MENGGUNAKAN SWING

1. 12Maret 2011
1. **Pengenalan Java Swing**

Swing adalah sebuah widget toolkit untuk Java yang merupakan bagian dari *Java Foundation Classes* (JFC) dari Sun Microsystem. Swing adaah sebuah API (*Application Programming Interface*) yang menyediakan *Graphical User Interface* (GUI) untuk program Java dan applet. Swing dikembangkan untuk menyediakan komponen GUI yang lebih canggih dari komponen sebelumnya yang bernama AWT dan bertujuan untuk mempermudah pengembangan aplikasi JAVA GUI.

AWT dan Swing keduanya menyediakan komponen GUI yang dapat digunakan dalam membuat aplikasi Java dan applet. Namun, tidak seperti beberapa komponen AWT yang menggunakan *native* code, keseluruhan Swing ditulis menggunakan bahasa pemrograman Java. Swing menyediakan implementasi platform-independent dimana aplikasi yang dikembangkan dengan platform yang berbeda dapat memiliki tampilan *look and feel* yang sama. Swing API dibangun dari beberapa API yang mengimplementasikan beberapa jenis bagian dari AWT. Kesimpulannya, komponen AWT dapat digunakan dengan komponen Swing.

☻. Komponen Swing

Package dari Swing menyediakan banyak kelas untuk membuat aplikasi GUI. Package tersebut dapat ditemukan di *javax.swing*. Komponen Swing ditulis menyeluruh menggunakan Java. Kesimpulannya, program GUI ditulis menggunakan banyak kelas dari package Swing yang mempunyai tampilan *look and feel* yang sama meski dijalankan pada beda paltform. Lebih dari itu, Swing menyediakan komponen yang lebih menarik seperti *color chooser* dan *option*

pane.

Nama dari komponen GUI milik Swing hampir sama persis dengan komponen GUI milik AWT. Perbedaan jelas terdapat pada penamaan komponen. Pada dasarnya, nama komponen Swing sama dengan nama komponen AWT tetapi dengan tambahan huruf J pada prefixnya. Sebagai contoh, satu komponen dalam AWT adalah *button class*. Sedangkan pada Swing, nama komponen tersebut menjadi *Jbutton class*.

Dalam ranah antarmuka pengguna, komponen merupakan bagian fundamental di Java. Pada prinsipnya, segala sesuatu yang kita lihat di tampilan aplikasi Java adalah suatu komponen—misalnya window, menu, dan button.

Di sisi lain, container adalah jenis komponen yang "menampung" dan mengelola komponen-komponen lainnya. Idealnya, suatu komponen harus diletakkan di sebuah container agar ia dapat digunakan.

Komponen-komponen Swing dapat diklasifikasikan ke dalam tiga bagian, yaitu top-level container, intermediate container, dan komponen atomic(tunggal).

1. Container tingkat atas (top-level) berfungsi untuk menyediakan ruang bagi komponen komponen lainnya. Container jenis ini terdiri dari JFrame, JWindow, JDialog, dan JApplet.
2. Container menengah adalah komponen (non top-level) yang keberadaannya untuk menampung komponen lainnya, misalnya panel, tabbed, dan tool bar.
3. Komponen atomic berfungsi untuk menampilkan dan/atau menerima informasi. Contoh komponen atomic adalah text field, button, dan label.

Berikut adalah ini adalah beberapa komponen dari swing

Contoh program untuk biodata Menggunakan Swing:

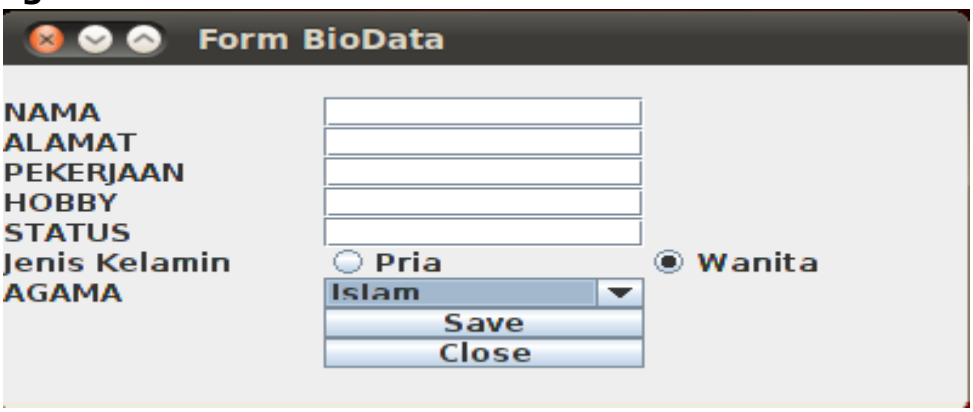
```
package javasswingdasar;
import java.awt.GridLayout;
import javax.swing.ButtonGroup;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;
import javax.swing.JComboBox;
import javax.swing.JTextField;

public class Data {
	JFrame frBioData;
	JPanel pnlData;
	JTextField jtfNama;
	JTextField jtfAlamat;
	JTextField jtfPekerjaan;
```

```
JTextField jtfHobby;
JTextField jtfStatus;
JRadioButton jrbPria;
JComboBox Agama;
JRadioButton jrbWanita;

JButton jbnSave;
JButton jbnClose;

public Data(){


        String lama="";
        String isi[]={"","Islam","Kristen","Hindu","Budha","Other","atheis"};
        frBioData = new JFrame("Form BioData");
        frBioData.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frBioData.setSize(600, 300);
        pnlData = new JPanel();


        pnlData.setLayout(new GridLayout(9,3));
        jtfNama = new JTextField("");
        jtfAlamat = new JTextField("");
jtfHobby = new JTextField("");
jtfPekerjaan = new JTextField("");
jrbPria = new JRadioButton("Pria");
        jrbWanita = new JRadioButton("Wanita" ,true);
        Agama= new JComboBox(isi);



        frBioData = new JFrame("Form BioData");
        frBioData.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frBioData.setSize(400, 200);
        pnlData = new JPanel();
        pnlData.setLayout(new GridLayout(11,3));

        jtfNama = new JTextField("");
        jtfAlamat = new JTextField("");
        jtfPekerjaan = new JTextField("");
        jtfHobby = new JTextField("");
        jtfStatus = new JTextField("");
        jrbPria = new JRadioButton("Pria");
        jrbWanita = new JRadioButton("Wanita",true);
        ButtonGroup bgjnsKel = new ButtonGroup();
        bgjnsKel.add(jrbPria);
        bgjnsKel.add(jrbWanita);
```

```
jbnSave = new JButton("Save");
jbnClose = new JButton("Close");
//row 1
pnlData.add(new JLabel(""));
pnlData.add(new JLabel(""));
pnlData.add(new JLabel(""));
//row 2
pnlData.add(new JLabel("NAMA"));
pnlData.add(jtfNama);
pnlData.add(new JLabel(""));
//row 3
pnlData.add(new JLabel("ALAMAT"));
pnlData.add(jtfAlamat);
pnlData.add(new JLabel(""));
//coba
pnlData.add(new JLabel("PEKERJAAN"));
pnlData.add(jtfPekerjaan);
pnlData.add(new JLabel(""));
//coba1
pnlData.add(new JLabel("HOBBY"));
pnlData.add(jtfHobby);
pnlData.add(new JLabel(""));
//coba3
pnlData.add(new JLabel("STATUS"));
pnlData.add(jtfStatus);
pnlData.add(new JLabel(""));

//row 4
pnlData.add(new JLabel("Jenis Kelamin"));
pnlData.add(jrbPria);
pnlData.add(jrbWanita);
//coba
pnlData.add(new JLabel("AGAMA"));
pnlData.add(Agama);


//row 5
pnlData.add(new JLabel(""));
pnlData.add(new JLabel(""));
pnlData.add(jbnSave);
//coba4
pnlData.add(new JLabel(""));
pnlData.add(new JLabel(""));
pnlData.add(jbnClose);
//row 6
pnlData.add(new JLabel(""));
pnlData.add(new JLabel(""));
```

```
            pnlData.add(new JLabel(""));



            frBioData.add(pnlData);


            }

        public static void main(String[] args){
            Data formData = new Data();
            formData.frBioData.setVisible(true);
        }


}
```

Hasil Runing



contoh program Calkulator Menggunakan Swing:

```
package javasswingdasar;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Container;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Window;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

```
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.KeyStroke;

public class Calculator extends JFrame implements ActionListener{
	// Variables
	final int MAX_INPUT_LENGTH = 20;
	final int INPUT_MODE = 0;
	final int RESULT_MODE = 1;
	final int ERROR_MODE = 2;
	int displayMode;

	boolean clearOnNextDigit, percent;
	double lastNumber;
	String lastOperator;

	private JMenu jmenuFile, jmenuHelp;
	private JMenuItem jmenuitemExit, jmenuitemAbout;

	private JLabel jlbOutput;
	private JButton jbnButtons[];
	private JPanel jplMaster, jplBackSpace, jplControl;

	/*
	 * Font(String name, int style, int size)
     Creates a new Font from the specified name, style and point size.
	 */

	Font f12 = new Font("Times New Roman", 0, 12);
	Font f121 = new Font("Times New Roman", 1, 12);

	// Constructor
	public Calculator()
	{
		/* Set Up the JMenuBar.
		 * Have Provided All JMenu's with Mnemonics
		 * Have Provided some JMenuItem components with Keyboard
Accelerators
		 */
```

```
            jmenuFile = new JMenu("File");
            jmenuFile.setFont(f121);
            jmenuFile.setMnemonic(KeyEvent.VK_F);

            jmenuitemExit = new JMenuItem("Exit");
            jmenuitemExit.setFont(f12);
            jmenuitemExit.setAccelerator(KeyStroke.getKeyStroke(
KeyEvent.VK_X,
                                                    ActionEvent.CTRL_MASK));
            jmenuFile.add(jmenuitemExit);

            jmenuHelp = new JMenu("Help");
            jmenuHelp.setFont(f121);
            jmenuHelp.setMnemonic(KeyEvent.VK_H);

            jmenuitemAbout = new JMenuItem("About Calculator");
            jmenuitemAbout.setFont(f12);
            jmenuHelp.add(jmenuitemAbout);

            JMenuBar mb = new JMenuBar();
            mb.add(jmenuFile);
            mb.add(jmenuHelp);
            setJMenuBar(mb);

            //Set frame layout manager

            setBackground(Color.gray);

            jplMaster = new JPanel();

            jlbOutput = new JLabel("0");
            jlbOutput.setHorizontalTextPosition(JLabel.RIGHT);
            jlbOutput.setBackground(Color.WHITE);
            jlbOutput.setOpaque(true);

            // Add components to frame
            getContentPane().add(jlbOutput, BorderLayout.NORTH);

            jbnButtons = new JButton[23];
//             GridLayout(int rows, int cols, int hgap, int vgap)

            JPanel jplButtons = new JPanel();             // container for Jbuttons

            // Create numeric Jbuttons
            for (int i=0; i<=9; i++)
            {
```

```
            // set each Jbutton label to the value of index
            jbnButtons[i] = new JButton(String.valueOf(i));
        }

        // Create operator Jbuttons
        jbnButtons[10] = new JButton("+/-");
        jbnButtons[11] = new JButton(".");
        jbnButtons[12] = new JButton("=");
        jbnButtons[13] = new JButton("/");
        jbnButtons[14] = new JButton("*");
        jbnButtons[15] = new JButton("-");
        jbnButtons[16] = new JButton("+");
        jbnButtons[17] = new JButton("sqrt");
        jbnButtons[18] = new JButton("1/x");
        jbnButtons[19] = new JButton("%");

        jplBackSpace = new JPanel();
        jplBackSpace.setLayout(new GridLayout(1, 1, 2, 2));

        jbnButtons[20] = new JButton("Backspace");
        jplBackSpace.add(jbnButtons[20]);

        jplControl = new JPanel();
        jplControl.setLayout(new GridLayout(1, 2, 2 ,2));

        jbnButtons[21] = new JButton(" CE ");
        jbnButtons[22] = new JButton("C");

        jplControl.add(jbnButtons[21]);
        jplControl.add(jbnButtons[22]);

//          Setting all Numbered JButton's to Blue. The rest to Red
        for (int i=0; i<jbnButtons.length; i++) {
            jbnButtons[i].setFont(f12);

            if (i<10)
                jbnButtons[i].setForeground(Color.blue);

            else
                jbnButtons[i].setForeground(Color.red);
        }

        // Set panel layout manager for a 4 by 5 grid
        jplButtons.setLayout(new GridLayout(4, 5, 2, 2));

        //Add buttons to keypad panel starting at top left
        // First row
```

```
for(int i=7; i<=9; i++)        {
      jplButtons.add(jbnButtons[i]);
}

// add button / and sqrt
jplButtons.add(jbnButtons[13]);
jplButtons.add(jbnButtons[17]);

// Second row
for(int i=4; i<=6; i++)
{
      jplButtons.add(jbnButtons[i]);
}

// add button * and x^2
jplButtons.add(jbnButtons[14]);
jplButtons.add(jbnButtons[18]);

// Third row
for( int i=1; i<=3; i++)
{
      jplButtons.add(jbnButtons[i]);
}

//adds button - and %
jplButtons.add(jbnButtons[15]);
jplButtons.add(jbnButtons[19]);

//Fourth Row
// add 0, +/-, ., +, and =
jplButtons.add(jbnButtons[0]);
jplButtons.add(jbnButtons[10]);
jplButtons.add(jbnButtons[11]);
jplButtons.add(jbnButtons[16]);
jplButtons.add(jbnButtons[12]);

jplMaster.setLayout(new BorderLayout());
jplMaster.add(jplBackSpace, BorderLayout.WEST);
jplMaster.add(jplControl, BorderLayout.EAST);
jplMaster.add(jplButtons, BorderLayout.SOUTH);

// Add components to frame
getContentPane().add(jplMaster, BorderLayout.SOUTH);
requestFocus();

//activate ActionListener
for (int i=0; i<jbnButtons.length; i++){
```

```
            jbnButtons[i].addActionListener(this);
        }

        jmenuitemAbout.addActionListener(this);
        jmenuitemExit.addActionListener(this);

        clearAll();

        //add WindowListener for closing frame and ending program
        addWindowListener(new WindowAdapter() {

                public void windowClosed(WindowEvent e)
                {
                    System.exit(0);
                }
            }
        );
}       //End of Contructor Calculator

// Perform action
public void actionPerformed(ActionEvent e){
        double result = 0;

        if(e.getSource() == jmenuitemAbout){
          JDialog dlgAbout = new CustomABOUTDialog(this,
                                        "About Java Swing Calculator", true);
            dlgAbout.setVisible(true);
        }else if(e.getSource() == jmenuitemExit){
            System.exit(0);
        }

        // Search for the button pressed until end of array or key found
        for (int i=0; i<jbnButtons.length; i++)
        {
            if(e.getSource() == jbnButtons[i])
            {
                switch(i)
                {
                    case 0:
                        addDigitToDisplay(i);
                        break;

                    case 1:
                        addDigitToDisplay(i);
                        break;

                    case 2:
```

```
            addDigitToDisplay(i);
            break;

        case 3:
            addDigitToDisplay(i);
            break;

        case 4:
            addDigitToDisplay(i);
            break;

        case 5:
            addDigitToDisplay(i);
            break;

        case 6:
            addDigitToDisplay(i);
            break;

        case 7:
            addDigitToDisplay(i);
            break;

        case 8:
            addDigitToDisplay(i);
            break;

        case 9:
            addDigitToDisplay(i);
            break;

        case 10:        // +/-
            processSignChange();
            break;

        case 11:        // decimal point
            addDecimalPoint();
            break;

        case 12:        // =
            processEquals();
            break;

        case 13:        // divide
            processOperator("/");
            break;
```

```
                        case 14:      // *
                              processOperator("*");
                              break;

                        case 15:      // -
                              processOperator("-");
                              break;

                        case 16:      // +
                              processOperator("+");
                              break;

                        case 17:      // sqrt
                              if (displayMode != ERROR_MODE)
                              {
                                try
                                    {
                                          if (getDisplayString().indexOf("-")
== 0)
                                             displayError("Invalid input for
function!");

                                          result =
Math.sqrt(getNumberInDisplay());
                                          displayResult(result);
                                    }

                                    catch(Exception ex)
                                    {
                                          displayError("Invalid input for
function!");
                                          displayMode = ERROR_MODE;
                                    }
                              }
                              break;

                        case 18:      // 1/x
                              if (displayMode != ERROR_MODE){
                                    try
                                    {
                                          if (getNumberInDisplay() == 0)
                                                displayError("Cannot divide by
zero!");

                                          result = 1 / getNumberInDisplay();
                                          displayResult(result);
                                    }
```

```
                                    catch(Exception ex)     {
                                        displayError("Cannot divide by
zero!");
                                        displayMode = ERROR_MODE;
                                    }
                                }
                                break;

                            case 19:        // %
                                if (displayMode != ERROR_MODE){
                                    try     {
                                        result = getNumberInDisplay() /
100;
                                        displayResult(result);
                                    }

                                    catch(Exception ex)     {
                                        displayError("Invalid input for
function!");
                                        displayMode = ERROR_MODE;
                                    }
                                }
                                break;

                            case 20:        // backspace
                                if (displayMode != ERROR_MODE){

setDisplayString(getDisplayString().substring(0,
                                                    getDisplayString().length() -
1));

                                    if (getDisplayString().length() < 1)
                                        setDisplayString("0");
                                }
                                break;

                            case 21:        // CE
                                clearExisting();
                                break;

                            case 22:        // C
                                clearAll();
                                break;
                        }
                    }
                }
```

```
	}

	void setDisplayString(String s){
		jlbOutput.setText(s);
	}

	String getDisplayString (){
		return jlbOutput.getText();
	}

	void addDigitToDisplay(int digit){
		if (clearOnNextDigit)
			setDisplayString("");

		String inputString = getDisplayString();

		if (inputString.indexOf("0") == 0){
			inputString = inputString.substring(1);
		}

		if ((!inputString.equals("0") || digit > 0)
					&& inputString.length() <
MAX_INPUT_LENGTH){
			setDisplayString(inputString + digit);
		}


		displayMode = INPUT_MODE;
		clearOnNextDigit = false;
	}

	void addDecimalPoint(){
		displayMode = INPUT_MODE;

		if (clearOnNextDigit)
			setDisplayString("");

		String inputString = getDisplayString();

		// If the input string already contains a decimal point, don't
		//  do anything to it.
		if (inputString.indexOf(".") < 0)
			setDisplayString(new String(inputString + "."));
	}

	void processSignChange(){
		if (displayMode == INPUT_MODE)
```

```
	{
		String input = getDisplayString();

		if (input.length() > 0 && !input.equals("0"))
		{
			if (input.indexOf("-") == 0)
				setDisplayString(input.substring(1));

			else
				setDisplayString("-" + input);
		}

	}

	else if (displayMode == RESULT_MODE)
	{
		double numberInDisplay = getNumberInDisplay();

		if (numberInDisplay != 0)
			displayResult(-numberInDisplay);
	}
}

void clearAll() {
	setDisplayString("0");
	lastOperator = "0";
	lastNumber = 0;
	displayMode = INPUT_MODE;
	clearOnNextDigit = true;
}

void clearExisting(){
	setDisplayString("0");
	clearOnNextDigit = true;
	displayMode = INPUT_MODE;
}

double getNumberInDisplay()	{
	String input = jlbOutput.getText();
	return Double.parseDouble(input);
}

void processOperator(String op) {
	if (displayMode != ERROR_MODE)
	{
		double numberInDisplay = getNumberInDisplay();
```

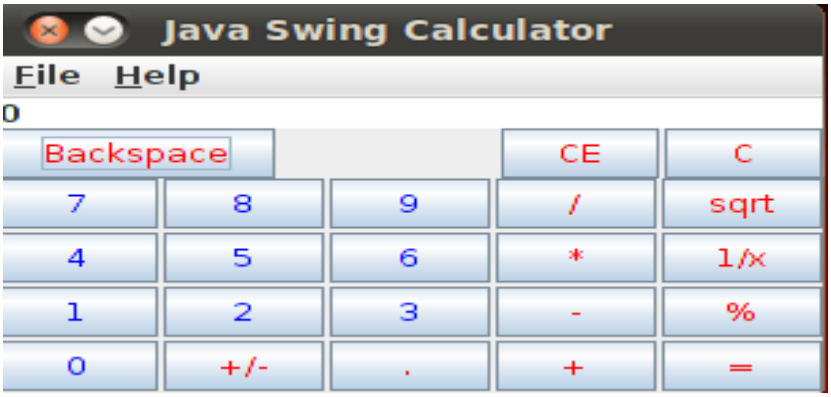
```
            if (!lastOperator.equals("0"))
            {
                  try
                  {
                        double result = processLastOperator();
                        displayResult(result);
                        lastNumber = result;
                  }

                  catch (DivideByZeroException e)
                  {
                  }
            }

            else
            {
                  lastNumber = numberInDisplay;
            }

            clearOnNextDigit = true;
            lastOperator = op;
      }
}

void processEquals(){
      double result = 0;

      if (displayMode != ERROR_MODE){
            try
            {
                  result = processLastOperator();
                  displayResult(result);
            }

            catch (DivideByZeroException e) {
                  displayError("Cannot divide by zero!");
            }

            lastOperator = "0";
      }
}

double processLastOperator() throws DivideByZeroException {
      double result = 0;
      double numberInDisplay = getNumberInDisplay();

      if (lastOperator.equals("/"))
```

```
            {
                if (numberInDisplay == 0)
                    throw (new DivideByZeroException());

                result = lastNumber / numberInDisplay;
            }

            if (lastOperator.equals("*"))
                result = lastNumber * numberInDisplay;

            if (lastOperator.equals("-"))
                result = lastNumber - numberInDisplay;

            if (lastOperator.equals("+"))
                result = lastNumber + numberInDisplay;

            return result;
        }

        void displayResult(double result){
            setDisplayString(Double.toString(result));
            lastNumber = result;
            displayMode = RESULT_MODE;
            clearOnNextDigit = true;
        }

        void displayError(String errorMessage){
            setDisplayString(errorMessage);
            lastNumber = 0;
            displayMode = ERROR_MODE;
            clearOnNextDigit = true;
        }

        public static void main(String args[]) {
            Calculator calci = new Calculator();
            Container contentPane = calci.getContentPane();
//          contentPane.setLayout(new BorderLayout());
            calci.setTitle("Java Swing Calculator");
            calci.setSize(241, 217);
            calci.pack();
            calci.setLocation(400, 250);
            calci.setVisible(true);
            calci.setResizable(false);
        }

    }           //End of Swing Calculator Class.
```

```
class DivideByZeroException extends Exception{
	public DivideByZeroException()
	{
		super();
	}

	public DivideByZeroException(String s)
	{
		super(s);
	}
}

class CustomABOUTDialog extends JDialog implements ActionListener {
	JButton jbnOk;

	CustomABOUTDialog(JFrame parent, String title, boolean modal){
		super(parent, title, modal);
		setBackground(Color.black);

		JPanel p1 = new JPanel(new FlowLayout(FlowLayout.CENTER));

		StringBuffer text = new StringBuffer();
		text.append("Calculator Information\n\n");
		text.append("Developer: Hemanth\n");
		text.append("Version:   1.0");

		JTextArea jtAreaAbout = new JTextArea(5, 21);
		jtAreaAbout.setText(text.toString());
		jtAreaAbout.setFont(new Font("Times New Roman", 1, 13));
		jtAreaAbout.setEditable(false);

		p1.add(jtAreaAbout);
		p1.setBackground(Color.red);
		getContentPane().add(p1, BorderLayout.CENTER);

		JPanel p2 = new JPanel(new FlowLayout(FlowLayout.CENTER));
		jbnOk = new JButton(" OK ");
		jbnOk.addActionListener(this);

		p2.add(jbnOk);
		getContentPane().add(p2, BorderLayout.SOUTH);

		setLocation(408, 270);
		setResizable(false);

		addWindowListener(new WindowAdapter() {
			public void windowClosing(WindowEvent e)
```

```
                        {
                                Window aboutDialog = e.getWindow();
                                aboutDialog.dispose();
                        }
                }
        );

        pack();
    }

    public void actionPerformed(ActionEvent e)
    {
        if(e.getSource() == jbnOk)      {
                this.dispose();
        }
    }

}
```

Hasil Running:



kesimpulan:

jika kita tidak teliti dalam membuat program ini maka program yang akan kita run akan mengalami masalah.