

Interfața grafică cu utilizatorul - AWT

Programare Orientată pe Obiecte



Interfața grafică cu utilizatorul



- Modelul AWT
- Componentele AWT
- Gestionarea poziționării
- Gruparea componentelor
- Folosirea ferestrelor

GUI – Interfața grafică cu utilizatorul

Comunicarea vizuală între un program și utilizatori.

- AWT(Abstract Windowing Toolkit)
- Swing - parte din JFC (Java Foundation Classes) Sun, Netscape și IBM

Etapele creării unei aplicații:

- Design
 - Crearea unei suprafețe de afișare
 - Crearea și asezarea componentelor
- Funcționalitate
 - Definirea unor acțiuni
 - "Ascultarea" evenimentelor

Modelul AWT



- Pachete:
java.awt, java.awt.event
- Obiectele grafice:
Component, MenuComponent.
- Suprafețe de afișare:
Container
- Gestionari de poziționare:
LayoutManager
- Evenimente:
AWTEvent

Exemplu: O fereastră cu două butoane

```
import java . awt . * ;
public class ExempluAWT1 {
    public static void main ( String args []) {
        // Crearea ferestrei - un obiect de tip Frame
        Frame f = new Frame ("O fereastră ");
        // Setarea modului de dipunere a componentelor
        f . setLayout (new FlowLayout ());
        // Crearea celor doua butoane
        Button b1 = new Button ("OK");
        Button b2 = new Button (" Cancel ");
        // Adaugarea butoanelor
        f . add (b1);
        f . add (b2);
        f . pack ();
        // Afisarea ferestrei
        f . show ();
    }
}
```

Componentele AWT

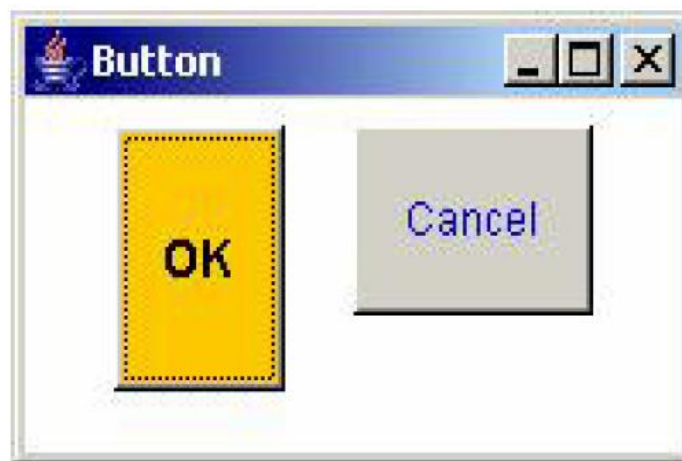
- orice obiect care poate avea o reprezentare grafică și care poate interacționa cu utilizatorul.
- Button
- Canvas
- Checkbox, CheckBoxLayout;
- Choice
- Container
- Label
- List
- Scrollbar
- TextComponent
 - TextField
 - TextArea

Componente grafice

Clasa Label



Clasa Button



Componente grafice

Clasa Checkbox



Clasa CheckboxGroup



Componente grafice

Clasa Choice

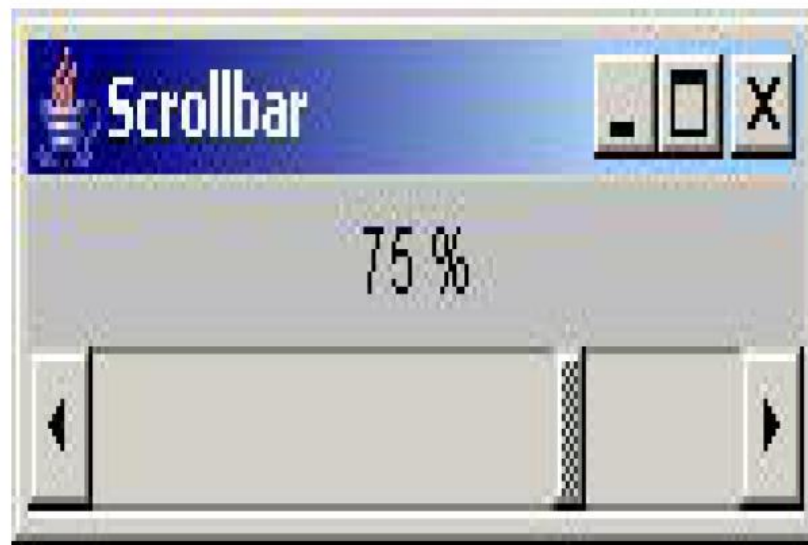


Clasa List



Componente grafice

Clasa ScrollBar

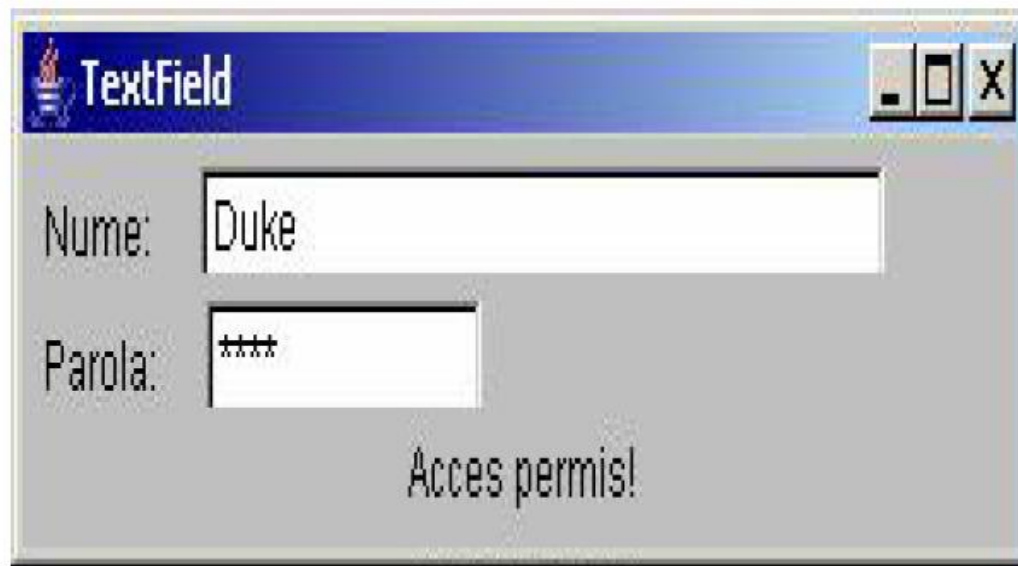


Clasa ScrollPane



Componente grafice

Clasa TextField



Clasa TextArea



Proprietăți comune

- Poziție
 - getLocation, getX, getY, getLocationOnScreen
 - setLocation, setX, setY
- Dimensiuni
 - getSize, getHeight, getWidth
 - setSize, setHeight, setWidth
- Dimensiuni și poziție
 - getBounds
 - setBounds
- Culoare (text și fundal)
 - getForeground, getBackground
 - setForeground, setBackground
- Font
 - getFont
 - setFont
- Vizibilitate
 - setVisible
 - isVisible
- Interactivitate
 - setEnabled
 - isEnabled

Suprafețe de afișare

- un container este folosit pentru a adăuga componente pe suprafața lui
- Container – superclasa pentru suprafețe de afișare
 - Window
 - Frame - ferestre standard
 - Dialog - ferestre de dialog
 - Panel, Applet
 - ScrollPane - derulare

Metode comune:

- add
- remove
- setLayout
- getInsets
- validate

Exemplu

```
Frame f = new Frame("O fereastră");  
// Adaugam un buton direct pe fereastră  
Button b = new Button("Hello");  
f.add(b);  
// Adaugam doua componente pe un panel  
Label et = new Label("Nume:");  
TextField text = new TextField();  
Panel panel = new Panel();  
panel.add(et);  
panel.add(text);  
// Adaugam panel-ul pe fereastră  
// si, indirect, cele doua componente  
f.add(panel);
```

Gestionarea poziționării

Exemplu: Poziționarea a 5 butoane

```
import java . awt . * ;  
public class TestLayout {  
    public static void main ( String args []) {  
        Frame f = new Frame (" Grid Layout ");  
        f. setLayout (new GridLayout (3, 2));  
        Button b1 = new Button (" Button 1");  
        Button b2 = new Button ("2");  
        Button b3 = new Button (" Button 3");  
        Button b4 = new Button ("Long - Named Button 4");  
        Button b5 = new Button (" Button 5");  
        f.add(b1); f.add (b2); f. add(b3); f.add(b4); f.add(b5);  
        f. pack ();  
        f. show ();  
    }  
}
```

Gestionarea poziționării



```
Frame f = new Frame("Flow Layout");  
f.setLayout(new FlowLayout());
```



Gestionarea poziționării (2)

- Un gestionar de poziționare (layout manager) este un obiect care controlează dimensiunea și aranjarea (poziția) componentelor unui container.
- Fiecare obiect de tip Container are asociat un gestionar de poziționare.
- Toate clasele care instanțiază obiecte pentru gestionarea poziționării implementează interfața `LayoutManager`.
- La instanțierea unui container se creează implicit un gestionar de poziționare asociat acestuia. (ferestre: `BorderLayout`, panel-uri: `FlowLayout`).

Folosirea gestionarilor de poziționare

- Gestionari AWT

FlowLayout, BorderLayout, GridLayout, CardLayout, GridBagLayout

- Metoda setLayout

FlowLayout gestionar = new FlowLayout();

container.setLayout(gestionar); // sau:

container.setLayout(new FlowLayout());

- Dimensionarea

getPreferredSize, getMinimumSize,
getMaximumSize

- Poziționarea absolută

container.setLayout(null);

Button b = new Button("Buton");

b.setSize(10, 10); b.setLocation (0, 0);

b.add();

Gestionarul FlowLayout

```
import java.awt.*;
public class TestFlowLayout {
    public static void main(String args[]) {
        Frame f = new Frame("Flow Layout");
        f.setLayout(new FlowLayout());

        Button b1 = new Button("Button 1");
        Button b2 = new Button("2");
        Button b3 = new Button("Button 3");
        Button b4 = new Button("Long-Named Button 4");
        Button b5 = new Button("Button 5");

        f.add(b1); f.add(b2); f.add(b3); f.add(b4); f.add(b5);
        f.pack();
        f.show();
    }
}
```



Gestionar implicit pentru Panel.

Gestionarul BorderLayout

Cinci regiuni: NORTH, SOUTH, EAST, WEST, CENTER

Implicit pentru Window

```
import java.awt.*;
public class TestBorderLayout {
    public static void main(String args[]) {
        Frame f = new Frame("Border Layout");
        // Apelul de mai jos poate sa lipseasca
        f.setLayout(new BorderLayout());

        f.add(new Button("Nord"), BorderLayout.NORTH);
        f.add(new Button("Sud"), BorderLayout.SOUTH);
        f.add(new Button("Est"), BorderLayout.EAST);
        f.add(new Button("Vest"), BorderLayout.WEST);
        f.add(new Button("Centru"), BorderLayout.CENTER);
        f.pack();

        f.show();
    }
}
```

Cele cinci butoane ale ferestrei vor fi afișate astfel:



Gestionarul GridLayout

Organizare tabelară

```
import java.awt.*;  
public class TestGridLayout {  
    public static void main(String args[]) {  
        Frame f = new Frame("Grid Layout");  
        f.setLayout(new GridLayout(3, 2));  
  
        f.add(new Button("1"));  
        f.add(new Button("2"));  
        f.add(new Button("3"));  
        f.add(new Button("4"));  
        f.add(new Button("5"));  
        f.add(new Button("6"));  
  
        f.pack();  
        f.show();  
    }  
}
```



Gestionarul CardLayout

Pachet de cărți

Prima "carte" este vizibilă



A doua "carte" este vizibilă

Swing: JTabbedPane



Gestionarul GridBagLayout

```
GridBagLayout gridBag = new GridBagLayout();  
container.setLayout(gridBag);  
GridBagConstraints c = new GridBagConstraints();  
//Specificam restrictiile. . .și apoi  
gridBag.setConstraints(componenta, c);  
container.add(componenta);
```



The screenshot shows a Java Swing window titled "Test GridBagLayout". Inside the window, there is a yellow header bar with the text "Evidenta persoane". Below the header, there are two input fields: "Nume:" and "Salariu:". To the right of the "Nume:" field is a button labeled "Adaugare". Below the "Salariu:" field are two buttons: "Salvare" and "Iesire". The layout is a simple form with a grid-like structure.

Gestionarul GridBagLayout (2)

- Cele mai utilizate tipuri de constrângeri pot fi specificate prin intermediul următoarelor variabile din clasa GridBagConstraints:
- **gridx, gridy** - celula ce reprezintă colțul stânga sus al componentei;
- **gridwidth, gridheight** - numărul de celule pe linie și coloană pe care va fi afișată componenta;
- **fill** - folosită pentru a specifica dacă o componentă va ocupa întreg spațiul pe care îl are destinat; valorile posibile sunt HORIZONTAL, VERTICAL, BOTH, NONE;
- **insets** - distanțele dintre componentă și marginile suprafeței sale de afișare;
- **anchor** - folosită atunci când componenta este mai mică decât suprafața sa de afișare pentru a forța o anumită dispunere a sa: nord, sud, est, vest, etc.
- **weightx, weighty** - folosite pentru distribuția spațiului liber; uzual au valoarea 1;

Gruparea componentelor

- Gruparea componentelor se face folosind clasa Panel.
- Un panel este cel mai simplu model de container.
- Nu are o reprezentare vizibilă, rolul său fiind de a oferi o suprafață de afișare pentru componente grafice, inclusiv pentru alte panel-uri.
- Aranjare eficientă a componentelor unei ferestre presupune:
 - gruparea componentelor "înfrățite";
 - aranjarea componentelor unui panel;
 - aranjarea panel-urilor pe suprafața ferestrei.
- Gestionarul implicit este FlowLayout.

Gruparea componentelor

```
import java.awt.*;
public class TestPanel {
    public static void main(String args[]) {
        Frame f = new Frame("Test Panel");

        Panel intro = new Panel();
        intro.setLayout(new GridLayout(1, 3));
        intro.add(new Label("Text:"));
        intro.add(new TextField("", 20));
        intro.add(new Button("Adaugare"));

        Panel lista = new Panel();
        lista.setLayout(new FlowLayout());
        lista.add(new List(10));
        lista.add(new Button("Stergere"));

        Panel control = new Panel();
        control.add(new Button("Salvare"));
        control.add(new Button("Iesire"));

        f.add(intro, BorderLayout.NORTH);
        f.add(lista, BorderLayout.CENTER);
        f.add(control, BorderLayout.SOUTH);

        f.pack();
        f.show();
    }
}
```

Folosirea ferestrelor

- Window: Frame, Dialog
- Gestionar de poziționare: BorderLayout.
- Window: crearea de ferestre care nu au chenar și nici bară de meniuri; nu interacționează cu utilizatorul ci doar oferă anumite informații.
- Metode:
 - show - face vizibilă fereastra. Implicit, o fereastră nou creată nu este vizibilă;
 - hide - face fereastra invizibilă fără a o distruge însă; pentru a redeveni vizibilă se poate apela metoda show;
 - isShowing - testează dacă fereastra este vizibilă sau nu;
 - dispose - închide fereastra și eliberează toate resursele acesteia;
 - pack - redimensionează automat fereastra la o suprafață optimă care să cuprindă toate componentele sale; trebuie apelată în general după adăugarea tuturor componentelor pe suprafața ferestrei.

Clasa Frame

```
import java.awt.*;
public class TestFrame {
    public static void main(String args[]) {
        Frame f = new Frame("Titlul ferestrei");
        f.show();
    }
}
import java.awt.*;
class Fereastra extends Frame{
// Constructorul
    public Fereastra(String titlu) {
        super(titlu);
        ...
    }
}
...
Fereastra f = new Fereastra("Titlul ferestrei");
f.show();
```

Clasa Dialog

- o fereastră de dialog este dependentă de o altă fereastră (normală sau tot fereastră de dialog), numită și fereastră părinte
- ferestrele de dialog pot fi:
 - **modale**: care blochează accesul la fereastra părinte în momentul deschiderii lor,
 - **nemodale**: care nu blochează fluxul de intrare către fereastra părinte

Constructorii:

- Dialog(Frame parinte)
- Dialog(Frame parinte, String titlu)
- Dialog(Frame parinte, String titlu, boolean modala)
- Dialog(Frame parinte, boolean modala)
- Dialog(Dialog parinte)
- Dialog(Dialog parinte, String titlu)
- Dialog(Dialog parinte, String titlu, boolean modala)

Clasa FileDialog

- fereastră de dialog folosită pentru selectarea unui nume de fișier în vederea încărcării sau salvării unui fișier

Constructori

- FileDialog(Frame parinte)
- FileDialog(Frame parinte, String titlu)
- FileDialog(Frame parinte, String titlu, boolean mod)

// Dialog pentru incarcarea unui fisier

- new FileDialog(parinte, "Alegere fisier", FileDialog.LOAD);

// Dialog pentru salvarea unui fisier

- new FileDialog(parinte, "Salvare fisier", FileDialog.SAVE);
- Swing: JFileChooser

Interfața grafică cu utilizatorul - Swing

Programare Orientată pe Obiecte



Swing



- JFC (Java Foundation Classes)
- Componentele Swing
- Asemănări și deosebiri cu AWT
- Folosirea ferestrelor
- Look and Feel

JFC (Java Foundation Classes)

- Componente Swing

Sunt componente ce înlocuiesc și în același timp extind vechiul set oferit de modelul AWT.

- Look-and-Feel

Permite schimbarea înfățișării și a modului de interacțiune cu aplicația în funcție de preferințe

- Accessibility API

Permite dezvoltarea de aplicații care să comunice cu dispozitive utilizate de către persoane cu diverse tipuri de handicap.

- Java 2D API

Permite crearea de aplicații care utilizează grafică la un nivel avansat. Clasele puse la dispoziție permit crearea de desene complexe, efectuarea de operații geometrice (rotiri, scalări, translații, etc.), prelucrarea de imagini, tipărire, etc.

- Drag-and-Drop

Oferă posibilitatea de a efectua operații drag-and-drop între aplicații Java și aplicații native.

- Internaționalizare

Permite dezvoltarea de aplicații care să poată fi configurate pentru exploatarea lor în diverse zone ale globului, utilizând limba și particularitățile legate de formatarea datei, numerelor sau a monedei din zona respectivă.

Swing API

- javax.accessibility
- javax.swing.plaf
- javax.swing.text.html
- javax.swing
- javax.swing.plaf.basic
- javax.swing.text.parser
- javax.swing.border
- javax.swing.plaf.metal
- javax.swing.text.rtf
- javax.swing.colorchooser
- javax.swing.plaf.multi
- javax.swing.tree
- javax.swing.event
- javax.swing.table
- javax.swing.undo
- javax.swing.filechooser
- javax.swing.text

Cel mai important: **javax.swing**

Componentele Swing

- **Componente atomice**

JLabel, JButton, JToggleButton (JCheckBox, JRadioButton), JScrollBar, JSlider, JProgressBar, JSeparator

- **Componente complexe**

JTable, JTree, JComboBox, JSpinner, JList, JFileChooser, JColorChooser, JOptionPane

- **Componente pentru editare de text**

JTextField, JFormattedTextField, JPasswordField, JTextArea, JEditorPane, JTextPane

- **Meniuri**

JMenuBar, JMenu, JPopupMenu, JMenuItem, JCheckboxMenuItem, JRadioButtonMenuItem

- **Containere intermediare**

JPanel, JScrollPane, JSplitPane, JTabbedPane, JDesktopPane, JToolBar

- **Containere de nivel înalt**

JFrame, JDialog, JWindow, JInternalFrame, JApplet

Asemănări și deosebiri cu AWT

- Tehnologia Swing extinde AWT.

```
import javax.swing.*;
```

```
import java.awt.*; //Font, Color, ...
```

```
import java.awt.event.*;
```

- Convenția "J"

```
java.awt.Button - javax.swing.JButton
```

```
java.awt.Label - javax.swing.JLabel, etc.
```

- Noi gestionari de poziționare: BorderLayout, SpringLayout

- Folosirea HTML

```
JButton simplu = new JButton("Text simplu");
```

```
JButton html = new JButton(
```

```
"<html><u>Text</u> <i>formatat</i></html>");
```

O aplicație simplă AWT

```
import java . awt . * ;
public class ExempluAWT extends Frame {
    public ExempluAWT ( String titlu ) {
        super ( titlu );
        setLayout ( new FlowLayout ( ) );
        add ( new Label ( " Hello AWT " ) );
        Button b = new Button ( " Close " );
        add ( b );
        pack ( );
        show ( );
    }
    public static void main ( String args [ ] ) {
        new ExempluAWT ( " Hello " );
    }
}
```

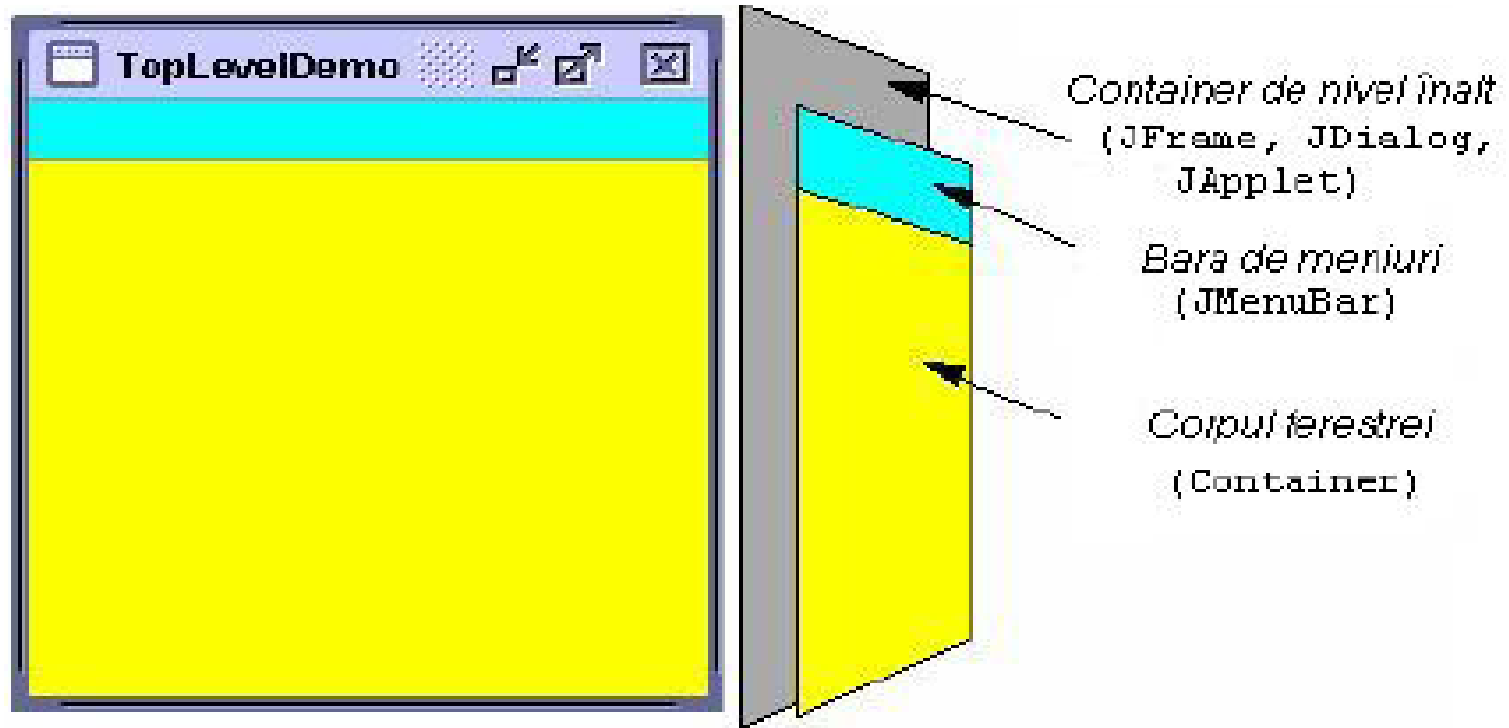
Aplicația rescrisă folosind Swing–1.4

```
import javax . swing . * ;
import java . awt . * ;
class ExempluSwing extends JFrame {
    public ExempluSwing ( String titlu ) {
        super ( titlu );
        // Metoda setLayout nu se aplica direct ferestrei
        getContentPane (). setLayout ( new FlowLayout () );
        // Componentele au denumiri ce incep cu litera J
        getContentPane (). add ( new JLabel ( "Swing" ) );
        JButton b = new JButton ( "Close" );
        // Metoda add nu se aplica direct ferestrei
        getContentPane (). add ( b );
        pack ();
        show ();
    }
    public static void main ( String args []) {
        new ExempluSwing ( " Hello " );
    }
}
```

Aplicația rescrisă folosind Swing–1.5

```
import javax . swing . * ;
import java . awt . * ;
class ExempluSwing extends JFrame {
    public ExempluSwing ( String titlu ) {
        super ( titlu );
        setLayout ( new FlowLayout ( ) );
        add( new JLabel ( "Swing" ) );
        JButton b = new JButton ( "Close" );
        add(b);
        pack ( );
        show ( );
    }
    public static void main ( String args [ ] ) {
        new ExempluSwing ( " Hello " );
    }
}
```

Folosirea ferestrelor



```
Frame f = new Frame();  
f.setLayout(new FlowLayout());  
f.add(new Button("OK"));
```

```
JFrame jf = new JFrame();
```

```
jf.getContentPane().setLayout(new FlowLayout()); //pana la vers 1.5
```

```
jf.getContentPane().add(new JButton("OK")); // //pana la vers 1.5
```

```
jf.setDefaultCloseOperation(WindowConstants.HIDE_ON_CLOSE);
```

- WindowConstants.DO_NOTHING_ON_CLOSE
- JFrame.EXIT_ON_CLOSE

Look and Feel

- Modul în care sunt desenate componentele Swing și felul în care acestea interacționează cu utilizatorul
- Același program poate utiliza diverse moduri Look-and-Feel, cum ar fi cele standard Windows, Mac, Java, Motif sau altele oferite de diverși dezvoltatori
- `javax.swing.plaf.metal.MetalLookAndFeel`
Varianta implicită de L&F și are un aspect specific **Java**.
- `com.sun.java.swing.plaf.windows.WindowsLookAndFeel`
Varianta specifică sistemelor de operare **Windows**. Incepând cu versiunea 1.4.2 există și implementarea pentru Windows XP .
- `com.sun.java.swing.plaf.mac.MacLookAndFeel`
Varianta specifică sistemelor de operare **Mac**.
- `com.sun.java.swing.plaf.motif.MotifLookAndFeel`
Specifică interfața **CDE/Motif**.
- `com.sun.java.swing.plaf.gtk.GTKLookAndFeel`
GTK+ reprezintă un standard de creare a interfețelor grafice dezvoltat independent de limbajul Java. (GTK este acronimul de la GNU ImageManipulation Program Toolkit).

Specificare unei anumite interfețe L&F

- **Folosirea clasei UIManager (metode statice):**
 - `getLookAndFeel` - Obține varianta curentă, returnând un obiect de tip `LookAndFeel`.
 - `setLookAndFeel` - Setează modul curent L&F. Metoda primește ca argument un obiect dintr-o clasă derivată din `LookAndFeel`, fie un șir de caractere cu numele complet al clasei L&F.
 - `getSystemLookAndFeelClassName` - Obține variantă specifică sistemului de operare folosit. În cazul în care nu există nici o astfel de clasă, returnează varianta standard.
 - `getCrossPlatformLookAndFeelClassName` - Returnează interfața grafică standard Java (JLF).

// Example:

```
UIManager.setLookAndFeel(  
    "com.sun.java.swing.plaf.motif.MotifLookAndFeel" );  
UIManager.setLookAndFeel (   
    UIManager.getSystemLookAndFeelClassName() );
```

Specificare unei anumite interfețe L&F

- **Setarea proprietății swing.defaultlaf**

1. direct de la linia de comandă prin setarea proprietății swing.defaultlaf:

```
java -Dswing.defaultlaf =  
com.sun.java.swing.plaf.gtk.GTKLookAndFeel App
```

2. În lib/swing.properties:

```
# Swing properties  
swing.defaultlaf =  
com.sun.java.swing.plaf.windows.WindowsLookAndFeel
```

- Există posibilitatea de a schimba varianta de L&F chiar și după afișarea componentelor. Acesta este un proces care trebuie să actualizeze ierarhiile de componente:

```
UIManager.setLookAndFeel(numeClasaLF);  
SwingUtilities.updateComponentTreeUI(f);  
f.pack();
```