2011 - 2012

Programare **O**rientata spre **O**biecte
(*Object-Oriented Programming*)

a.k.a. Programare Obiect-Orientata

Titular curs: Eduard-Cristian Popovici

Suport curs *http://discipline.elcom.pub.ro/POO-Java/*

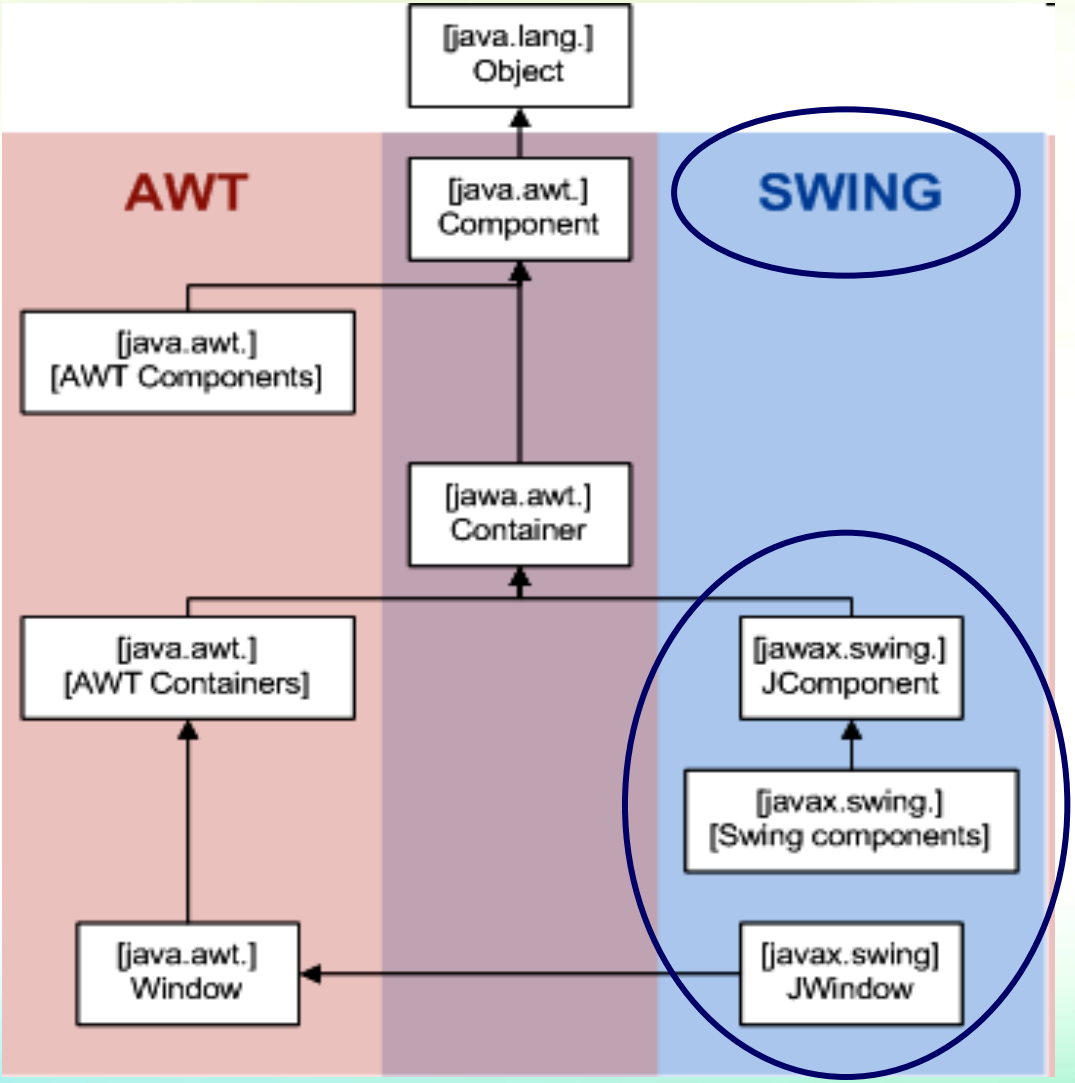
2. Orientarea spre obiecte in limbajul Java

2.7. Clase pentru interfete grafice (GUI) din Java Swing

Tutorialul Swing:

http://java.sun.com/docs/books/tutorial/uiswing/TOC.html

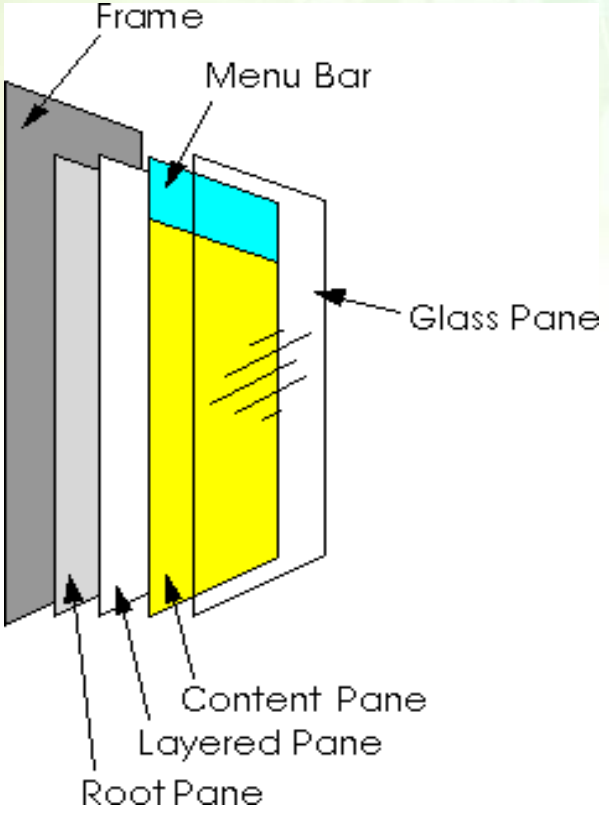
Ierarhiile de clase Java pentru interfete grafice

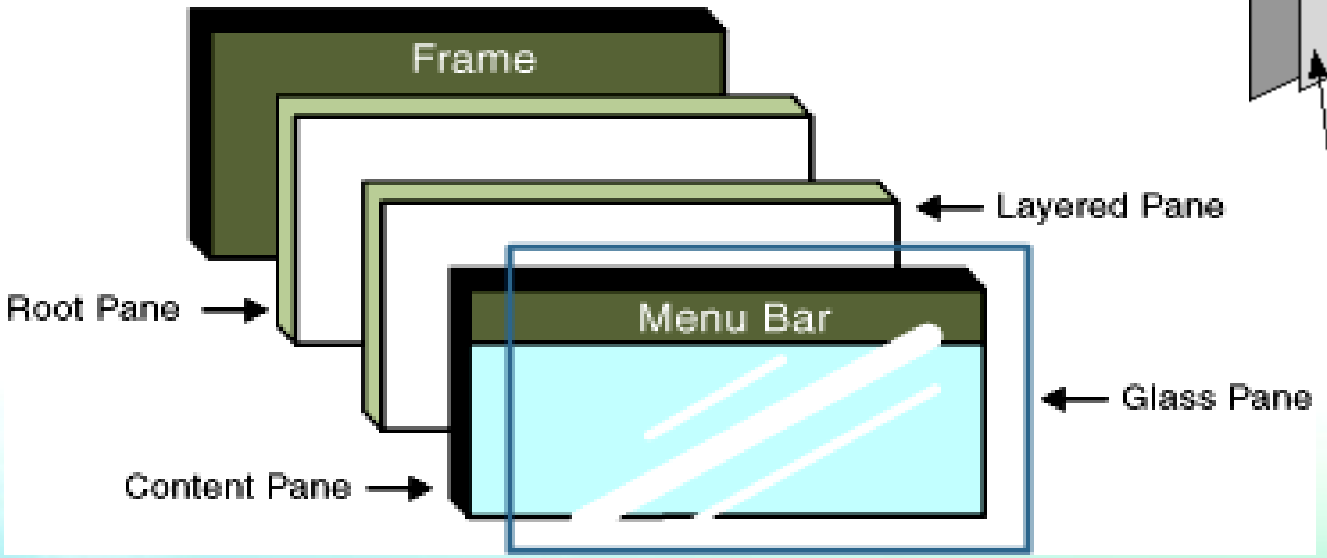


Containere de nivel maxim (*top-level*)

Fiecare aplicaţie care utilizează componente **Swing** are **cel puţin un "*top-level container*"**

- care **contine un container intermediar** numit ***Root Pane***
  - iar acesta la randul lui este **rădăcină a unei ierarhii de containere** care **conţin toate celelalte componente Swing**

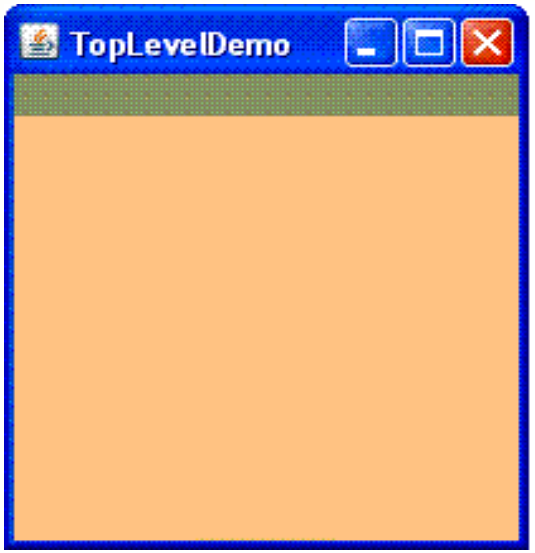


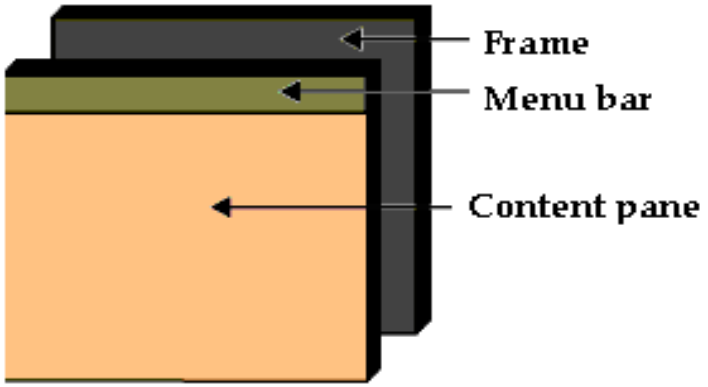


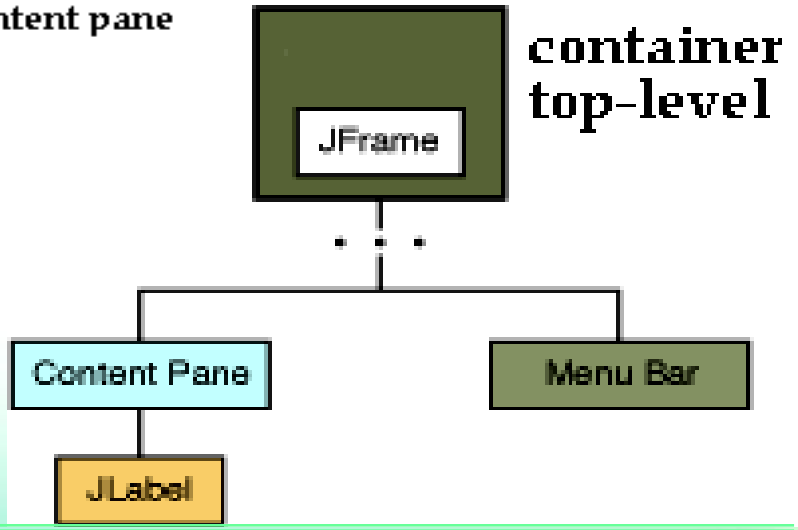
Containere de nivel maxim (*top-level*)

Containerele de nivel maxim (*top-level cantainers*)

- ferestrele principale – ***JFrame*** in Swing (*Frame* in AWT)
- ferestrele de dialog – ***JDialog*** in Swing (*Dialog* in AWT)
- miniaplicatiile Web – ***JApplet*** in Swing (*Applet* in AWT)



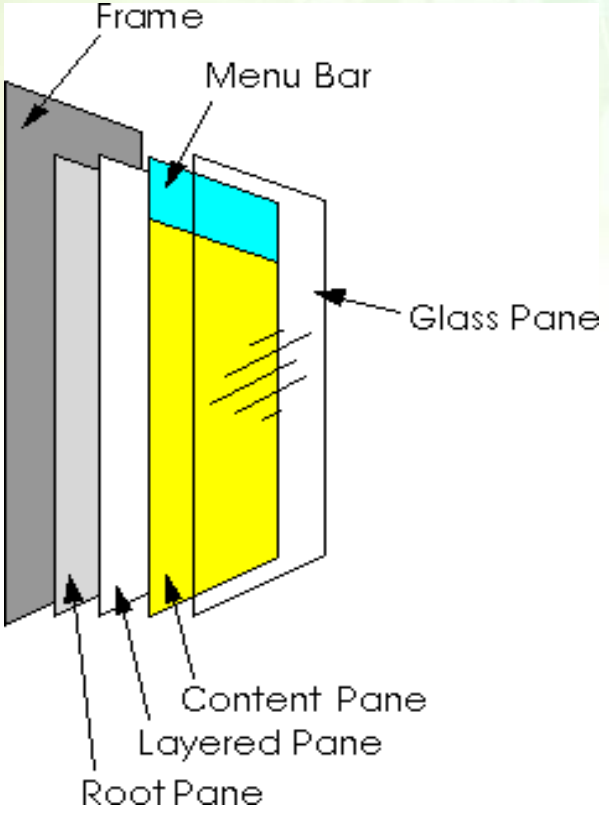


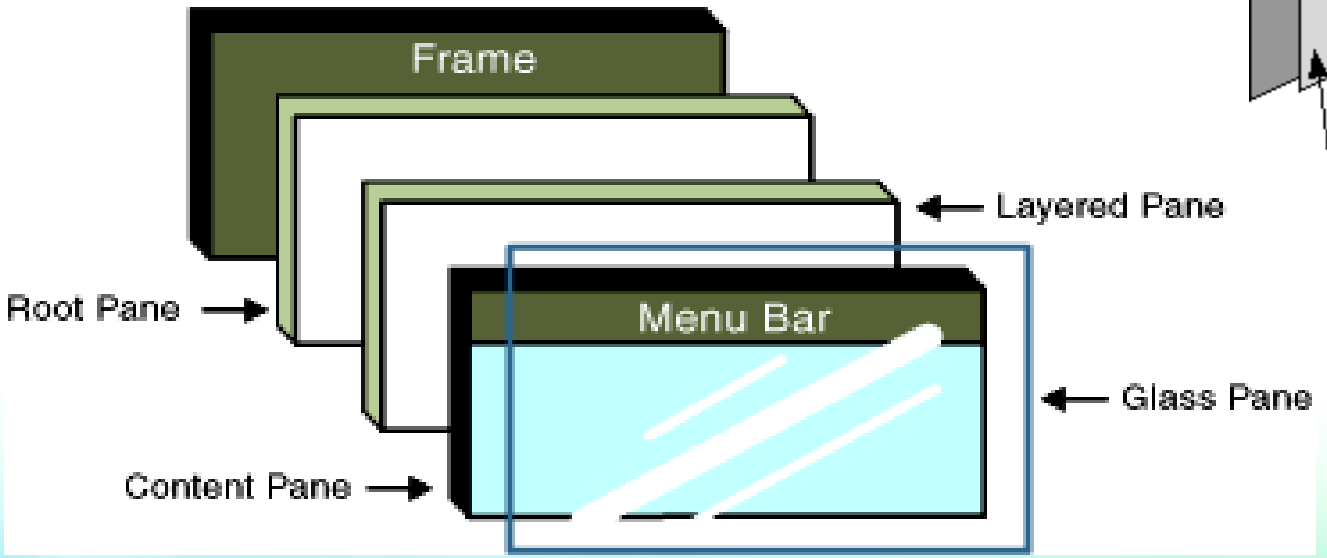


Containere de nivel maxim (*top-level*)

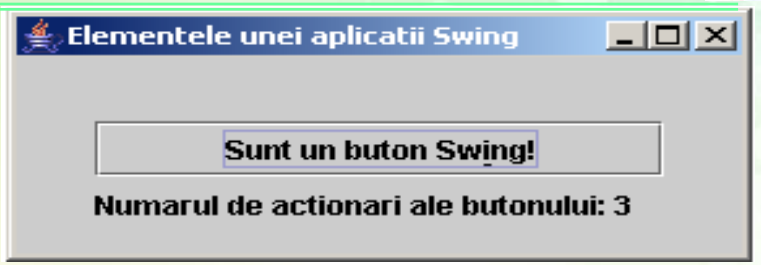
Containerul intermediar *Root Pane*

- contine **bara de meniu** (***Menu Bar***)
- **si un container de continut** numit ***Content Pane***
  - **in care programatorul poate adauga** de fapt **ierarhia de containere care contin restul componentelor** Swing



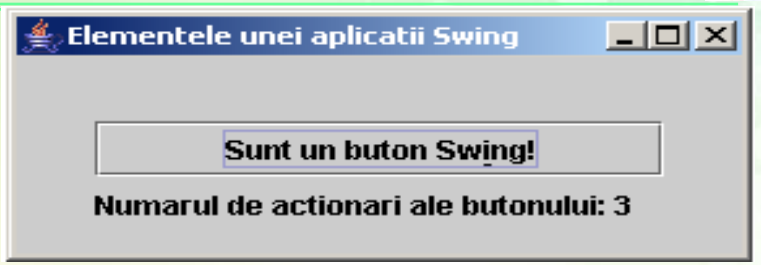


Elementele unei aplicatii grafice Swing




```
// 1. Importul pachetelor Swing si AWT necesare

import javax.swing.*;          // Numele actual al pachetului Swing (JFC)
import java.awt.*;             // Numele pachetului AWT (necesar uneori)
import java.awt.event.*;       // Numele pachetului pentru tratarea
                               // evenimentelor, pentru interactivitate

public class ElementeAplicatieSwing {

  public static void main(String[] args) {

    // 2. Optional: stabilirea aspectului (Java, Windows,...)

    try { UIManager.setLookAndFeel(
              UIManager.getCrossPlatformLookAndFeelClassName());
    } catch (Exception e) { }

    // 3. Stabilirea containerului de nivel maxim
    //    (JFrame, JApplet, JDialog)

    JFrame frame = new JFrame("Elementele unei aplicatii Swing");
```

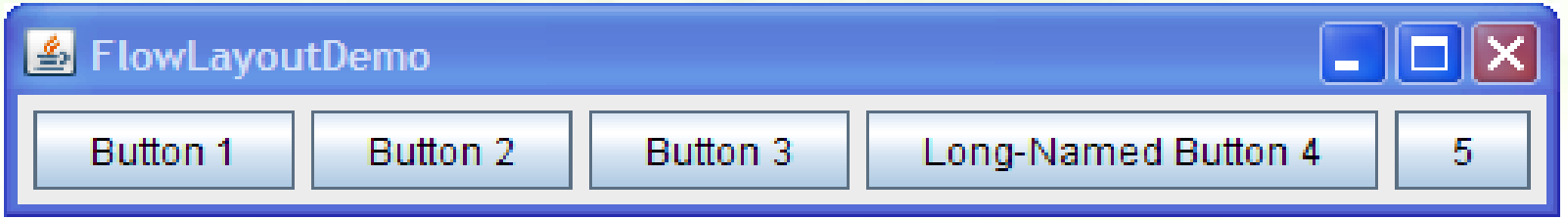
Elementele unei aplicatii grafice Swing



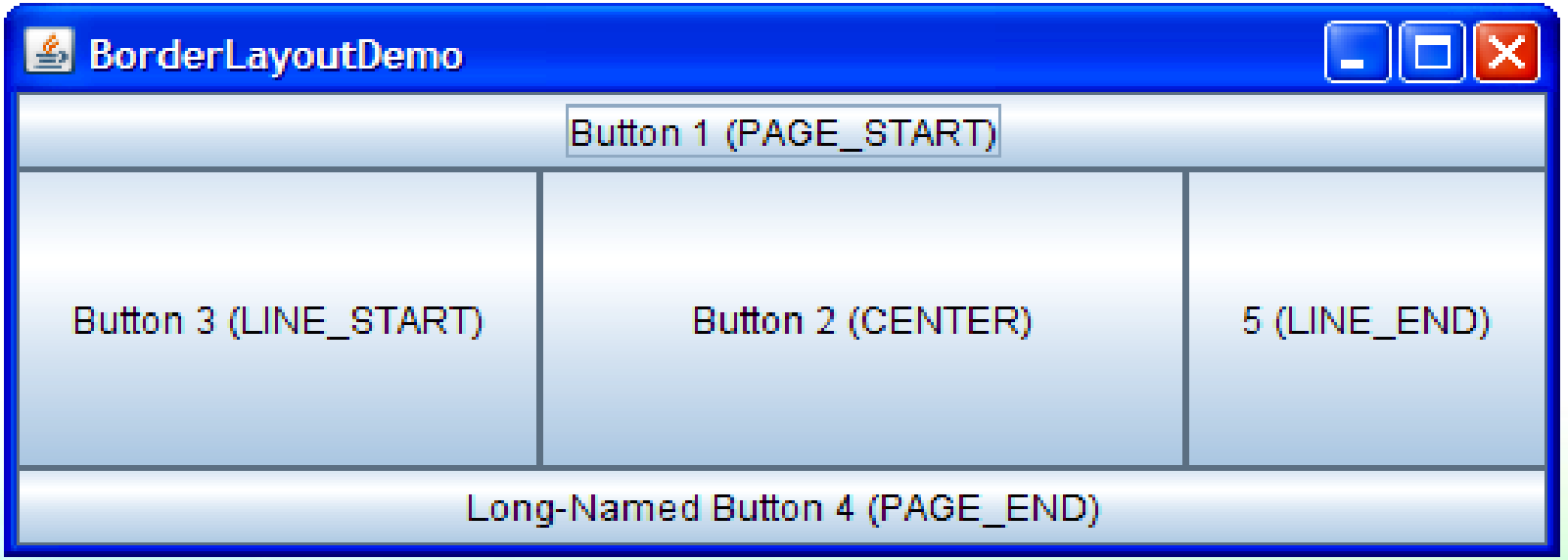

```
// 4. Obtinerea panoului de continut intern cadrului
//    (container in care vor fi plasate componentele ferestrei)
Container container = frame.getContentPane();

// 5. Optional: Stabilirea modului de asezare (layout)
//              in panoul ContentPane (implicit FlowLayout)
container.setLayout(new BorderLayout());

// 6. Crearea si configurarea componentelor grafice (controale)
// 6.1. Crearea unei etichete
final String textEticheta = "Numarul de actionari ale butonului: ";
final JLabel eticheta = new JLabel(textEticheta + "0    ");

// 6.2. Crearea unui buton cu combinatie de taste atasata
JButton buton = new JButton("Sunt un buton Swing!");
buton.setMnemonic(KeyEvent.VK_I); // optional
```

Moduri de asezare a componentelor in containere (*Layouts*)

Pe o linie (*FlowLayout*) - mod de asezare **implicit**

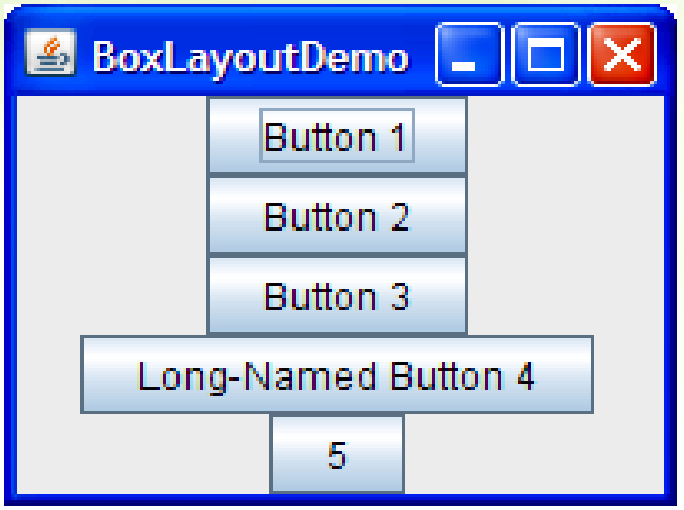


Relativ la margini (*BorderLayout*)

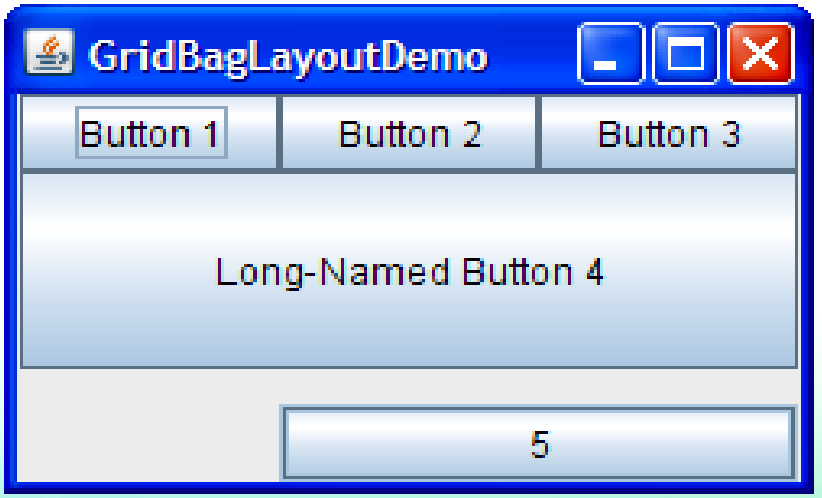


Moduri de asezare a componentelor in containere (*Layouts*)

Pe o coloana (*BoxLayout*)

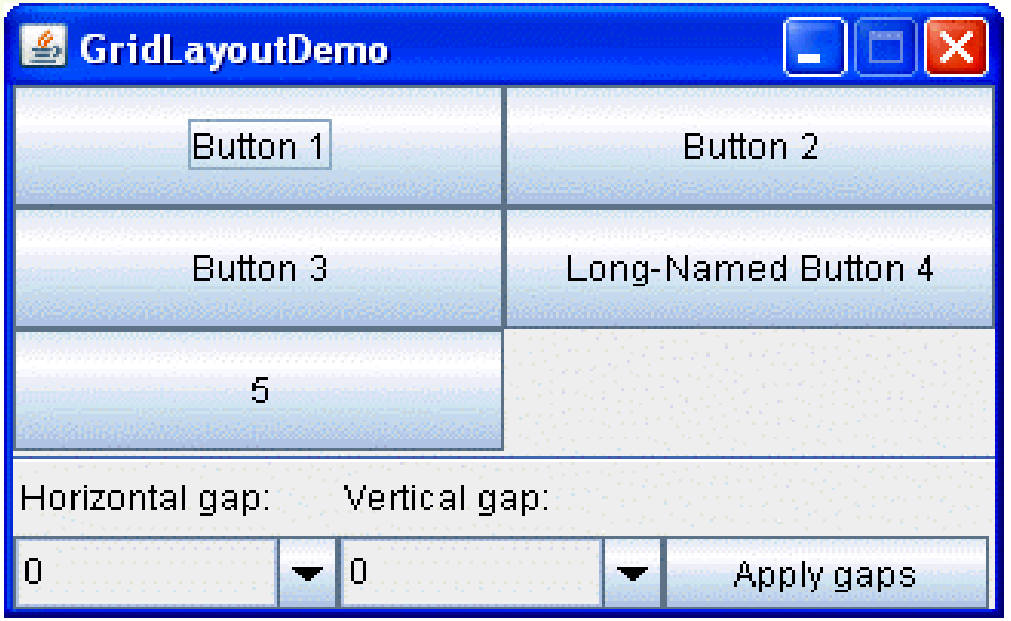


Grid pe coloane de latime diferita (*GridBagLayout*)



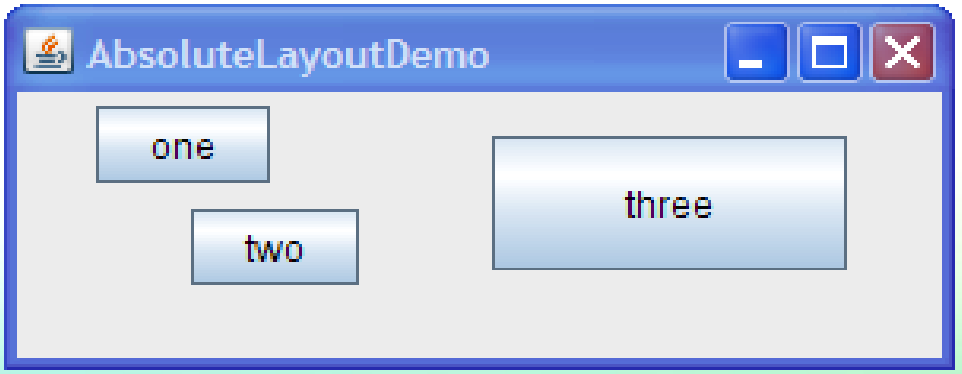
Moduri de asezare a componentelor in containere (*Layouts*)

Grid pe coloane de latime egala (*GridLayout*)

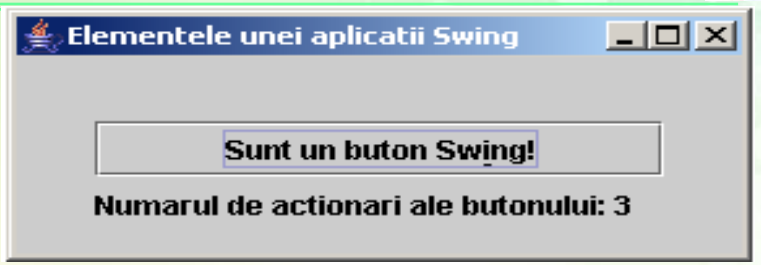


Asezarea fara LayoutManager

- utilizarea **pozitionarii absolute**

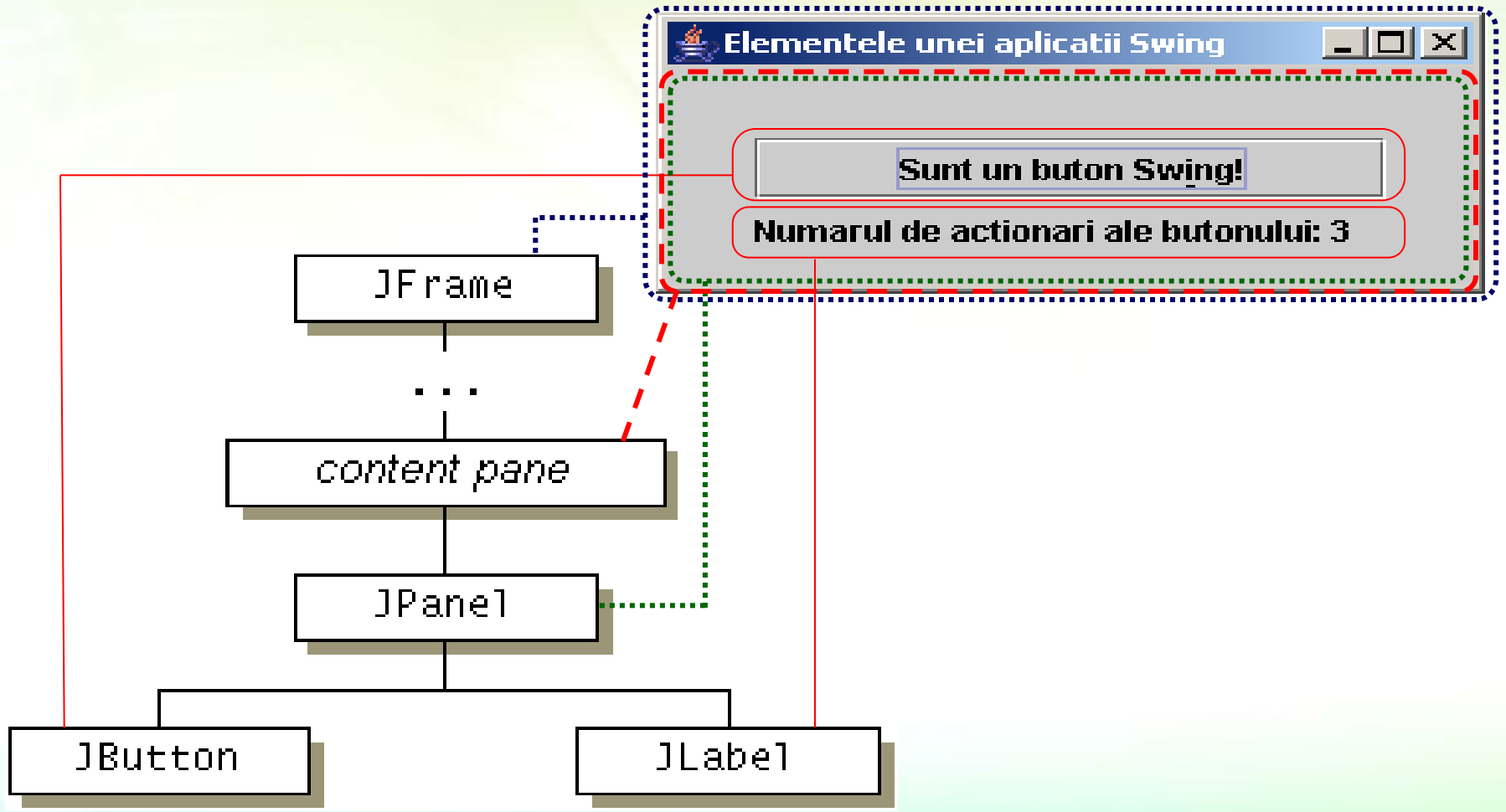


Elementele unei aplicatii grafice Swing

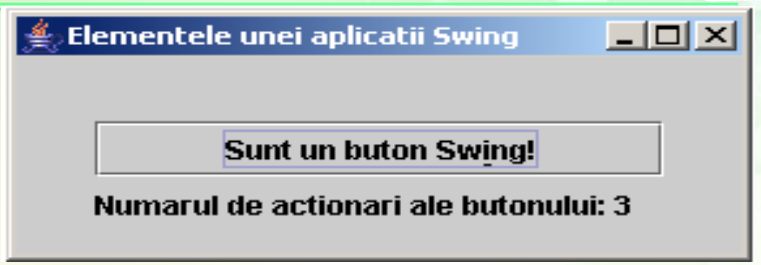



```
// 6.3. Crearea si configurarea unui nou panou (care permite
//      gruparea spatiala, ierarhica, a componentelor)
JPanel panou = new JPanel();

// 6.3.1. Stabilirea spatiului care inconjoara continutul
panou.setBorder(BorderFactory.createEmptyBorder(
                                30,    // sus
                                30,    // in stanga
                                10,    // jos
                                30));  // in dreapta

// 6.3.2. Stabilirea modului de asezare (layout) in noul panou
panou.setLayout(new GridLayout(0, 1));

// 6.3.3. Adaugarea componentelor in noul panou
panou.add(buton);
panou.add(eticheta);

 // 7. Adaugarea in ContentPane a componentelor grafice
container.add(panou, BorderLayout.CENTER);
```

Elementele unei aplicatii grafice Swing



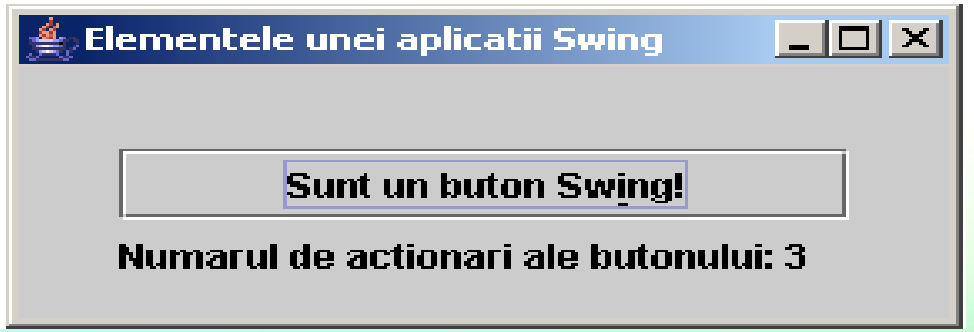
Elementele unei aplicatii grafice Swing




```
// 8. Crearea codului pentru tratarea evenimentelor
// (interactivitatii)

// 8.1. Atasarea unui obiect al unei clase anonime care
//      implementeaza interfata ActionListener, a carei metoda
//      actionPerformed() trateaza actionarea butonului

 buton.addActionListener( new ActionListener() {
   int numActionari = 0;
   public void actionPerformed(ActionEvent e) {
     numActionari++;
     eticheta.setText(textEticheta + numActionari);
   }
 });
```

Elementele unei aplicatii grafice Swing

```
        // 8.2. Stabilirea iesirii din program la inchiderea ferestrei
        frame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {  System.exit(0);  }
        });

        // Alternativa, incepand cu Java 2, versiunea 1.3:
        //        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // 9. Stabilirea dimensiunii ferestrei
        //    - fie in functie de cea a componentelor:
        frame.pack();

        //    - fie prin impunerea dimensiunii ferestrei:
        //      frame.setSize(400,     // latime
        //                    300);    // inaltime

        // 10. Afisarea ferestrei
        frame.setVisible(true);
    }
}
```



Elementele unei aplicatii grafice Swing

Tratarea evenimentelor – crearea inline a "ascultatoarelor"

```
// Atasarea unui obiect al unei clase anonime care
// implementeaza interfata ActionListener, a carei metoda
// actionPerformed() trateaza actionarea butonului

buton.addActionListener( new ActionListener() {
    int numActionari = 0;
    public void actionPerformed(ActionEvent e) {
      numActionari++;
      eticheta.setText(textEticheta + numActionari);
    }
});
```

Elementele unei aplicatii grafice Swing

Tratarea evenimentelor – transformarea clasei principale in "ascultator"

```
public class ElementeAplicatieSwing implements ActionListener {
   int numActionari = 0;
   public static void main(String[] args) {
      // ...
      // Atasarea unui obiect al clasei curente, care implementeaza
      // ActionListener, inclusiv metoda actionPerformed() de tratare
      buton.addActionListener( new ElementeAplicatieSwing() )
      // ...
   }
   public void actionPerformed(ActionEvent e) {
      numActionari++;
      eticheta.setText(textEticheta + numActionari);
   }
}
```

Elementele unei aplicatii grafice Swing

Tratarea evenimentelor – utilizarea altei clase "ascultator" – varianta compacta

```
public class ElementeAplicatieSwing {

   public static void main(String[] args) {
      // ...

      // Atasarea unui obiect al unei clasei ascultator,care implementeaza
      // ActionListener, inclusiv metoda actionPerformed() de tratare

      buton.addActionListener( new AscultatorActionare() )

      // ...
   }

class AscultatorActionare implements ActionListener {
   int numActionari = 0;
   public void actionPerformed(ActionEvent e) {
      numActionari++;
      eticheta.setText(textEticheta + numActionari);
   }
}
```

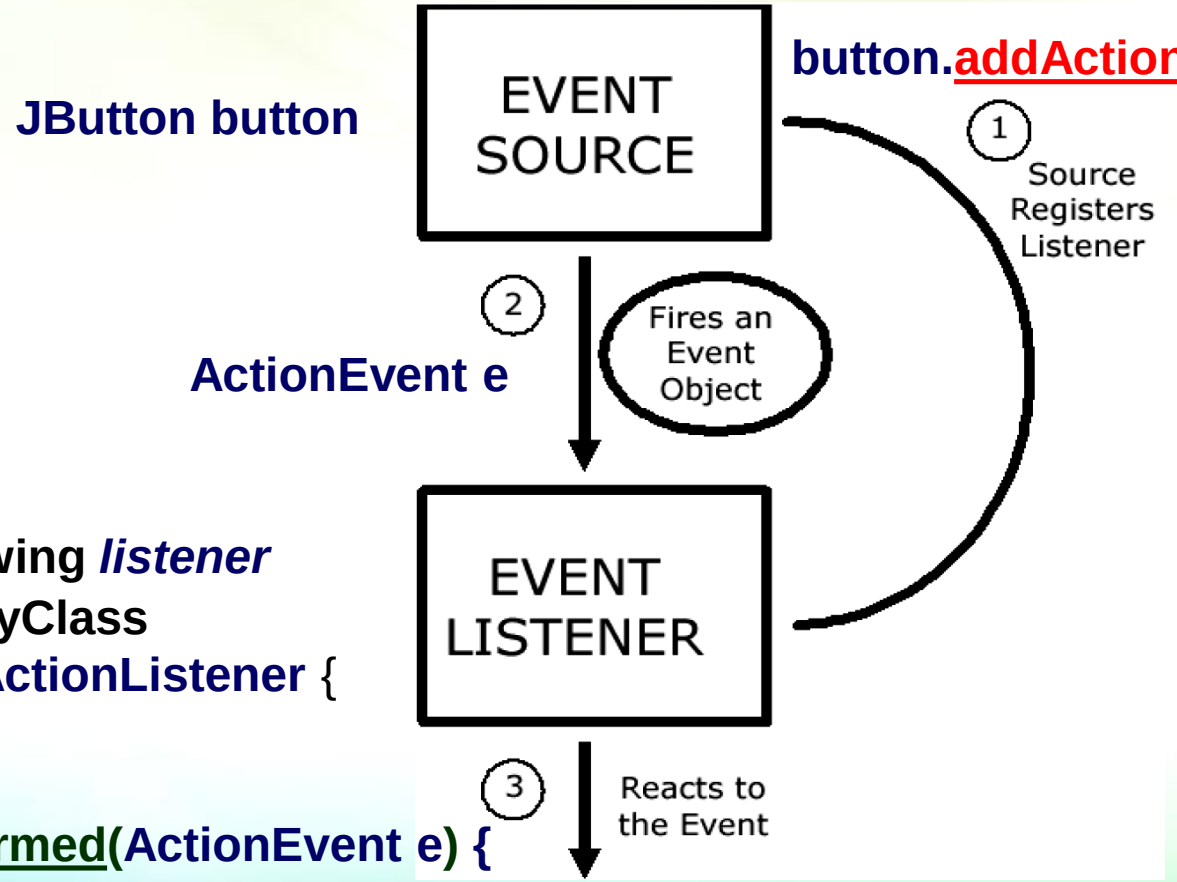
Elementele unei aplicatii grafice Swing

Tratarea evenimentelor – utilizarea altei clase "ascultator" – varianta detaliata

```
public class ElementeAplicatieSwing {
  public static void main(String[] args) {
     // ...
     // Atasarea unui obiect al unei clasei ascultator,care implementeaza
     // ActionListener, inclusiv metoda actionPerformed() de tratare
     AscultatorActionare obiectAscultator = new AscultatorActionare()
     buton.addActionListener( obiectAscultator )
     // ...
  }
class AscultatorActionare implements ActionListener {
  int numActionari = 0;
  public void actionPerformed(ActionEvent e) {
     numActionari++;
     eticheta.setText(textEticheta + numActionari);
  }
}
```

Tratarea evenimentelor in interfetele grafice Swing

Modelul folosit in tratarea evenimentelor (crearea interactivitatii) in interfetele grafice Swing



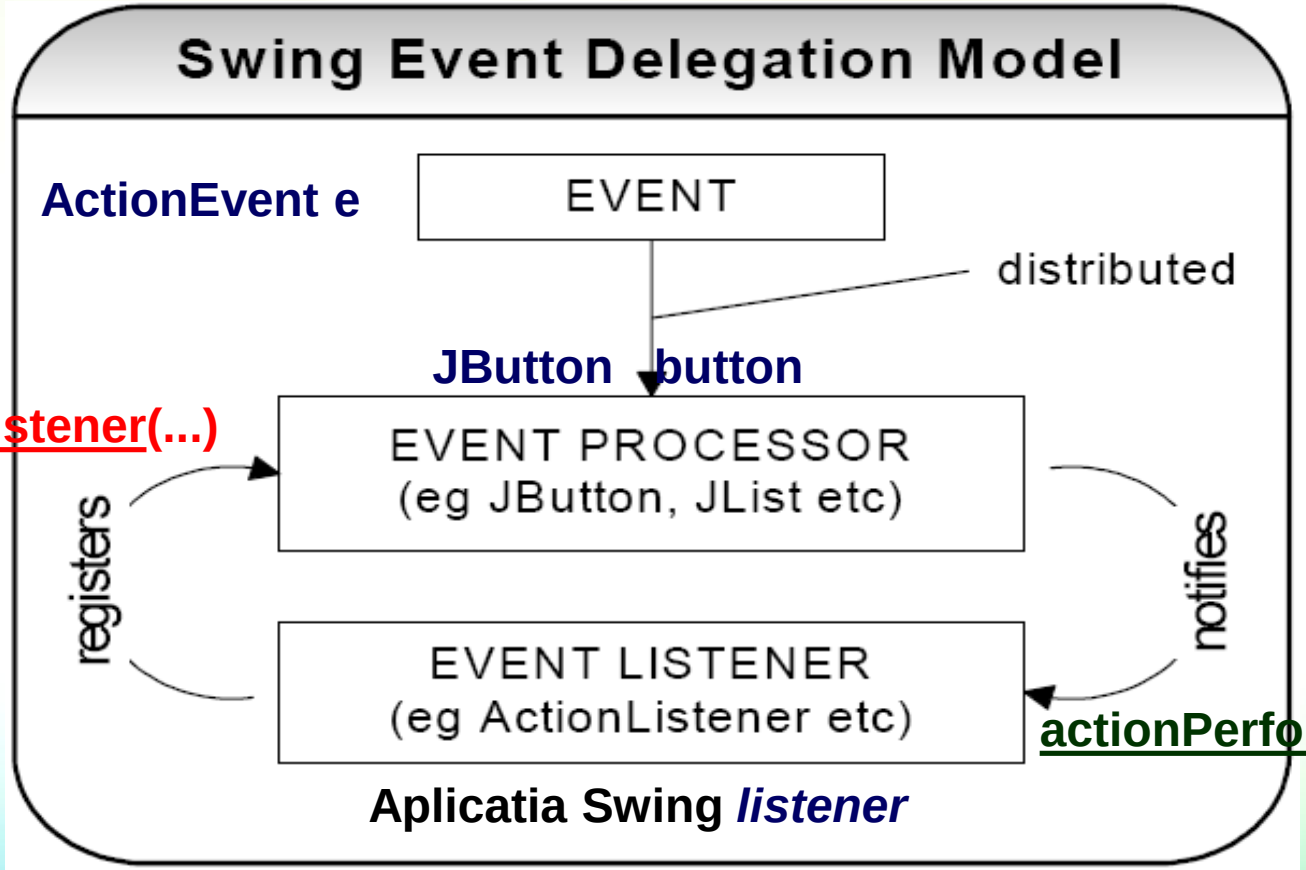

```
// Aplicatia Swing listener
public class MyClass
implements ActionListener {
```

```
// Codul prin care aplicatia reactioneaza la apasarea butonului
```

Tratarea evenimentelor in interfetele grafice Swing

Modelul folosit in tratarea evenimentelor (crearea interactivitatii)

- modelul delegarii evenimentelor Swing

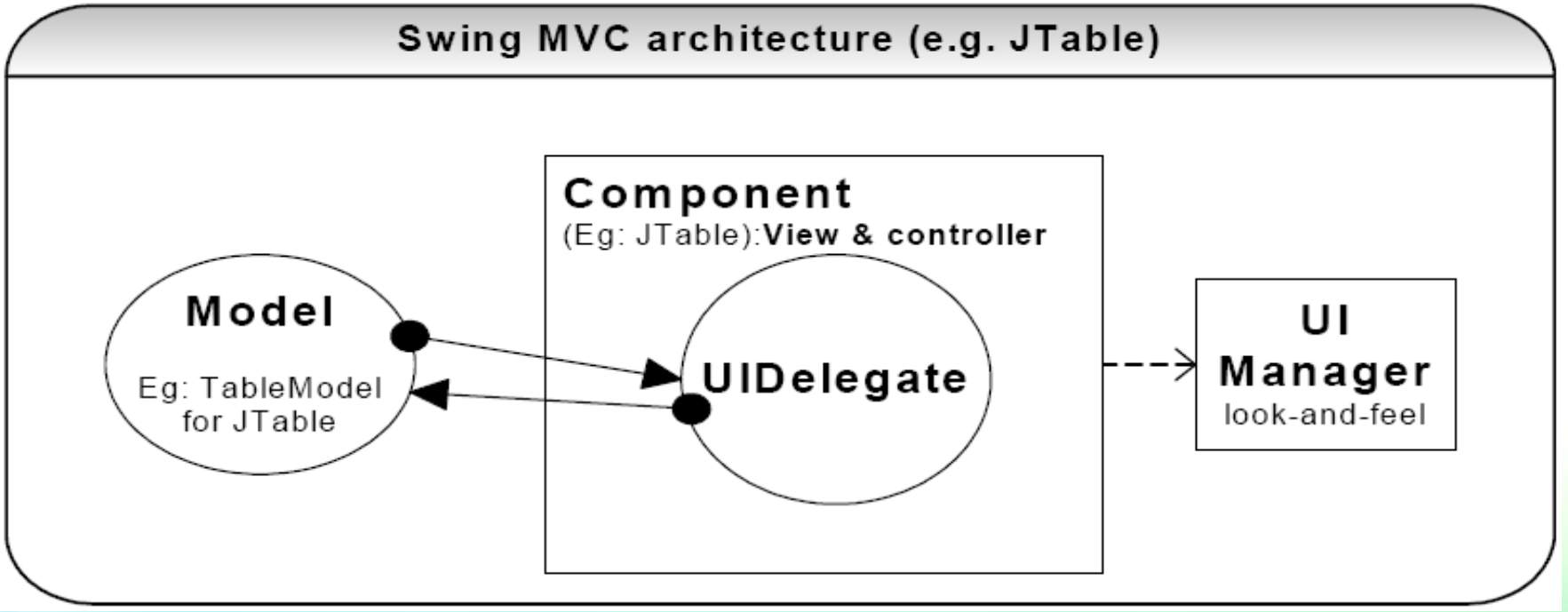


Tratarea evenimentelor in interfetele grafice Swing

Modelul folosit in tratarea evenimentelor (crearea interactivitatii)

- forma de **arhitectura MVC (*model – view – controller*)**

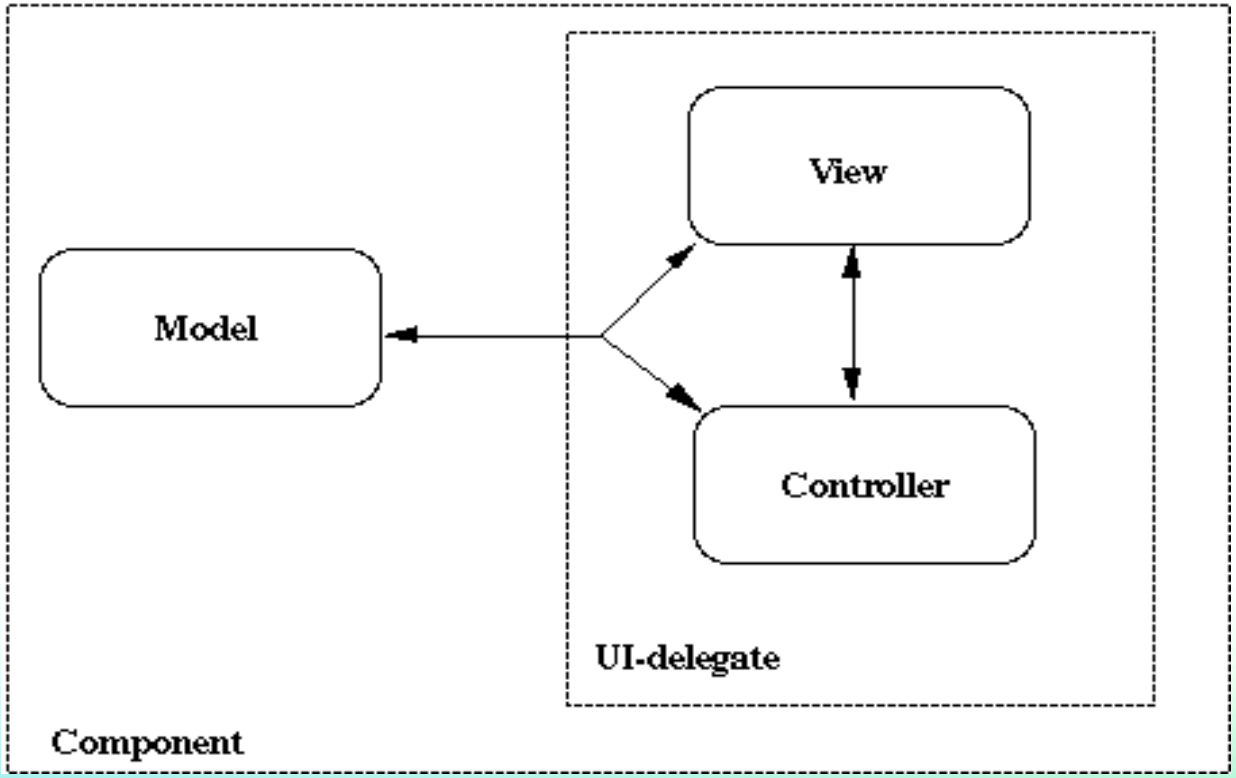
- in care **subsistemul de prezentare** (***View***) si **subsistemul de control** (***Controller***) formeaza un **subsistem de tratare a evenimentelor prin delegare** (***UIDelegate***)



Tratarea evenimentelor in interfetele grafice Swing

Swing foloseste un **MVC simplificat**: numit ***Model – UI-delegate***

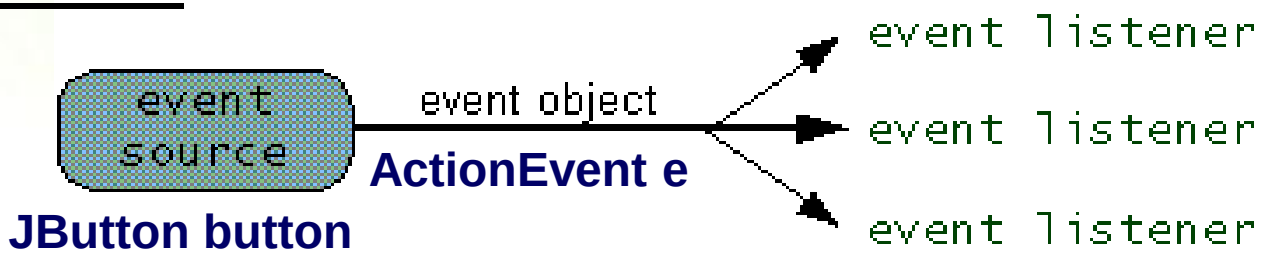
- subsistemul **Model mentine informatiile componentei**
- subsistemul **UI-delegate afiseaza componenta** si **reactioneaza la evenimentele care se propaga prin componenta**



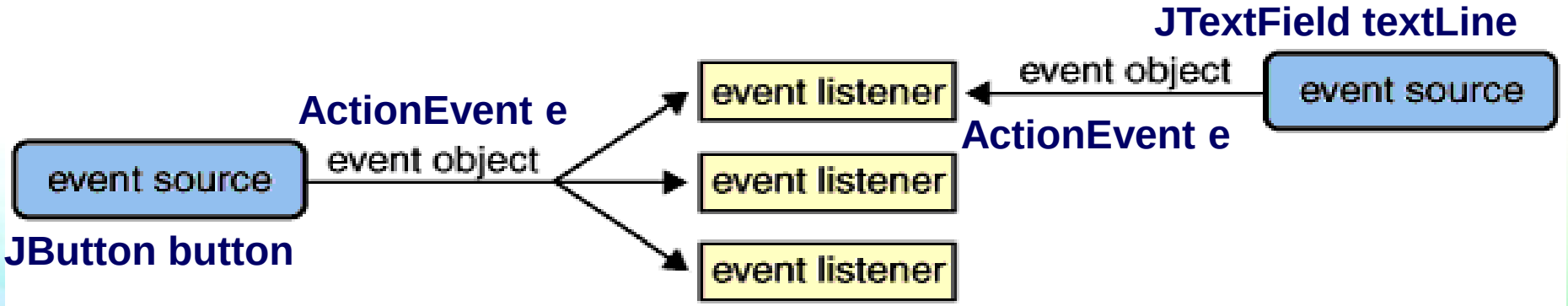
Tratarea evenimentelor in interfetele grafice Swing

Modelul folosit in tratarea evenimentelor (crearea interactivitatii)

- o **sursa de evenimente** poate **trimite obiecte eveniment catre mai multi ascultatori de evenimente**



- in plus, **mai multe surse de evenimente** pot **trimite obiecte eveniment catre acelasi ascultator de evenimente**



Tratarea evenimentelor in interfetele grafice Swing

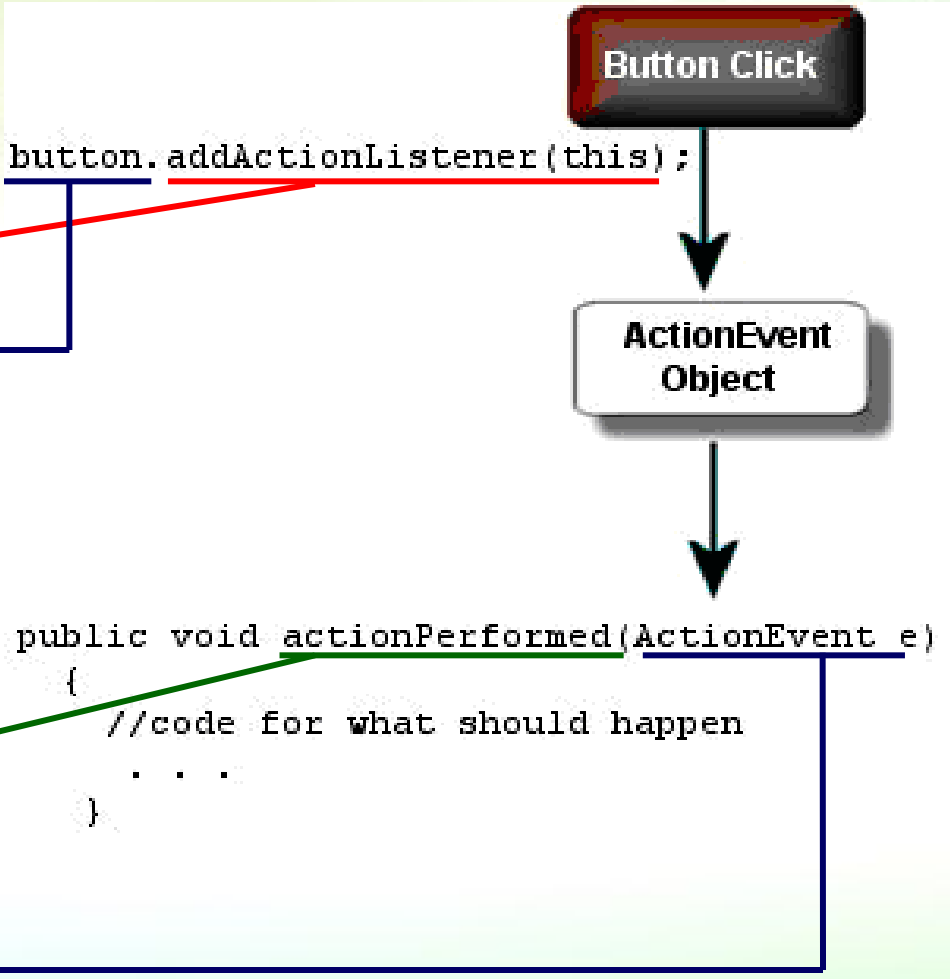
Exemple de surse de evenimente si ascultatoarele de evenimente asociate lor

Act that results in the event	Listener type
User clicks a button, or chooses a menu item, or presses Enter while typing in a text field	`ActionListener`
User closes a frame (main window)	`WindowListener`
User presses a mouse button while the cursor is over a component	`MouseListener`
User moves the mouse over a component	`MouseMotionListener`
Component becomes visible	`ComponentListener`
Component gets the keyboard focus	`FocusListener`
Table or list selection changes	`ListSelectionListener`

Tratarea evenimentelor in interfetele grafice Swing

Ascultatorul de evenimente (`ActionListener`) trebuie:

- **sa se INREGISTREZE (`addActionListener(this)`) la sursa de evenimente (`button`)**
- **si sa ofere cel putin o metoda prin care**
  - **sursa de evenimente sa il poata NOTIFICA pe ascultatorul evenimentului (`actionPerformed(...)`)**
  - **pasandu-i o referinta catre obiectul eveniment (`ActionEvent e`)**

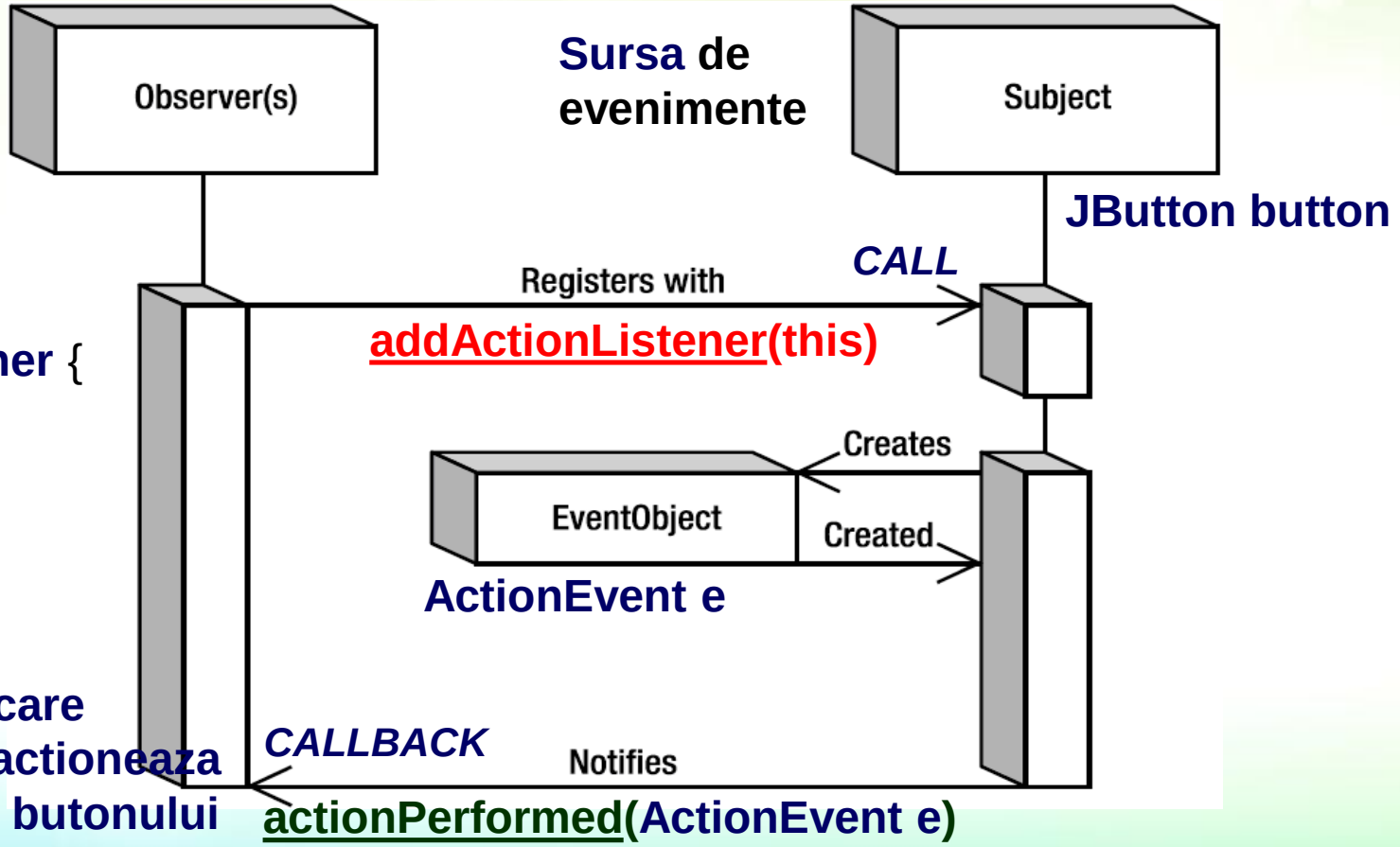


Tratarea evenimentelor in interfetele grafice Swing

Acesta este un exemplu de pattern de proiectare *Observer* (cunoscut si ca ***Publisher-Subscriber***, **REGISTER–NOTIFY**, ***callback***, **etc.**):

Aplicatia Swing *listener*

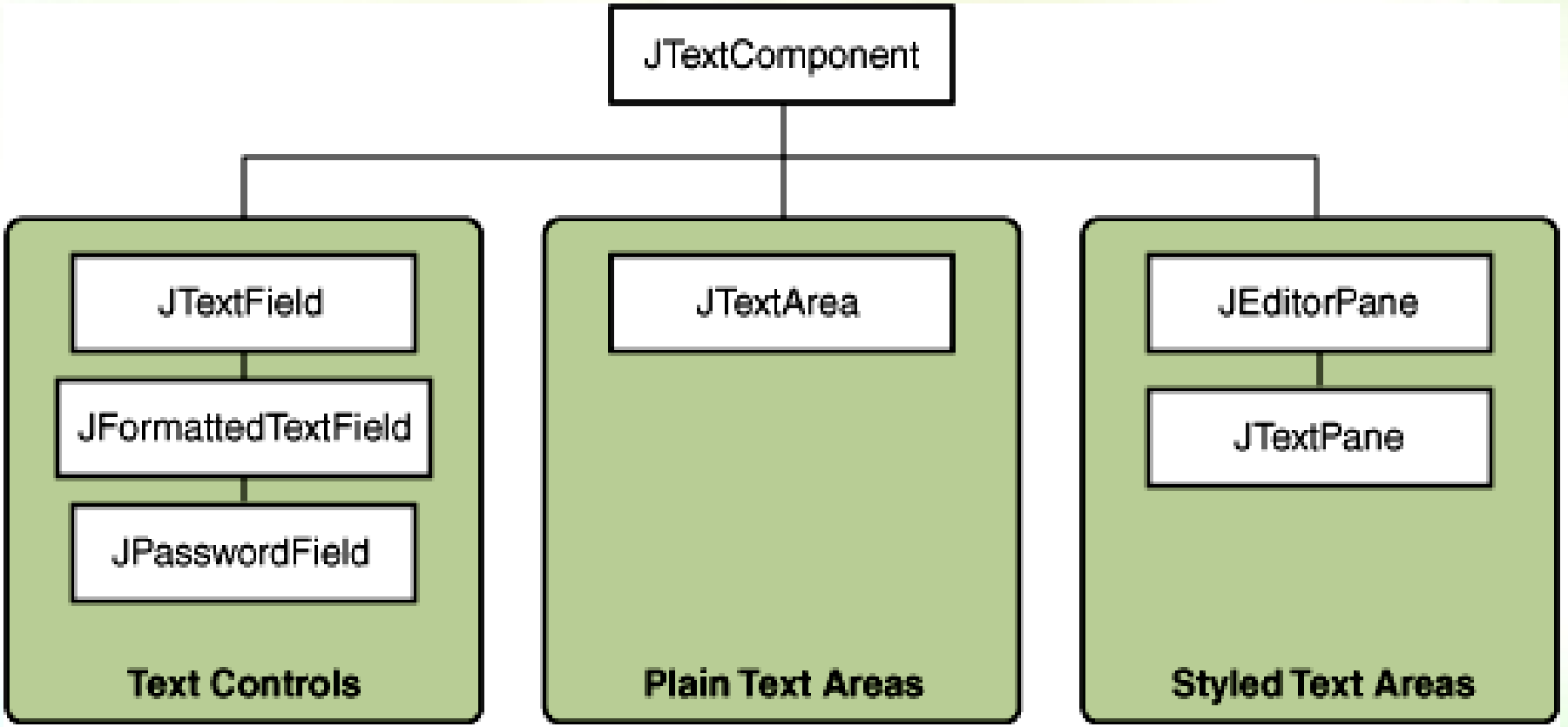
public class **MyClass implements ActionListener** {



Codul prin care aplicatia reactioneaza la apasarea butonului

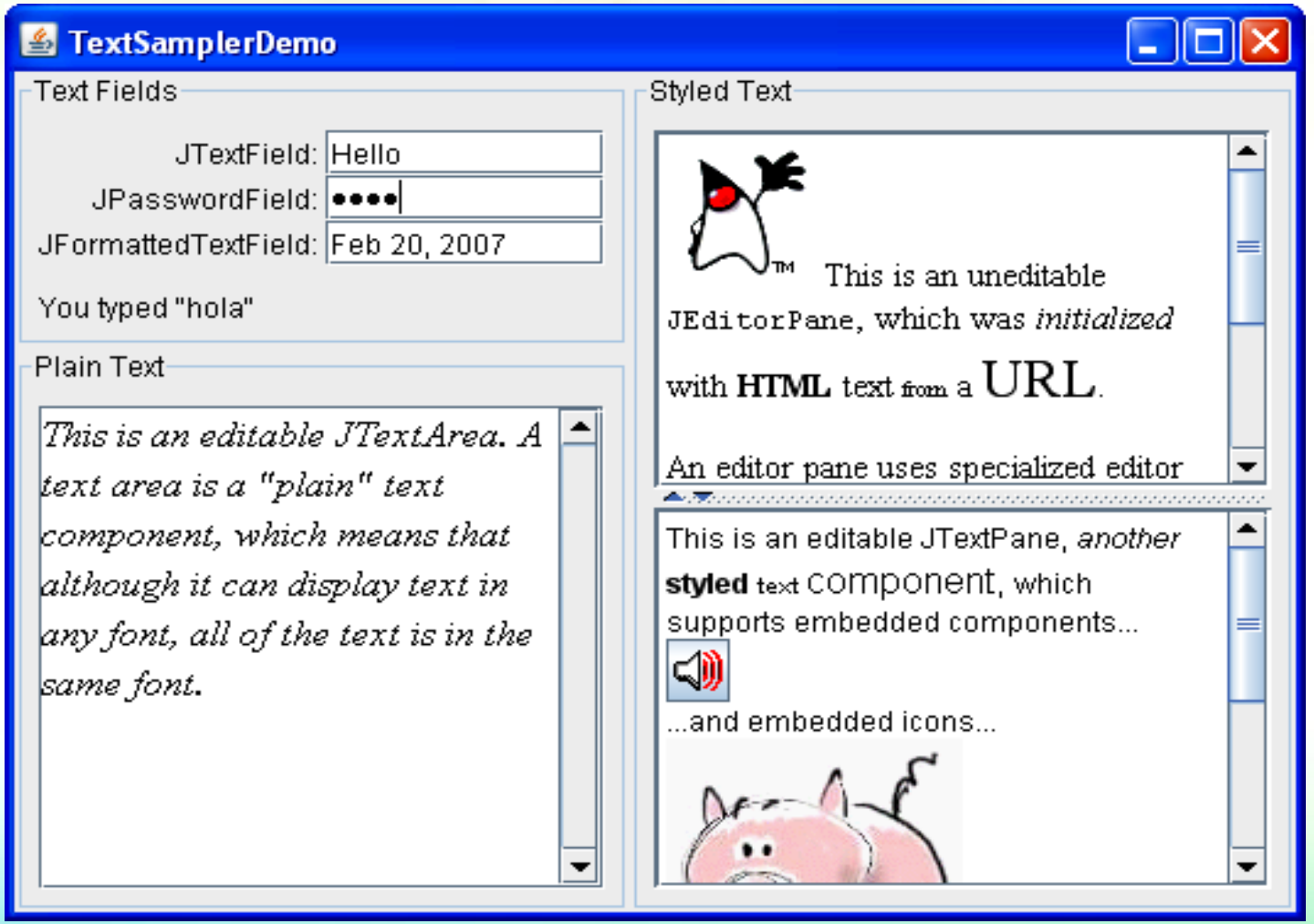
Componente Swing de tip text

Ierarhia de clase Swing de tip text



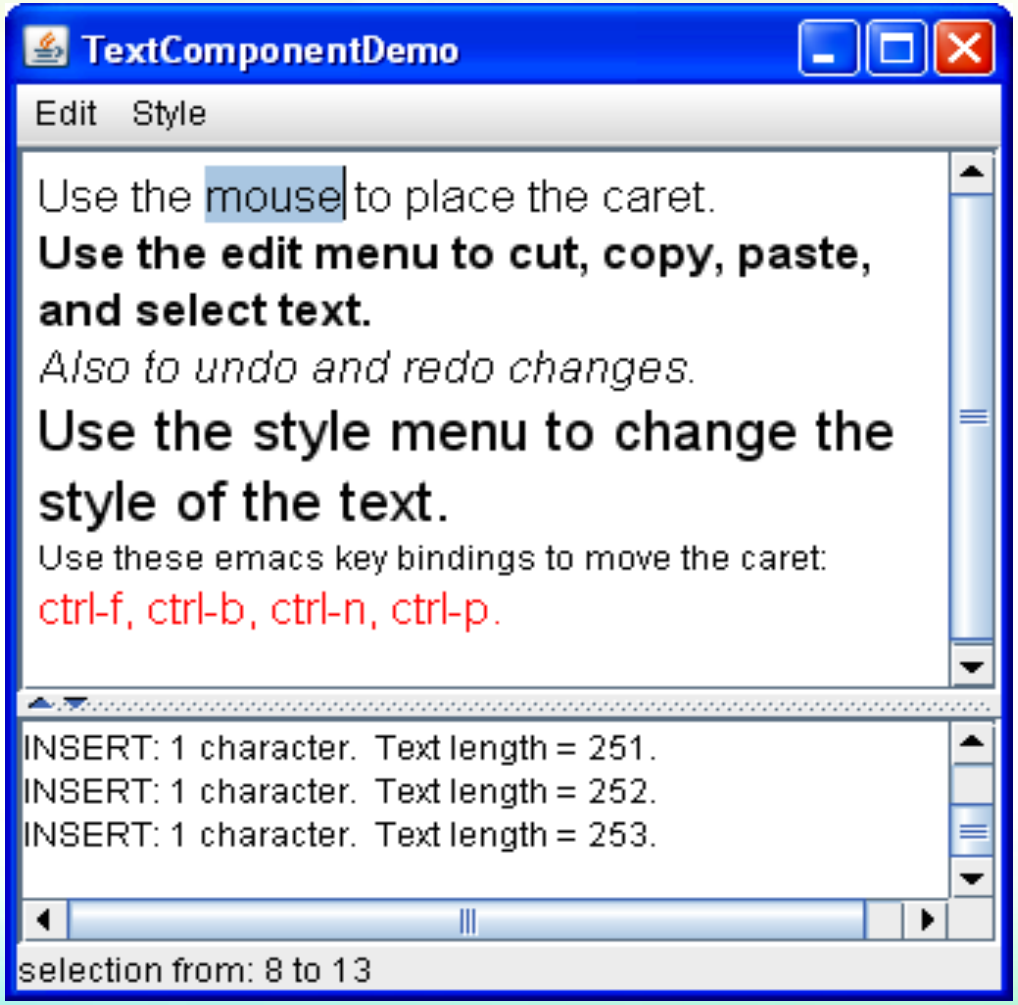
Componente Swing de tip text

Ilustrarea diferitelor componente Swing de tip text



Componente Swing de tip text

Caracteristici care pot fi modificate in componentele Swing de tip text



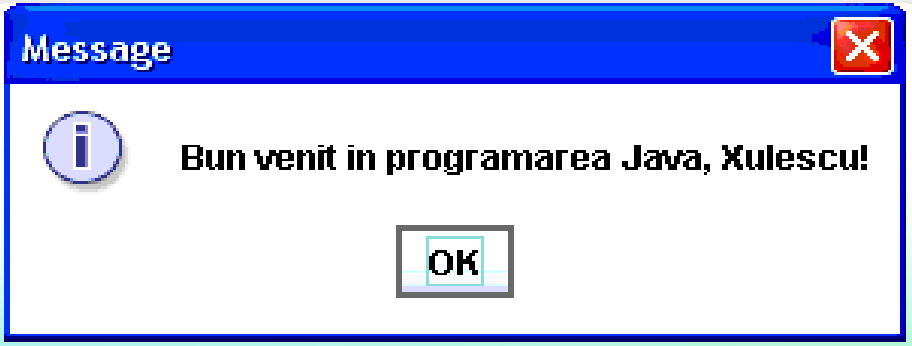
Ferestre de dialog Swing

Pentru a obtine de la utilizator o valoare de tip sir de caractere (`String`):

```
String nume = JOptionPane.showInputDialog("Introduceti-va numele:");
```

Pentru a se afisa un mesaj catre utilizator se foloseste sintaxa:

```
JOptionPane.showMessageDialog(null,"Bun venit in lumea Java,Xulescu!");
```



Ferestre de dialog Swing

Ambele situatii impun importul clasei JOptionPane din pachetul de clase de biblioteca grafice javax.swing, (**sau** a **intregului pachet de clase** de biblioteca **javax.swing**).

Pentru **a se importa clasa** `JOptionPane` din pachetul de clase Swing `javax.swing` se foloseste sintaxa:

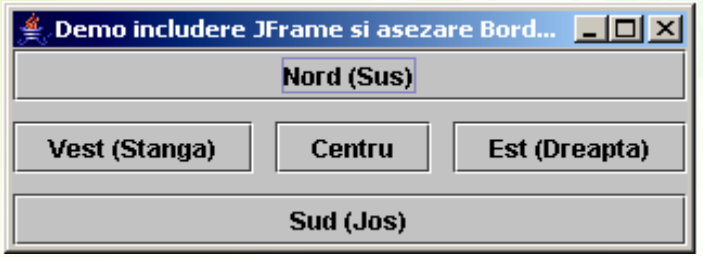
```
import javax.swing.JOptionPane;
```

Pentru **a se importa intreg pachetul de clase** `javax.swing` se foloseste sintaxa:

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

Modalitati de a crea containerul de nivel maxim (*top-level*)

1. Utilizarea unui obiect de tip Frame




```
import java.awt.*;
import javax.swing.*;

public class IncludereJFrame {
    public static void main(String[] args) {
      // Crearea obiectului cadru, cu titlu specificat
      JFrame cadru = new JFrame("Demo includere JFrame si asezare BorderLayout");

      // Obtinerea panoului de continut intern cadrului (container de componente)
      Container container = cadru.getContentPane();

      // Asezarea componentelor in panou (la 10 pixeli de marginea panoului)
      container.setLayout(new BorderLayout(10, 10));

      // Adaugarea a 5 butoane la panoul cadrului (ferestrei)
      container.add(new JButton("Est (Dreapta)"), BorderLayout.EAST);
      container.add(new JButton("Sud (Jos)"), BorderLayout.SOUTH);
      container.add(new JButton("Vest (Stanga)"), BorderLayout.WEST);
      container.add(new JButton("Nord (Sus)"), BorderLayout.NORTH);
      container.add(new JButton("Centru"), BorderLayout.CENTER);

      // Stabilirea dimensiunii ferestrei
      cadru.pack();
      // Stabilirea vizibilitatii ferestrei (Atentie: implicit e false!)
      cadru.setVisible(true);
    }
}
```

Modalitati de a crea containerul de nivel maxim (*top-level*)

2. Extinderea prin mostenire a clasei JFrame

```
import java.awt.*;
import javax.swing.*;

public class ExtensieJFrame extends JFrame {
    public ExtensieJFrame() {
      // Obtinerea panoului de continut intern cadrului (container de componente)
      Container container = getContentPane();
      // Asezarea componentelor in panou (la 10 pixeli de marginea panoului)
      container.setLayout(new BorderLayout(10, 10));
      // Adaugarea a 5 butoane la panoul cadrului (ferestrei)
      container.add(new JButton("Est (Dreapta)"), BorderLayout.EAST);
      container.add(new JButton("Sud (Jos)"), BorderLayout.SOUTH);
      container.add(new JButton("Vest (Stanga)"), BorderLayout.WEST);
      container.add(new JButton("Nord (Sus)"), BorderLayout.NORTH);
      container.add(new JButton("Centru"), BorderLayout.CENTER);
    }
    public static void main(String[] args) {
      // Crearea obiectului cadru
      ExtensieJFrame cadru = new ExtensieJFrame();
      // Adaugarea titlului ferestrei
      cadru.setTitle("Demo extindere JFrame si asezare BorderLayout");
      // Compactarea componentelor
      cadru.pack();
      // Stabilirea vizibilitatii ferestrei (Atentie: implicit e false!)
      cadru.setVisible(true);
    }
}
```

Modalitati de a crea containerul de nivel maxim (*top-level*)

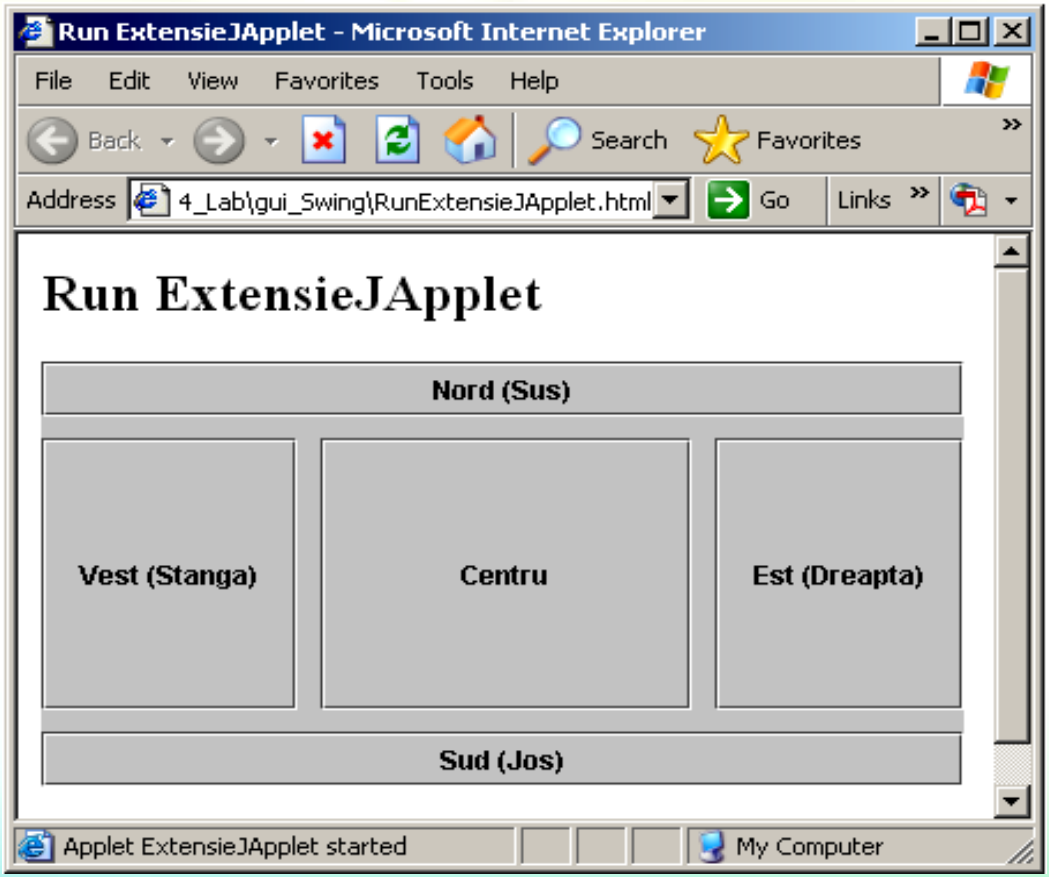
3. Extinderea clasei JApplet (in cazul miniaplicatiilor Java)

```
import java.awt.*;
import javax.swing.*;
/**
 * Se executa prin includerea intr-o pagina Web a unui tag HTML de genul:
 *    <APPLET  CODE = "ExtensieJApplet.class" WIDTH = 400 HEIGHT = 200 >
 *    </APPLET>
 */
public class ExtensieJApplet extends JApplet {
    /**
     * Metoda de initializare a appletului. Apelata de browser la prima
     * utilizare a appletului, stabileste layout-ul (modul de dispunere a
     * componentelor in panoul de continut) si adauga componentele in panou.
     */
    public void init() {
      // Obtinerea panoului de continut intern cadrului (container de componente)
      Container container = getContentPane();
      // Asezarea componentelor in panou (la 10 pixeli de marginea panoului)
      container.setLayout(new BorderLayout(10, 10));
      // Adaugarea a 5 butoane la panoul appletului
      container.add(new JButton("Est (Dreapta)"), BorderLayout.EAST);
      container.add(new JButton("Sud (Jos)"), BorderLayout.SOUTH);
      container.add(new JButton("Vest (Stanga)"), BorderLayout.WEST);
      container.add(new JButton("Nord (Sus)"), BorderLayout.NORTH);
      container.add(new JButton("Centru"), BorderLayout.CENTER);
    }
}
```

Modalitati de a crea containerul de nivel maxim (*top-level*)

Pentru includerea unui applet intr-o pagina HTML in vederea vizualizarii ei **se va adauga in corpul fisierului .html** urmatorul cod:

```
<APPLET  CODE = "ExtensieJApplet.class" WIDTH = 400 HEIGHT = 200 >
</APPLET>
```

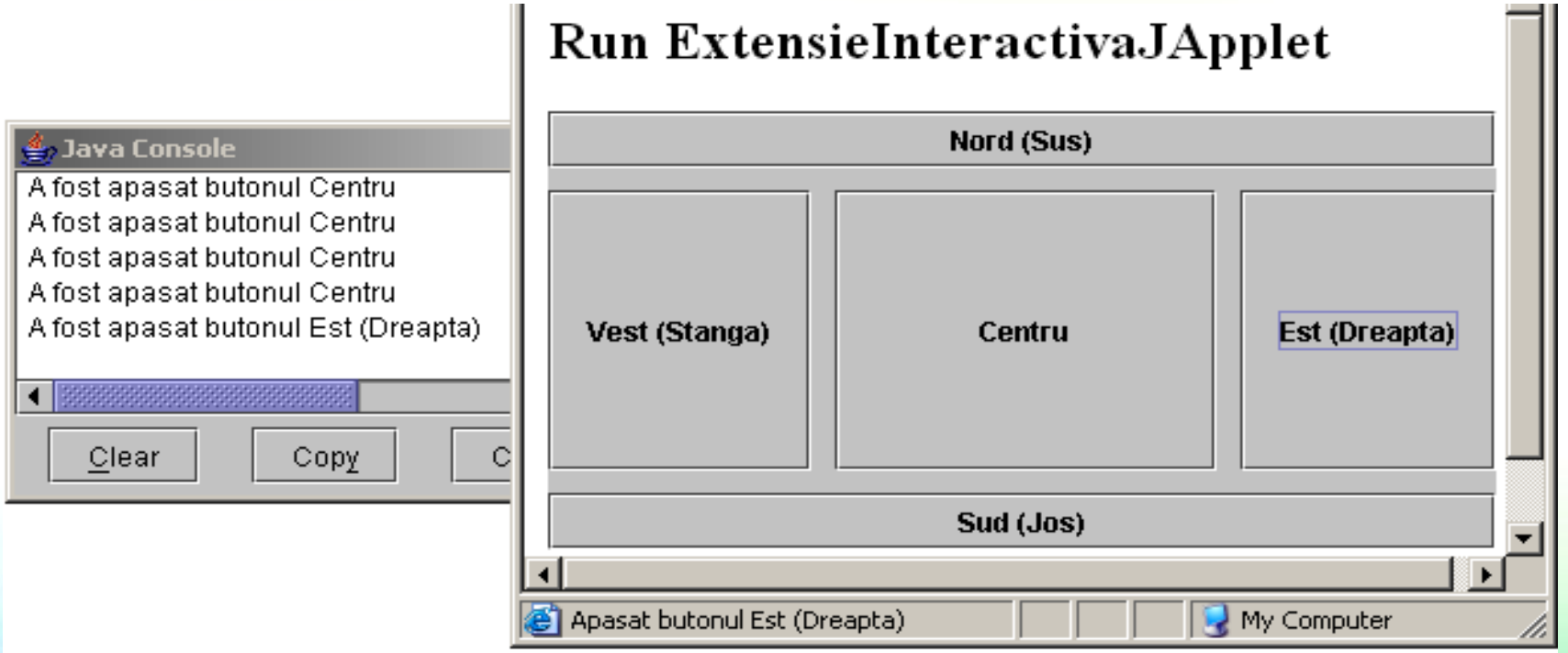


Tratarea evenimentelor in interfetele grafice Swing

Programul ExtensieInteractivaJApplet ilustreaza:

- **crearea unui applet** prin **extinderea clasei JApplet**, si

- **tratarea evenimentului "actionare"** (ActionEvent) pentru **componentele buton** (JButton) folosind implementarea interfetei de tratare a evenimentelor (ActionListener)



Tratarea evenimentelor in interfetele grafice Swing

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

public class ExtensieInteractivaJApplet extends JApplet {

    public void init() {

        // Obtinerea panoului de continut (content pane) creat de browser pentru
        // executia appletului (container in care vor fi plasate componentele)
        Container container = getContentPane();

        // Stabilirea layout-ului panoului, BorderLayout cu spatiu 10 pixeli
        container.setLayout(new BorderLayout(10, 10));

        // Adaugarea a 5 butoane la panoul appletului
        JButton b1 = new JButton("Est (Dreapta)");
        JButton b2 = new JButton("Sud (Jos)");
        JButton b3 = new JButton("Vest (Stanga)");
        JButton b4 = new JButton("Nord (Sus)");
        JButton b5 = new JButton("Centru");

        container.add(b1, BorderLayout.EAST);
        container.add(b2, BorderLayout.SOUTH);
        container.add(b3, BorderLayout.WEST);
        container.add(b4, BorderLayout.NORTH);
        container.add(b5, BorderLayout.CENTER);
```

Tratarea evenimentelor in interfetele grafice Swing

```
        // Crearea unui obiect "ascultator" de "evenimente actionare"
        // (pe care le trateaza)
        ActionListener obiectAscultatorActionare = new ActionListener() {

          // Tratarea actionarii unui buton
          public void actionPerformed(ActionEvent ev) {

            // Mesaj informare in consola Java
            System.out.println("A fost apasat butonul " + ev.getActionCommand());

            // Mesaj informare in bara de stare
            showStatus("Apasat butonul " + ev.getActionCommand());
          }
        };

        // Inregistrarea "ascultatorului" de "evenimente actionare" la "sursele"
        // de evenimente
        b1.addActionListener(obiectAscultatorActionare);
        b2.addActionListener(obiectAscultatorActionare);
        b3.addActionListener(obiectAscultatorActionare);
        b4.addActionListener(obiectAscultatorActionare);
        b5.addActionListener(obiectAscultatorActionare);
    }
}
```

Utilizarea componentelor grafice Swing pentru lucrul cu text

Programul EcouGrafic_Swing ilustreaza utilizarea componentelor grafice Swing pentru lucrul cu text, de tip **JTextField** si **JTextArea**

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

// Ecou grafic bazat pe utilizarea JtextField si a JTextArea
public class EcouGrafic_Swing extends JFrame {

  // Intrare - linie de text grafica (JtextField)
  private static JTextField inTextGrafic;

  // Iesire - zona de text grafica (JtextArea)
  private static JTextArea outTextGrafic;

  // Initializari grafice
  public EcouGrafic_Swing() {

    // Stabilire titlu fereastra (JFrame)
    super("Ecou grafic simplu Swing");

    // Obtinerea panoului de continut intern cadrului (container de componente)
    Container containerCurent = this.getContentPane();

    // Asezarea componentelor in panou
    containerCurent.setLayout(new BorderLayout());
```

Utilizarea componentelor grafice Swing pentru lucrul cu text

```
        // Zona de text non-editabila de iesire (cu posibilitati de defilare)
        outTextGrafic = new JTextArea(5, 40);
        JScrollPane scrollPane = new JScrollPane(outTextGrafic,
                        JScrollPane.VERTICAL_SCROLLBAR_ALWAYS,
                        JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
        containerCurent.add("Center", scrollPane);
        outTextGrafic.setEditable(false);

        // Camp de text editabil de intrare
        inTextGrafic = new JTextField(40);
        containerCurent.add("South", inTextGrafic);

        System.out.println("\nPentru oprire introduceti '.' si <Enter>\n");

        // Inregistrarea "ascultatorului" de "evenimente actionare" la
        // "obiectul sursa" intrare de text
        inTextGrafic.addActionListener(ascultatorInText);

        // Inregistrarea "ascultatorului" de "evenimente fereastra" la
        // "sursa" (fereastra curenta)
        this.addWindowListener(ascultatorInchidere);

        pack();
        show();

        // Cerere focus pe intrarea de text din fereastra curenta
        inTextGrafic.requestFocus();
    }
```

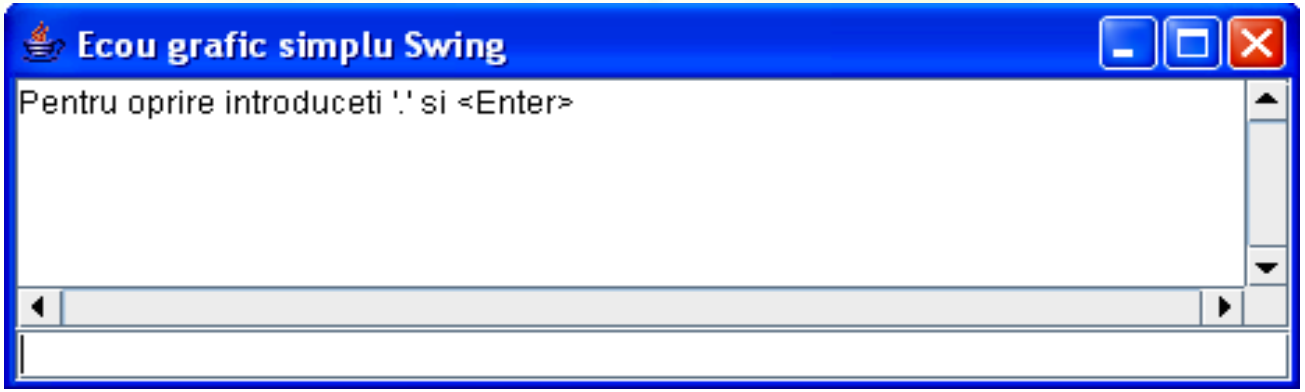
Utilizarea componentelor grafice Swing pentru lucrul cu text

```
  // Crearea unui "ascultator" de "evenimente actionare", obiect al unei
  // clase "anonime" care implementeaza interfata ActionListener
ActionListener ascultatorInText = new ActionListener() {

    // Tratarea actionarii intrarii de text (introducerii unui "Enter")
    public void actionPerformed(ActionEvent ev) {

      // Citirea unei linii de text din intrarea de text grafica
      String sirCitit = inTextGrafic.getText();

      // Pregatirea intrarii de text pentru noua intrare (golirea ei)
      inTextGrafic.setText("");

      // Scrierea liniei de text in zona de text grafica
      outTextGrafic.append("S-a introdus: " + sirCitit + "\n");

      // Conditie terminare program
      if (sirCitit.equals(new String(".")))
            System.exit(0);
    }
  };

  // Punctul de intrare in program
  public static void main (String args[]) {

    EcouGrafic_Swing ecouGraficJTFJTA = new EcouGrafic_Swing();
  }
}
```

Utilizarea componentelor grafice Swing pentru lucrul cu text

Executia programului EcouGrafic_Swing

- initial



- dupa introducerea “Hello! <enter>”