Ingineria Sistemelor de Programare

Interfete grafice (Swing)

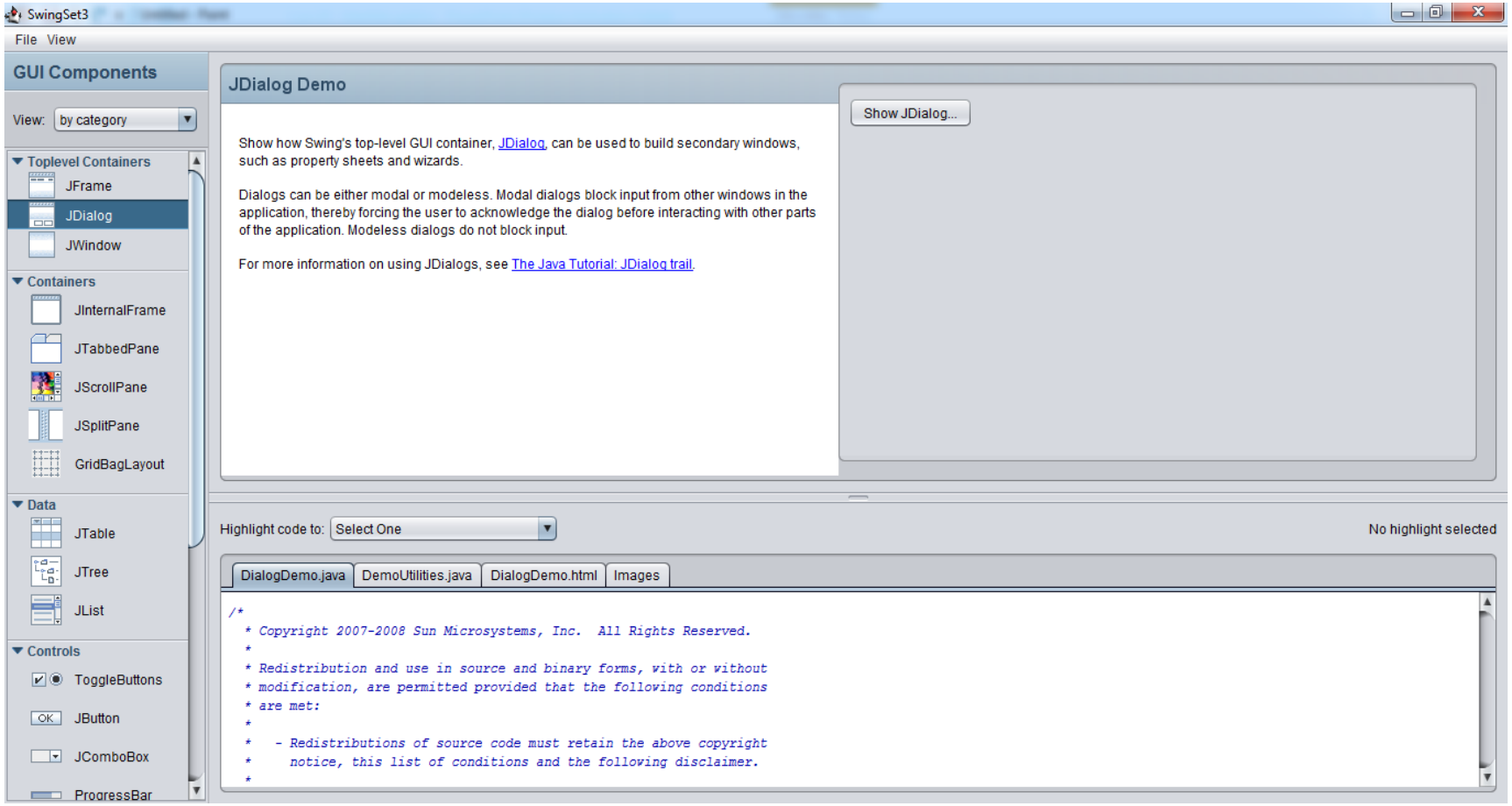
mihai.hulea@aut.utcluj.ro

2017

Scurt istoric

- AWT: Abstract Windowing Toolkit
    - import java.awt.*
- Swing
- Java FX

Swing Demo


SwingSet3
File View
GUI Components
View: by category
Toplevel Containers
JFrame
JDialog
JWindow
Containers
JInternalFrame
JTabbedPane
JScrollPane
JSplitPane
GridBagLayout
Data
JTable
JTree
JList
Controls
ToggleButtons
JButton
JComboBox
ProgressBar
JDialog Demo
Show JDialog...
Show how Swing's top-level GUI container, JDialog, can be used to build secondary windows, such as property sheets and wizards.
Dialogs can be either modal or modeless. Modal dialogs block input from other windows in the application, thereby forcing the user to acknowledge the dialog before interacting with other parts of the application. Modeless dialogs do not block input.
For more information on using JDialogs, see The Java Tutorial: JDialog trail.
Highlight code to: Select One
No highlight selected
DialogDemo.java
DemoUtilities.java
DialogDemo.html
Images
/*
* Copyright 2007-2008 Sun Microsystems, Inc. All Rights Reserved.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
*
* - Redistributions of source code must retain the above copyright
* notice, this list of conditions and the following disclaimer.
*

Libraria Swing

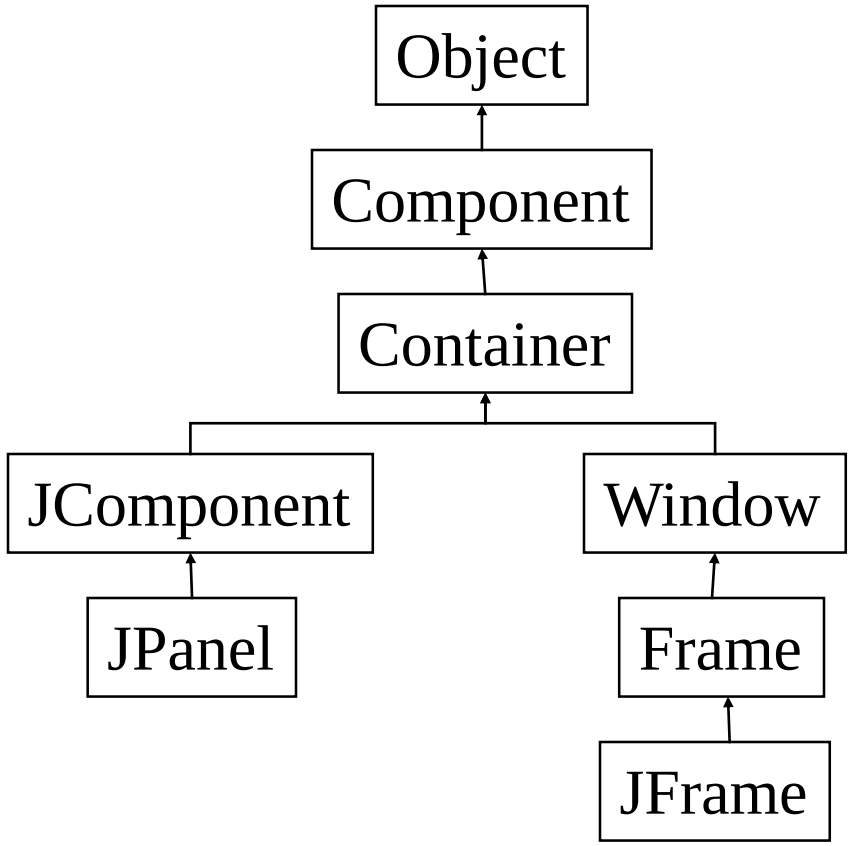
- *Swing* este o colectie de clase utilizate pentru construirea interfetelor grafice (eng. GUIs – Graphical User Interfaces). Marea majoritate a claselor sunt localizate in pachetul *javax.swing.*

- Exemple de clase: JButton, JTextBox, JTextArea, JPanel, JFrame, JMenu, JSlider, JLabel, JIcon, …

- Scopul prezentarii este acela de a face o introducere in arhitectura Swing

Componetele Swing

- Componentele Swing sunt clase Java:

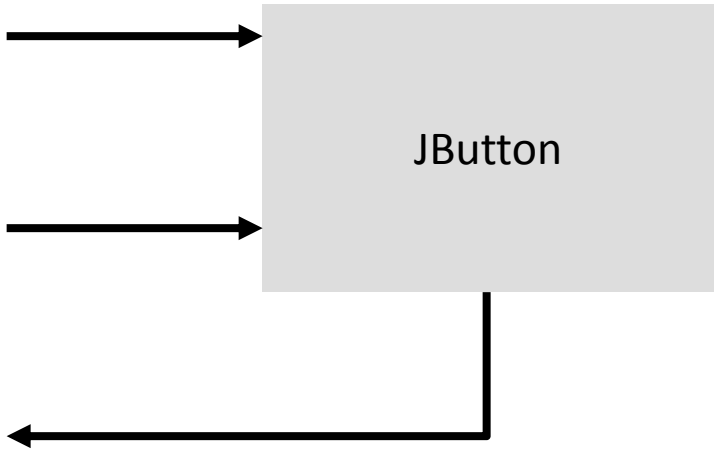


Utilizarea componentelor Swing

Structura unei component GUI

- Java: GUI component = class

- Properties
  -
- Methods
  -
- Events
  -

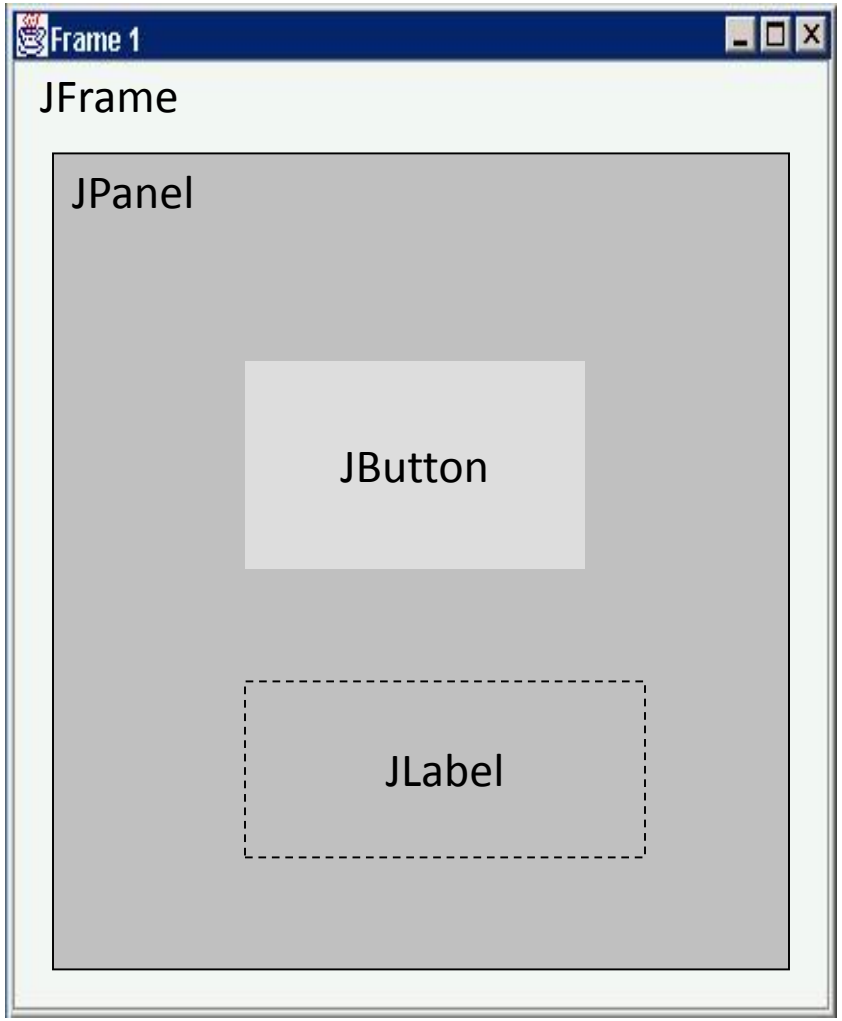


Utilizarea unei component GUI

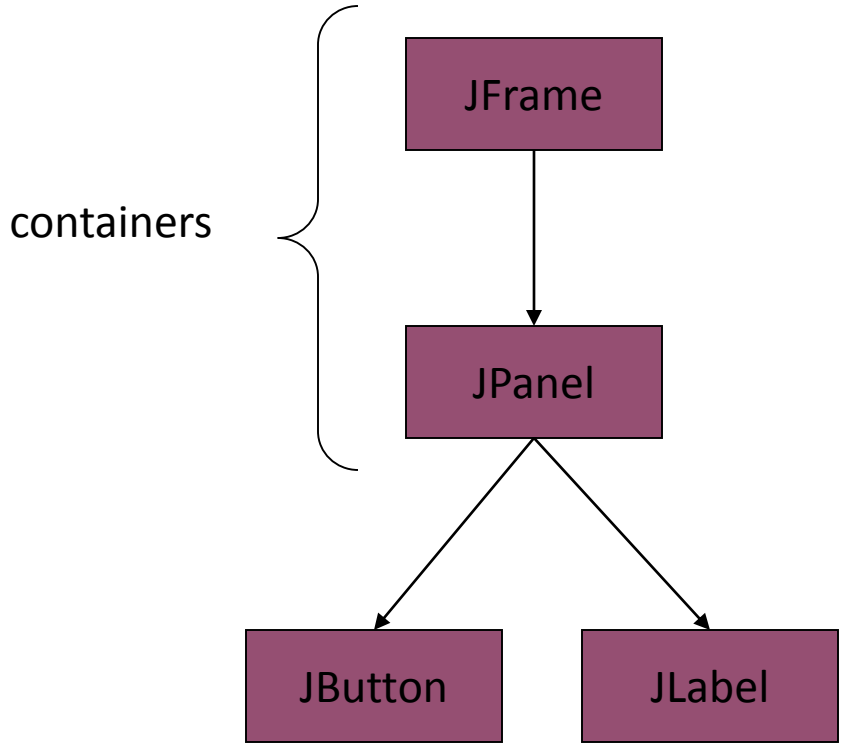
1. Create it
   - Instantiate object: b = new JButton(“press me”);
2. Configure it
   - Properties: b.text = “press me”; [avoided in java]
   - Methods: b.setText(“press me”);
3. Add it
   - panel.add(b);
4. Listen to it
   - Events: Listeners

Anatomia unei aplicatii cu GUI

GUI

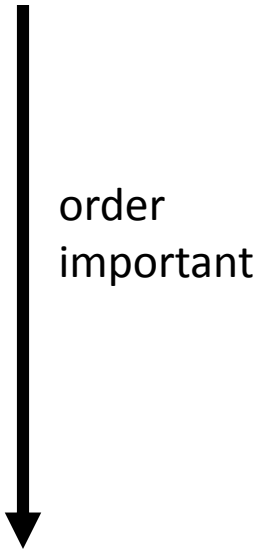


Internal structure

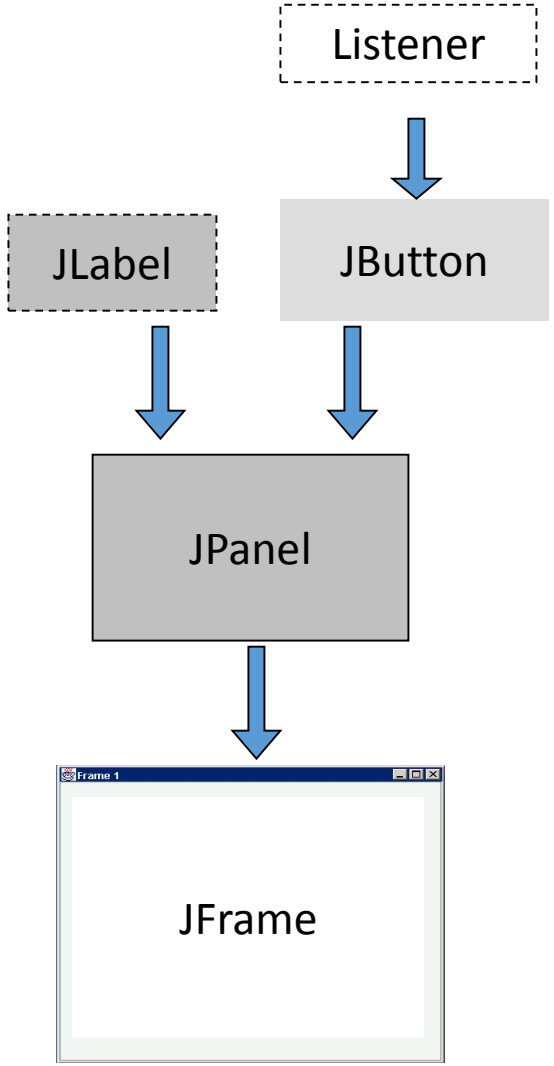


Utilizarea unei component GUI

1. Create it
2. Configure it
3. Add children (if container)
4. Add to parent (if not JFrame)
5. Listen to it



Construirea unui GUI



Exemplu

```
JFrame f = new JFrame(“title”);
JPanel p = new JPanel( );
JButton b = new JButton(“press me”);

p.add(b);                    // add button to panel
f.setContentPane(p);         // add panel to frame

f.show();
```

Aplicatie completa

```
import javax.swing.*;

class hello {
  public static void main(String[] args){
        JFrame f = new JFrame("title");
        JPanel p = new JPanel();
        JButton b = new JButton("press me");

        p.add(b);                    // add button to panel
        f.setContentPane(p);     // add panel to

        f.show();
  }
}
```

Layout Managers

Layout Managers

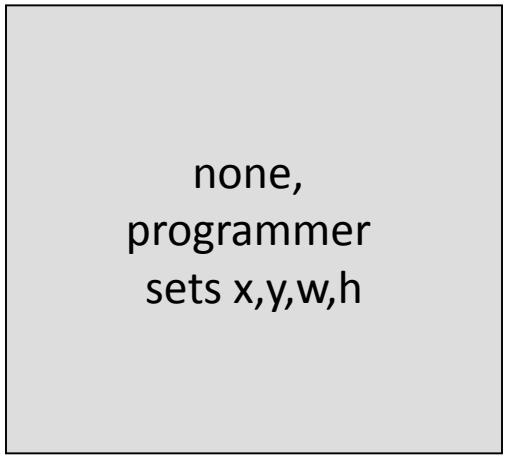
- Layout manager – responsabil cu dimensionarea si pozitionarea componentelor grafice in interiorul containerului din care fac parte
- Fiecare container are asociat un *layout manager*
- JPanel este un container ce poate fi utilizat pentru gruparea componetelor grafice. Fiecare JPanel poate avea un alt *layout manager.*

Layout Managers

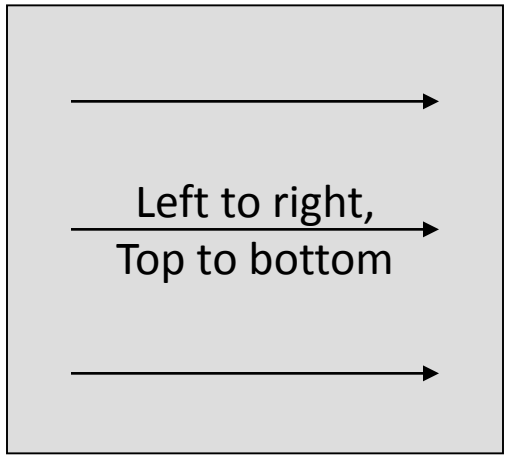
- Java vine cu 8 clase de tip Layout Manager. Cele mai commune si usor de utilizat sunt:
  - FlowLayout
  - BorderLayout
  - GridLayout
- Utilizarea acestora permite construirea interfetelor grafice pentru aplicatii simple.

Layout Managers

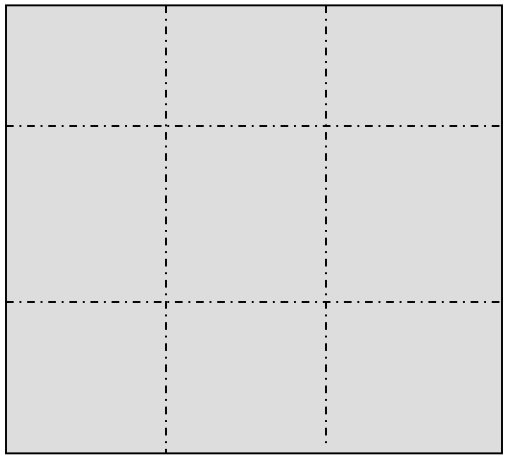
null



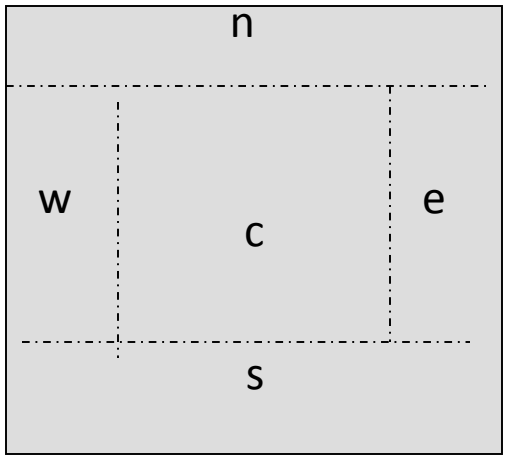
FlowLayout



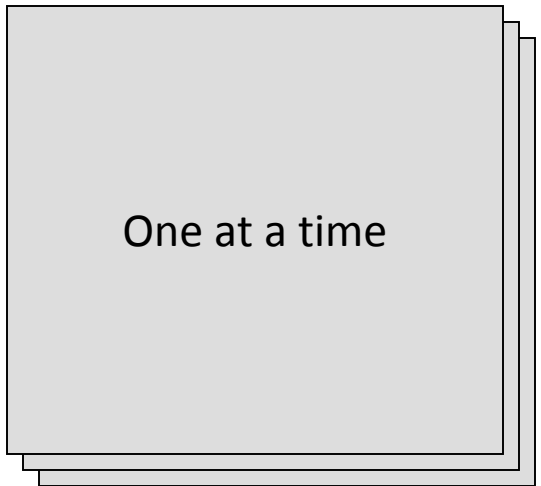
GridLayout



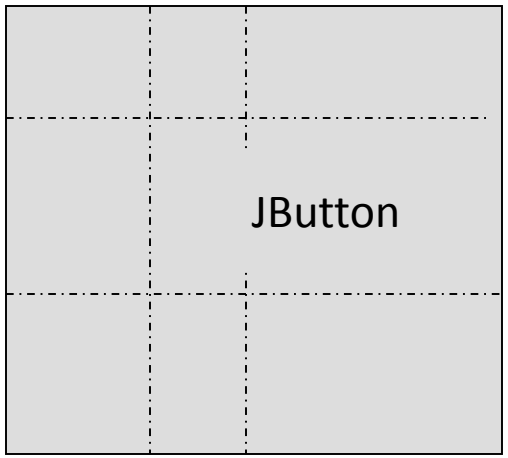
BorderLayout



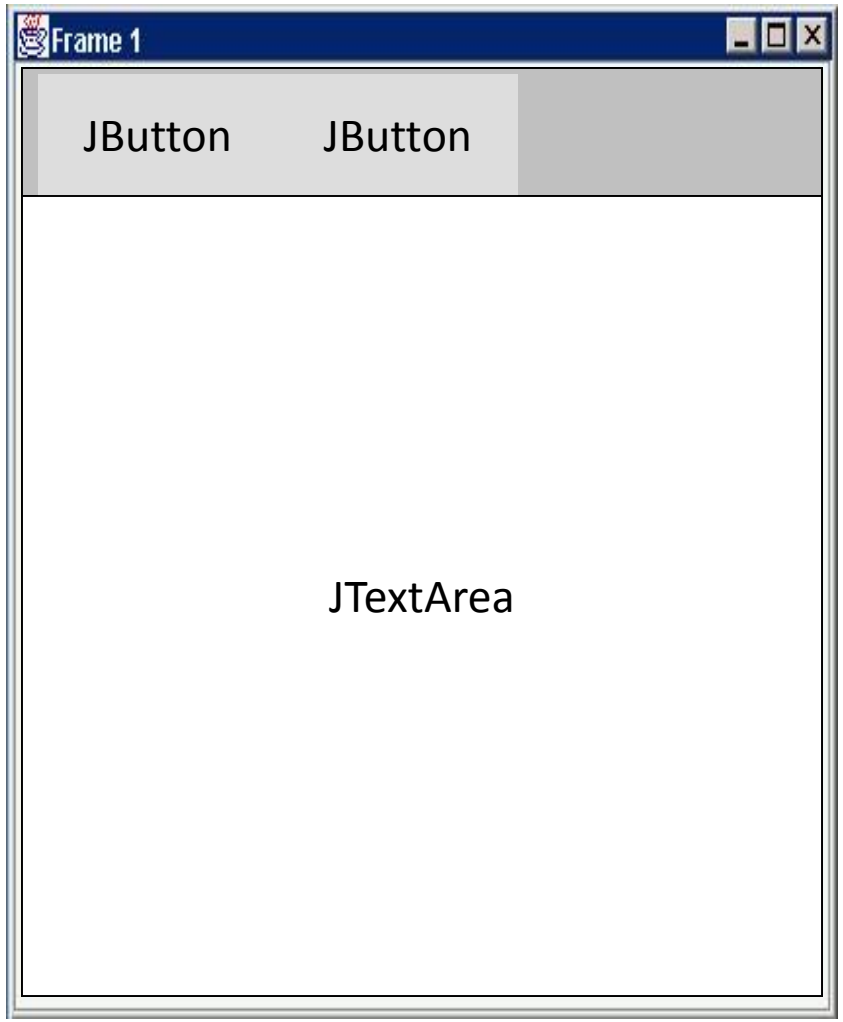
CardLayout



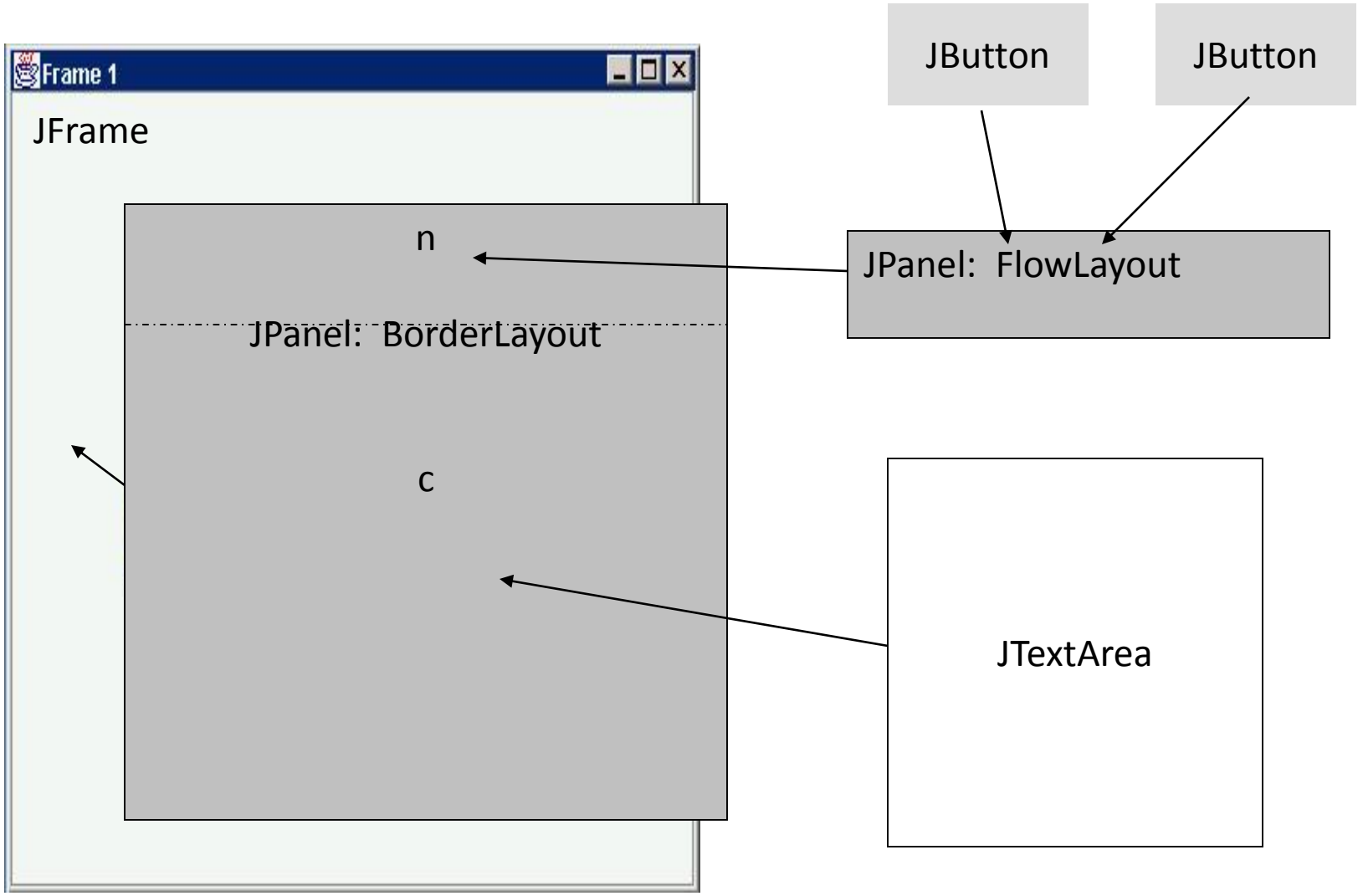
GridBagLayout



Combinatii de *layout managers*



Combinatii de *layout managers*



Exemplu: null layout

```
JFrame f = new JFrame("title");
JPanel p = new JPanel( );
JButton b = new JButton("press me");

b.setBounds(new Rectangle(10,10, 100,50));
p.setLayout(null);          // x,y layout
p.add(b);
f.setContentPane(p);
```

Exemplu: FlowLayout

```
JFrame f = new JFrame("title");
JPanel p = new JPanel( );
FlowLayout L = new FlowLayout( );
JButton b1 = new JButton("press me");
JButton b2 = new JButton("then me");

p.setLayout(L);
p.add(b1);
p.add(b2);
f.setContentPane(p);
```

Setarea *layout manager* se face inainte de adaugarea componentelor in container.

Gestionarea evenimentelor

Evenimente grafice

- Componentele grafice pot produce eveniment
- Evenimente tipice:
  - Mouse movements
  - Mouse clicks
  - Hitting any key
  - Hitting return key
  - etc.
- *Gestionarea evenimentelor este mecanismul prin care specificam ce cod sa se execute in momentul in care s-a produs un eveniment graphic.*

ActionEvent

- Marea majoritatea e componentelor au un eveniment *ActionEvent*
- Pentru a intercepta *ActionEvent* trebuie sa inregistram in cadrul componentei un *ActionListener.*
  - button.addActionListener(new MyAL())
- *ActionListener* contine o metoda actionPerfomed a carei semnatura este:
  - void actionPerformed(ActionEvent)
- Metode utile in *ActionEvent:*
  - *getSource() – obiectul care a generat evenimentul*
  - *getActionCommand() – text asociat cu evenimentul*

Exemple Simple

GUI Simplu

```
import javax.swing.JFrame;
class SimpleGUI extends JFrame{
        SimpleGUI(){
            setSize(400,400); //set frames size in pixels
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            show();
        }

        public static void main(String[] args){
            SimpleGUI gui = new SimpleGUI();
            System.out.println(“main thread coninues”);
        }
}
```

GUI Simplu - continuare

```
import javax.swing.*;
class SimpleGUI extends JFrame{
        SimpleGUI(){
          setSize(400,400); //set frames size in pixels
          setDefaultCloseOperation(EXIT_ON_CLOSE);
          JButton but1 = new JButton(“Click me”);
           Container cp = getContentPane();//must do this
          cp.add(but1);
            show();
      }

      public static void main(String[] args){
          SimpleGUI gui = new SimpleGUI();
          System.out.println(“main thread coninues”);
      }}
```

Adaugare Layout Manager

```
import javax.swing.*; import java.awt.*;
class SimpleGUI extends JFrame{
        SimpleGUI(){
          setSize(400,400); //set frames size in pixels
          setDefaultCloseOperation(EXIT_ON_CLOSE);
          JButton but1 = new JButton("Click me");
           Container cp = getContentPane();//must do this
          cp.setLayout(new FlowLayout(FlowLayout.CENTER);
            cp.add(but1);
            show();
      }

      public static void main(String[] args){
          SimpleGUI gui = new SimpleGUI();
          System.out.println("main thread coninues");
      }}
```

Interceptare eveniment ActionEvent

```
import javax.swing.*; import java.awt.*;
class SimpleGUI extends JFrame{
        SimpleGUI(){
          setSize(400,400); //set frames size in pixels
          setDefaultCloseOperation(EXIT_ON_CLOSE);
          JButton but1 = new JButton(“Click me”);
           Container cp = getContentPane();//must do this
          cp.setLayout(new FlowLayout(FlowLayout.CENTER);
            but1.addActionListener(new MyActionListener());
            cp.add(but1);
            show();
      }
        public static void main(String[] args){
          SimpleGUI gui = new SimpleGUI();
          System.out.println(“main thread coninues”);
      }}
```

Codul pentru interceptarea evenimentului

```
class MyActionListener implements ActionListener{
   public void actionPerformed(ActionEvent ae){
        JOptionPane.showMessageDialog(“I got clicked”, null);
   }



}
```

Adaugarea unui nou buton

```
class SimpleGUI extends JFrame{
        SimpleGUI(){
            /* .... */
            JButton but1 = new JButton(“Click me”);
            JButton but2 = new JButton(“exit”);
            MyActionListener al = new MyActionListener();
            but1.addActionListener(al);
            but2.addActionListener(al);
            cp.add(but1);
             cp.add(but2);
            show();
        }
}
```

Diferenteiere intre evenimentele celor 2 butoane – metoda 1

```
class MyActionListener implents ActionListener{
    public void actionPerformed(ActionEvent ae){
      if (ae.getActionCommand().equals(“Exit”){
          System.exit(1);
       }
       else if (ae.getActionCommand().equals(“Click me”){
           JOptionPane.showMessageDialog(null, “I’m clicked”);
       }


}
```

Diferenteiere intre evenimentele celor 2 butoane – metoda 2

```
class MyActionListener implents ActionListener{
    public void actionPerformed(ActionEvent ae){
      if (ae.getSource()  ==  but2){
          System.exit(1);
       }
       else if (ae.getSource() == but1){
           JOptionPane.showMessageDialog(null, “I’m clicked”);
       }
}
```

Question: How are but1, but2 brought into scope to do this?
Question: Why is this better?

Demo NetBeans UI Designer