String), **số sản phẩm**
- (int). Các hàm khởi tạo.
- Các hàm get/set.
- Hàm **getChuan**(): trả về giá trị 300 nếu là phân xưởng A, còn lại trả về 500. (chỉ có các phân xưởng là A, B, C, D).
- Hàm **VuotChuan**(): trả về true khi số sản phẩm vượt chuẩn (chuẩn tùy theo phân xưởng), ngược lại trả về false.
- Hàm **TinhLuong**(): trả về lương của nhân viên, lương = số sản phẩm * đơn giá, nếu số sản phẩm vượt chuẩn thì phần vượt được tính đơn giá là 30000, còn lại tính đơn giá là 20000.
- Hàm **toString**(): trả về mã nhân viên.

ii. Thiết kế giao diện như hình bên dưới

Giao diện cho nhập mã nhân viên, số sản phẩm, chọn phân xưởng. Yêu cầu ô Tiền lương được định dạng như hình.



Yêu cầu xử lý:

+ Khi người dùng chọn phân xưởng nào trên combobox thì hiện Số sản phẩm chuẩn tương ứng.

+ Nút "**Tính lương**": xuất ra lương của nhân viên đang nhập (xem hình). Yêu cầu kiểm tra dữ liệu nhập.

+ Nút "**Thêm**": thêm nhân viên đang nhập vào JTable với các cột như hình. Chú ý không được thêm khi:

- Không nhập đủ dữ liệu.
- Số sản phẩm không phải là số.
- Trùng mã nhân viên.

+ Nút "**Xóa**": xóa một nhân viên đang chọn trên JTable. Chú ý phải hỏi trước khi xóa và không được xóa khi chưa chọn dòng nào.

+ Khi chọn 1 dòng trên JTable thì hiện thông tin nhân viên đó lên các ô nhập liệu (xem hình).

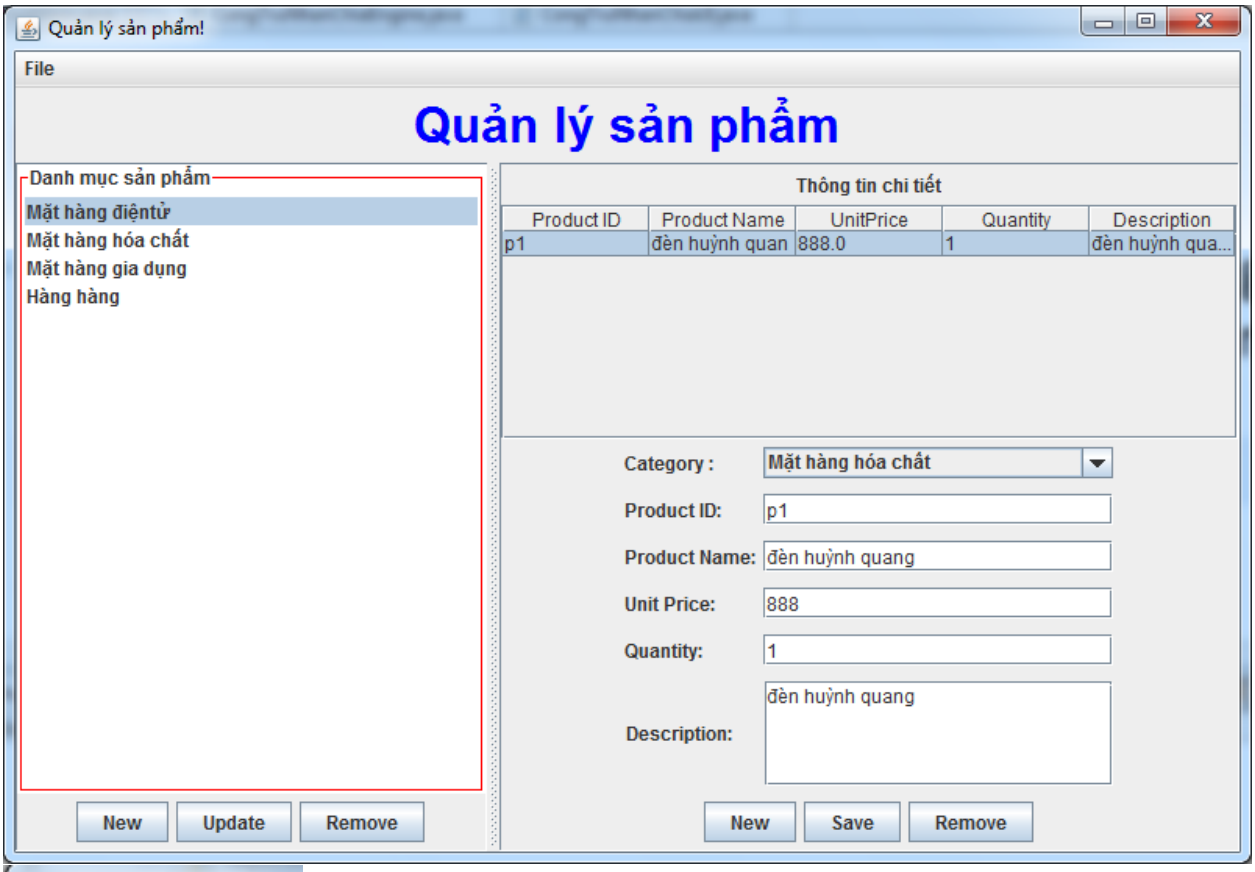
+ Khi người dùng nhấn nút "**Sửa**" thì chương trình sửa thông tin của nhân viên đang chọn vào JTable. Chú ý không được sửa khi:

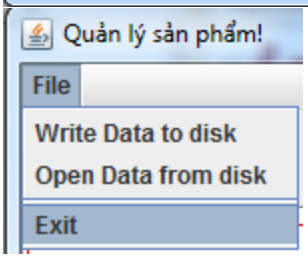
- Không có đủ dữ liệu.
- Số SP không phải là số.
- Trùng mã nhân viên.

Bài tập 80.

Viết chương trình quản lý sản phẩm
Yêu cầu chức năng: Cho phép nhập/ xuất danh mục, danh sách sản phẩm

- Cho phép cập nhật thông tin
- Cho phép lưu / đọc danh mục sản phẩm
- Yêu cầu sử dụng JMenuBar, JList, JTable, JCombobox, …





Menu **Write Data to disk** dùng để lưu dữ liệu xuống ổ cứng
Menu **Open Data from disk** để đọc dữ liệ từ ổ cứng
Menu **Exit** dùng để thoát chương trình

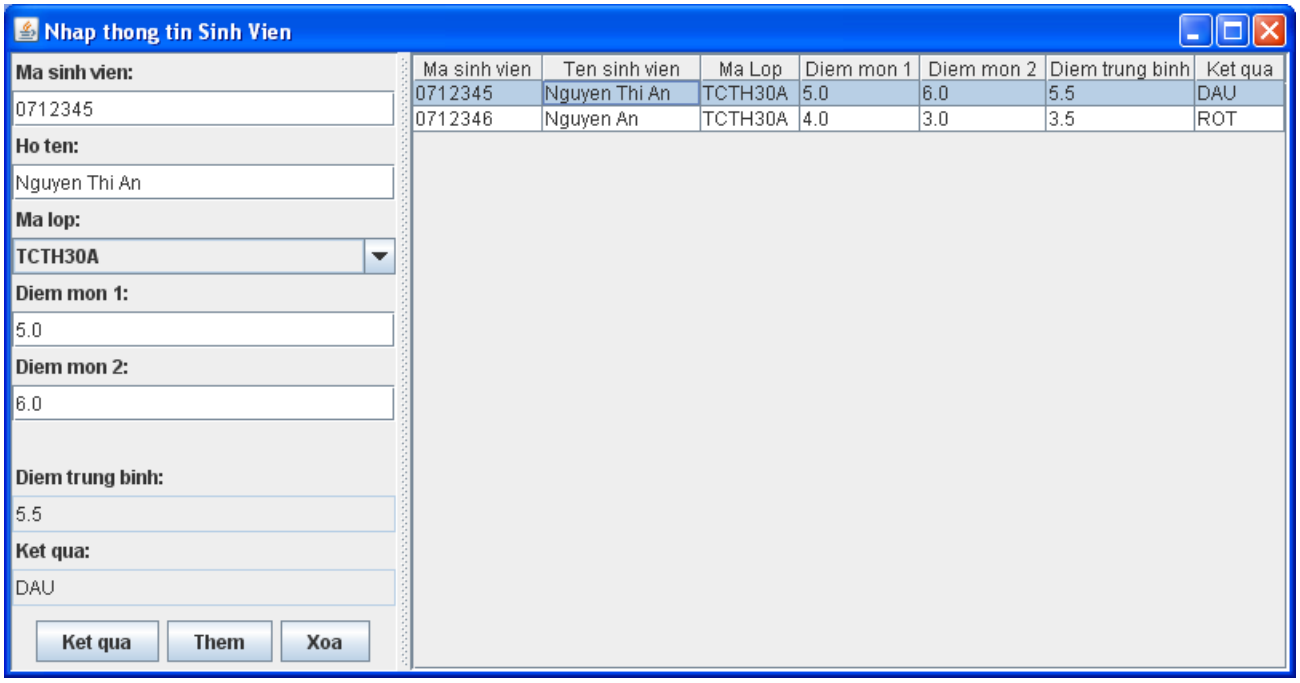
Bài tập 81.

(i).Viết lớp SinhVien với yêu cầu sau:

- Thuộc tính: mã sinh viên (String), họ tên (String), mã lớp (String), điểm môn 1 (double), điểm môn 2 (double).
- Các hàm khởi tạo.
- Các hàm get/set.
- Hàm DiemTrungBinh(): trả về điểm trung bình của 2 môn học.
- Hàm KetQua(): trả về "Đậu" khi điểm trung bình >=5, ngược lại là "Rớt".
- Hàm toString(): trả về mã sinh viên.

(ii) Thiết kế giao diện như hình bên dưới

Giao diện cho nhập mã sinh viên, họ tên, chọn mã lớp, nhập điểm môn 1, môn 2.



Yêu cầu xử lý:

+ Nút "**Kết quả**": xuất ra điểm trung bình và kết quả của sinh viên đang nhập (xem hình). Chú ý ô nhập Điểm môn 1 và môn 2 phải là số.

+ Nút "**Thêm**": thêm một sinh viên vào table với các cột như hình. Chú ý không được thêm khi:

- Không nhập đủ dữ liệu.
- Điểm môn 1 và môn 2 không phải là số.
- Trùng mã sinh viên.

+ Nút "**Xóa**": xóa một sinh viên đang chọn trên table. Chú ý phải hỏi trước khi xóa và không được xóa khi chưa chọn dòng nào.

+ Khi chọn 1 dòng trên table thì hiện thông tin sinh viên đó lên các ô nhập liệu (xem hình).

Bài tập 82.

Thiết kế giao diện và thực hiện như hình bên dưới:

Chương trình cho phép thêm 1 nút con hoặc nút anh em của nút đang chọn khi người dùng

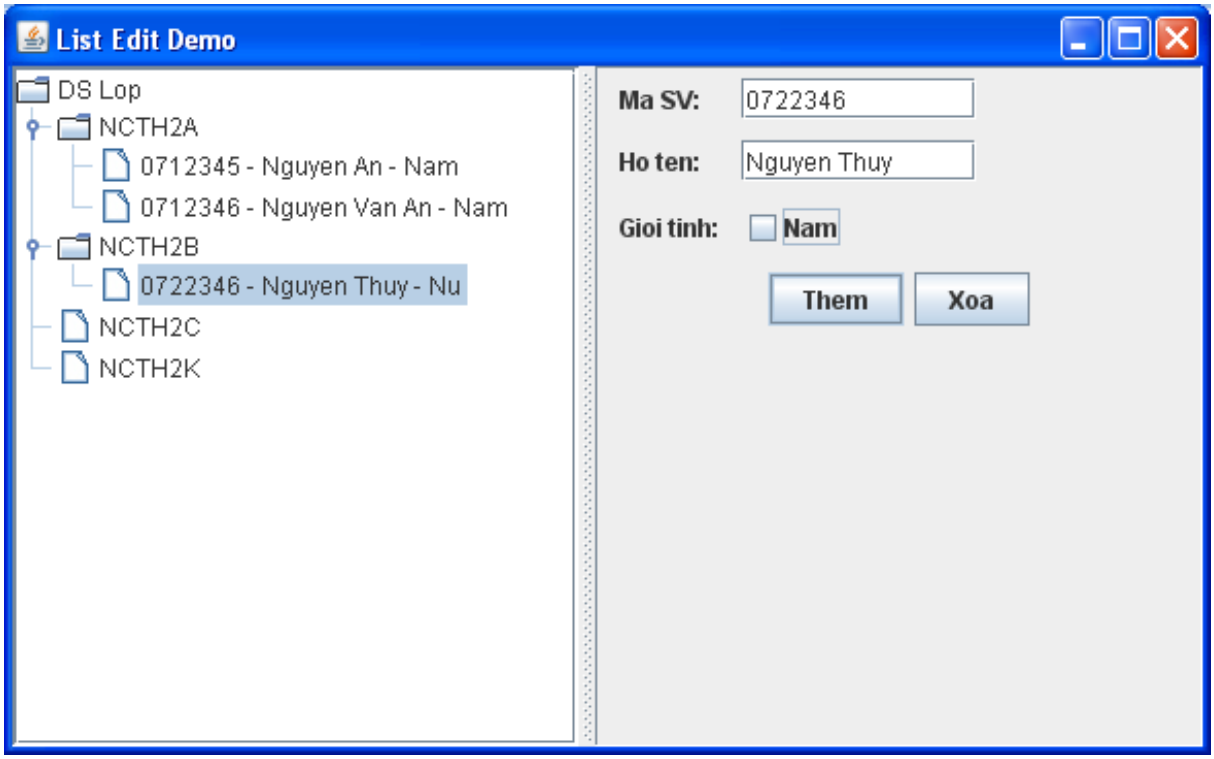
nhấn nút "**Add Child**" hoặc "**Add Sibling**", nút "**Delete**" sẽ xóa nút đang chọn.

Bài tập 83.

Viết lớp SinhVien với yêu cầu sau:

- Thuộc tính: mã sinh viên (String), họ tên (String), giới tính (boolean), mã lớp (String).
- Các hàm khởi tạo.
- Các hàm get/set.
- Hàm toString(): trả về chuỗi thông tin gồm: mã sinh viên – họ tên – giới tính.

(i) Thiết kế giao diện như hình bên dưới:



Yêu cầu giao diện: Trên JTree có sẵn nút gốc "DS lop" và các nút con là các mã lớp: NCTH2A, NCTH2B, NCTH2C, NCTH2K.

Yêu cầu xử lý:

+ Nút "**Thêm**": thêm một sinh viên vào lớp đang chọn trên JTree (xem hình). Chú ý không được thêm khi:

- Không nhập đủ dữ liệu.
- Không chọn mã lớp trên JTree.
- Trùng mã sinh viên.

+ Nút "**Xóa**": xóa sinh viên đang chọn trên JTree. Chú ý phải hỏi trước khi xóa và không được xóa khi:

- Không chọn nút muốn xóa.
- Chọn vào nút gốc hoặc nút mã lớp.

+ Khi chọn nút sinh viên thì hiện thông tin sinh viên đó lên các ô nhập liệu (xem hình).

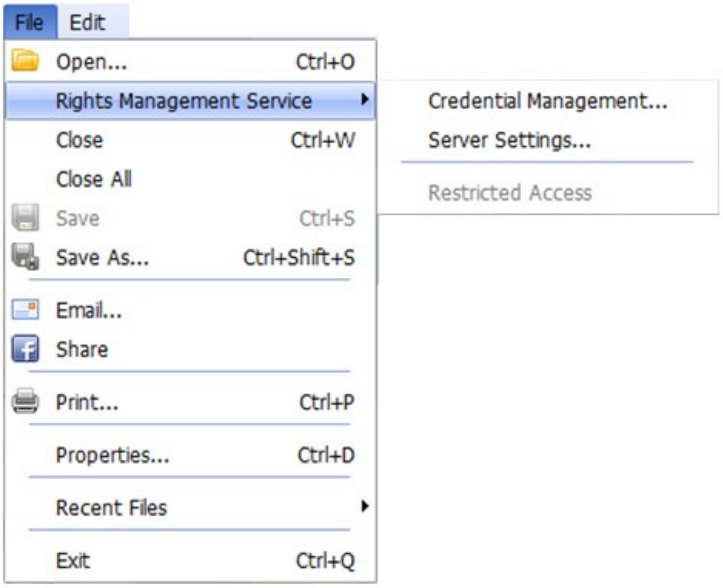
Bài tập 84.

Thực hành về tạo Menu. Yêu cầu thiết kế Menu theo hình sau, ứng với mỗi menu item sinh viên
hãy cài đặt coding để hiển thị thông báo là đang chọn menu nào.

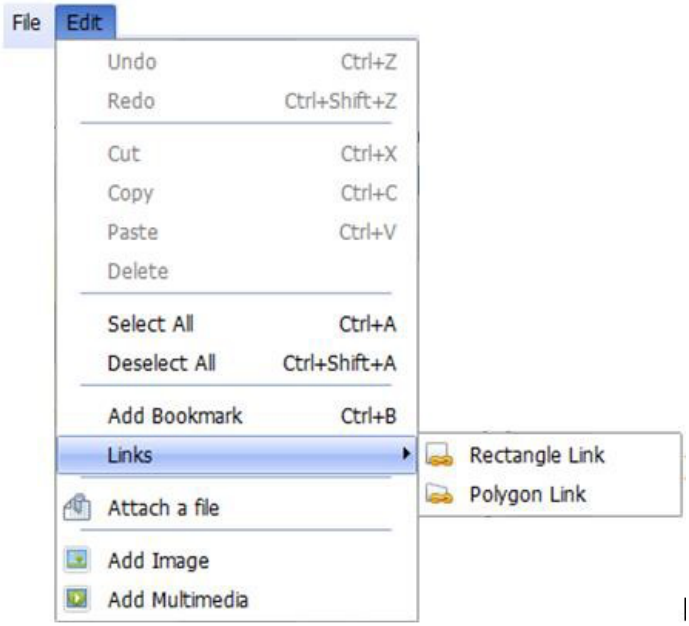
Hướng dẫn: JMenuBar JMenu JMenuItem. Phải biết kết hợp các class này.

MenuBar sẽ add Menu, Menu sẽ add MenuItem, rồi gọi setJMenuBar(menuBar); Yêu cầu giả

lập Menu giống như chương trình Foxit Reader:



Menu File có giao diện như trên



Menu Edit có giao diện như trên

Bài tập 85.

Thực hành về JToolBar, tương tự như câu 12, giả lập Toolbar của chương trình Foxit Reader, ứng với mỗi lệnh trên JToolBar, sinh viên hãy xuất thông báo đang sử dụng chức năng nào.
Hướng dẫn: tạo các JButton rồi add vào JToolBar

Bài tập 86.

Thực hành về Timer class

Dùng class Timer để thiết kế ứng dụng **ImageAnimation**.

Giao diện sẽ có 2 JButton: Start và Stop. Khi bấm Start chương trình sẽ hiển thị hình ảnh tuần tự trong mảng 10 hình ảnh có sẵn. Bấm Stop để tạm dừng duyệt hình ảnh. Xem hình yêu cầu

Hướng dẫn: Dùng CardLayout và Timer

```
import java.awt.*; import
java.awt.event.*; import
javax.swing.*;
public class ImageAnimation extends JFrame{
        private static final long serialVersionUID = 1L; Timer timer;
        private int pos=1;
        public ImageAnimation(String title)
        {
                super(title);
                timer=new Timer(500, null);
        }
        public void doShow()
```

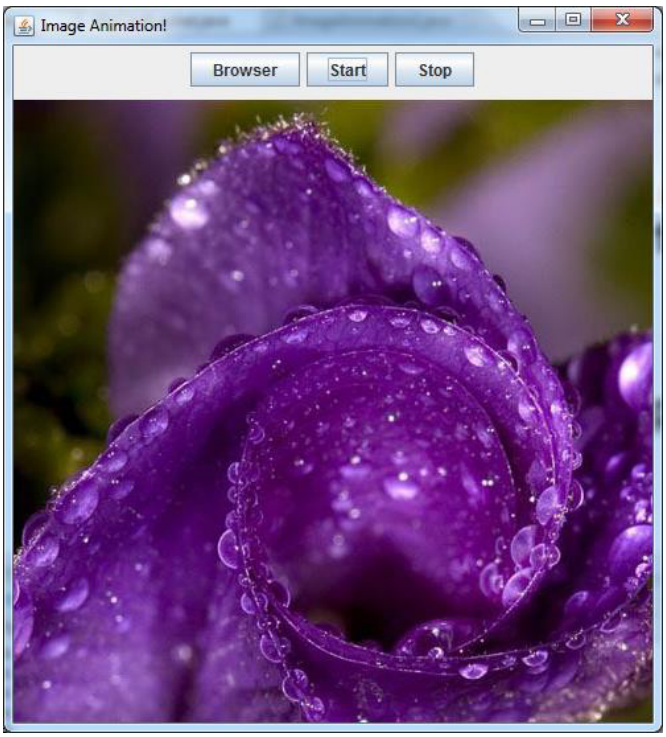
```
{
        setSize(500,550); setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        addControl();
        setVisible(true);
}
public void addControl()
{
        JPanel pnBorder=new JPanel();
        pnBorder.setLayout(new BorderLayout()); JPanel
        pnNorth=new JPanel();
        JButton btnStart=new JButton("Start"); JButton
        btnStop=new JButton("Stop");
        pnNorth.add(btnStart); pnNorth.add(btnStop);
        pnBorder.add(pnNorth,BorderLayout.NORTH);

        final JPanel pnCenter=new JPanel();
        pnCenter.setLayout(new CardLayout());
        pnBorder.add(pnCenter,BorderLayout.CENTER);
        pnCenter.setBackground(Color.RED);
        JPanel []pnArr=new JPanel[10]; addImage(pnCenter,pnArr);
        showImage(pnCenter,"card1");
        btnStart.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) {
                        timer.start();
                        timer.addActionListener(new TimerPanel(pnCenter));
                }
        });
        btnStop.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent arg0) {
                        timer.stop();
                }
        });
        Container con=getContentPane();
        con.add(pnBorder);
}
private void addImage(JPanel pnCenter,JPanel []pnArr)
{
        for(int i=0;i<pnArr.length;i++)
        {
                pnArr[i]=new JPanel(); JLabel
                lbl=new JLabel();
                ImageIcon icon=new ImageIcon("E:\\hoa\\"+i+".jpg");
                lbl.setIcon(icon);
                pnArr[i].add(lbl);
                pnCenter.add(pnArr[i],"card"+i);
        }
}
public void showImage(JPanel pn,String cardName)
{
        CardLayout cl=(CardLayout)pn.getLayout();
        cl.show(pn, cardName);
```

```
	}
	private class TimerPanel implements ActionListener
	{
		JPanel pn=null;
		public TimerPanel(JPanel pn) {
			this.pn=pn;
		}
		public void actionPerformed(ActionEvent arg0) {
			showImage(pn,"card"+pos);
			pos++;
			if(pos>=10)
				pos=1;
		}
	}
	public static void main(String[] args) {;
		ImageAnimation imgUi=new ImageAnimation("Image Animation!");
		imgUi.doShow();
	}
}
```

Bài tập 87.

Cải tiến bài tập 10. Chương trình sẽ cho phép đọc danh sách các hình ảnh bất kỳ trong ổ đĩa.



Hướng dẫn:

```
import java.awt.*;
import java.awt.event.*;
import java.io.File;
import javax.swing.*;
```

```
public class ImageAnimation2 extends JFrame {
    private static final long serialVersionUID = 1L;
    Timer timer;
    private int pos = 0;

    public ImageAnimation2(String title) {
        super(title);
        timer = new Timer(500, null);
    }

    public void doShow() {
        setSize(500, 550);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        addControl();
        setVisible(true);
    }

    public void addControl() {
        JPanel pnBorder = new JPanel();
        pnBorder.setLayout(new BorderLayout());
        JPanel pnNorth = new JPanel();
        JButton btnBrowser = new JButton("Browser");
        JButton btnStart = new JButton("Start");
        JButton btnStop = new JButton("Stop");
        pnNorth.add(btnBrowser);
        pnNorth.add(btnStart);
        pnNorth.add(btnStop);
        pnBorder.add(pnNorth, BorderLayout.NORTH);

        final JPanel pnCenter = new JPanel();
        pnCenter.setLayout(new CardLayout());
        pnBorder.add(pnCenter, BorderLayout.CENTER);
        pnCenter.setBackground(Color.RED);

        btnStart.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                timer.start();
                timer.addActionListener(new TimerPanel(pnCenter));
            }
        });

        btnStop.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                timer.stop();
            }
        });
        btnBrowser.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent arg0) {
                JFileChooser jfc = new JFileChooser();
                jfc.setMultiSelectionEnabled(true);
                if (jfc.showOpenDialog(null) ==
```

```
JFileChooser.APPROVE_OPTION) {
                    File[] files = jfc.getSelectedFiles();
                    for (int i = 0; i < files.length; i++) {
                        File f = files[i];
                        ImageIcon icon = new ImageIcon(f.getPath());
                        JPanel pn = new JPanel();
                        JLabel lbl = new JLabel(icon);
                        pn.add(lbl);
                        pnCenter.add(pn, "card" + i);
                    }
                }
            });
        showImage(pnCenter, "card0");
        }
        Container con = getContentPane(); con.add(pnBorder);
    }

    public void showImage(JPanel pn, String cardName) {
        CardLayout cl = (CardLayout) pn.getLayout();
        cl.show(pn, cardName);
    }

    private class TimerPanel implements ActionListener {
        JPanel pn = null;

        public TimerPanel(JPanel pn) {
            this.pn = pn;
        }

        public void actionPerformed(ActionEvent arg0) {
            showImage(pn, "card" + pos);
            pos++;
            if (pos >= pn.getComponentCount()) pos = 0;
        }
    }

    public static void main(String[] args) {
        ImageAnimation2 imgUi = new ImageAnimation2("Image Animation!");
        imgUi.doShow();
    }
}
```