



Προγραμματισμός II (Java)

7. Γραφικές Διεπαφές

Περιεχόμενα

- Γραφικά περιβάλλοντα (GUI)
- Abstract Windowing Toolkit (AWT)
 - Containers
 - Components
 - Layout managers
 - (Listeners)
- Swing
 - Αρχιτεκτονική Model-View-Controller
 - Διάφορα στοιχεία του Swing
 - Παράθυρα και μενού
 - Περιγράμματα
 - Τοποθέτηση στοιχείων στο παράθυρο
- Διαχειριστές τοποθέτησης
 - Βασικά interfaces
 - Τάξεις

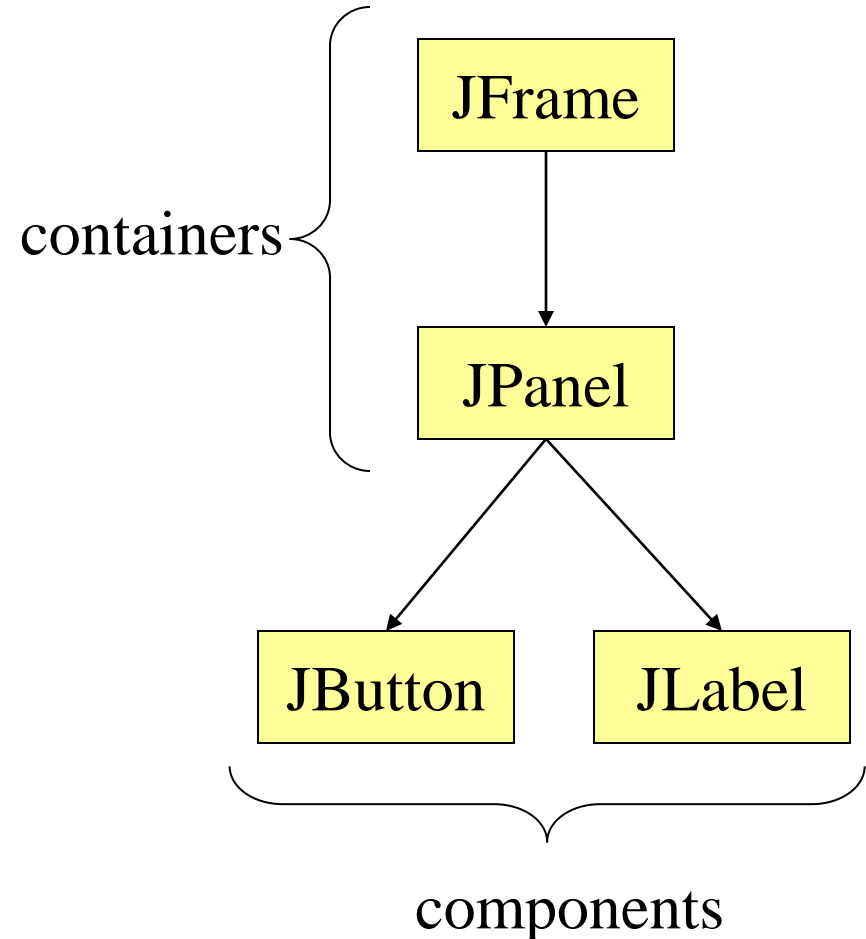
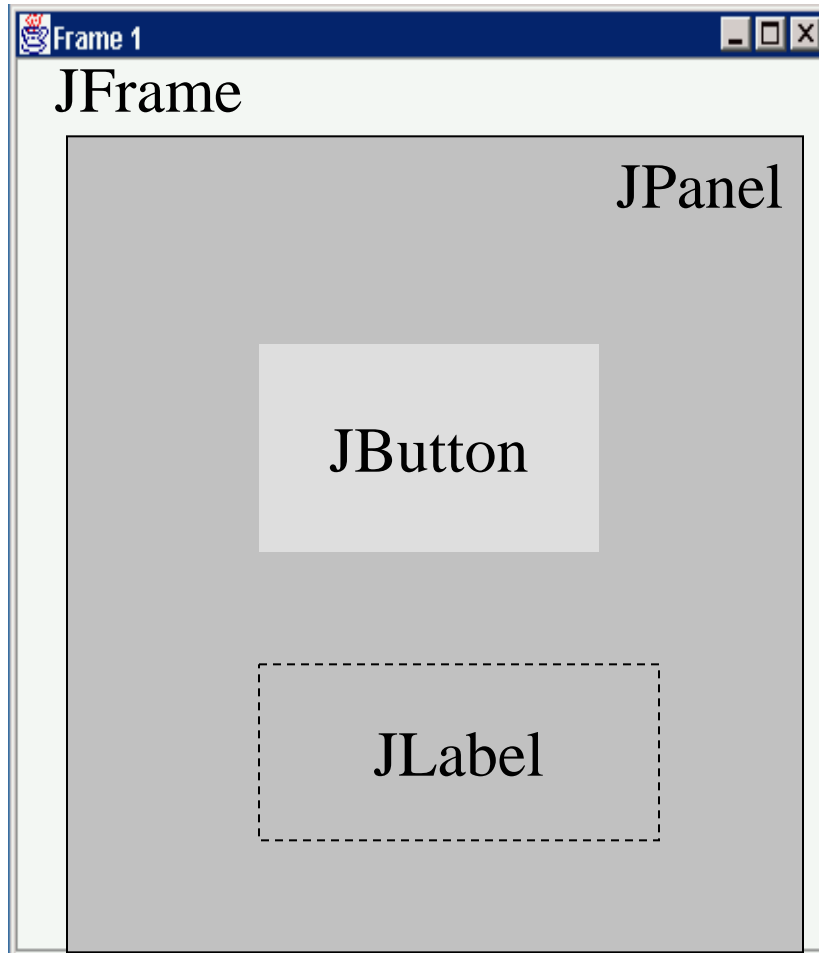
Περιεχόμενα

- Στοιχεία
 - Περιοχές κειμένου
 - Κουμπιά
 - Λίστες επιλογών
- Παράθυρα διαλόγου
 - Παράθυρα επιβεβαίωσης
 - Παράθυρο επιλογής αρχείου
 - Παράθυρο επιλογής χρώματος

Τα μέρη ενός γραφικού περιβάλλοντος

GUI

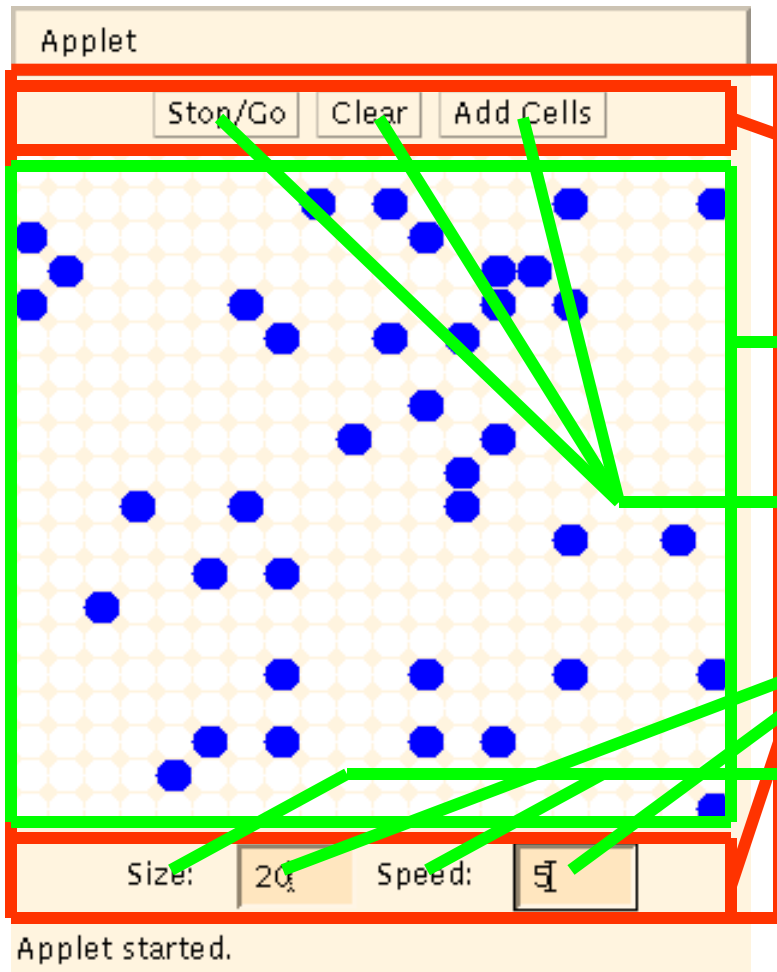
Σύνθεση τάξεων



Abstract Windowing Toolkit (AWT)

- Εμφανίστηκε στην αρχική έκδοση της Java αλλά υποστηρίζεται μέχρι σήμερα
- ***java.awt.**, *java.awt.event.****
- Περιλαμβάνει τάξεις για τη σχεδίαση ενός βασικού γραφικού περιβάλλοντος:
 - Παράθυρα (Containers): Frame, Window, Panel, Applet
 - Στοιχεία (Components): Button, Checkbox, Label, Scrollbar, TextField, TextArea
 - Διαχειριστές (LayoutManagers): FlowLayout, BorderLayout, GridLayout
 - Διεπαφές για ακρόαση γεγονότων (Listener interfaces): ActionListener, TextListener κλπ.

Παράδειγμα - Containers



Container (Applet)

Containers (Panels)

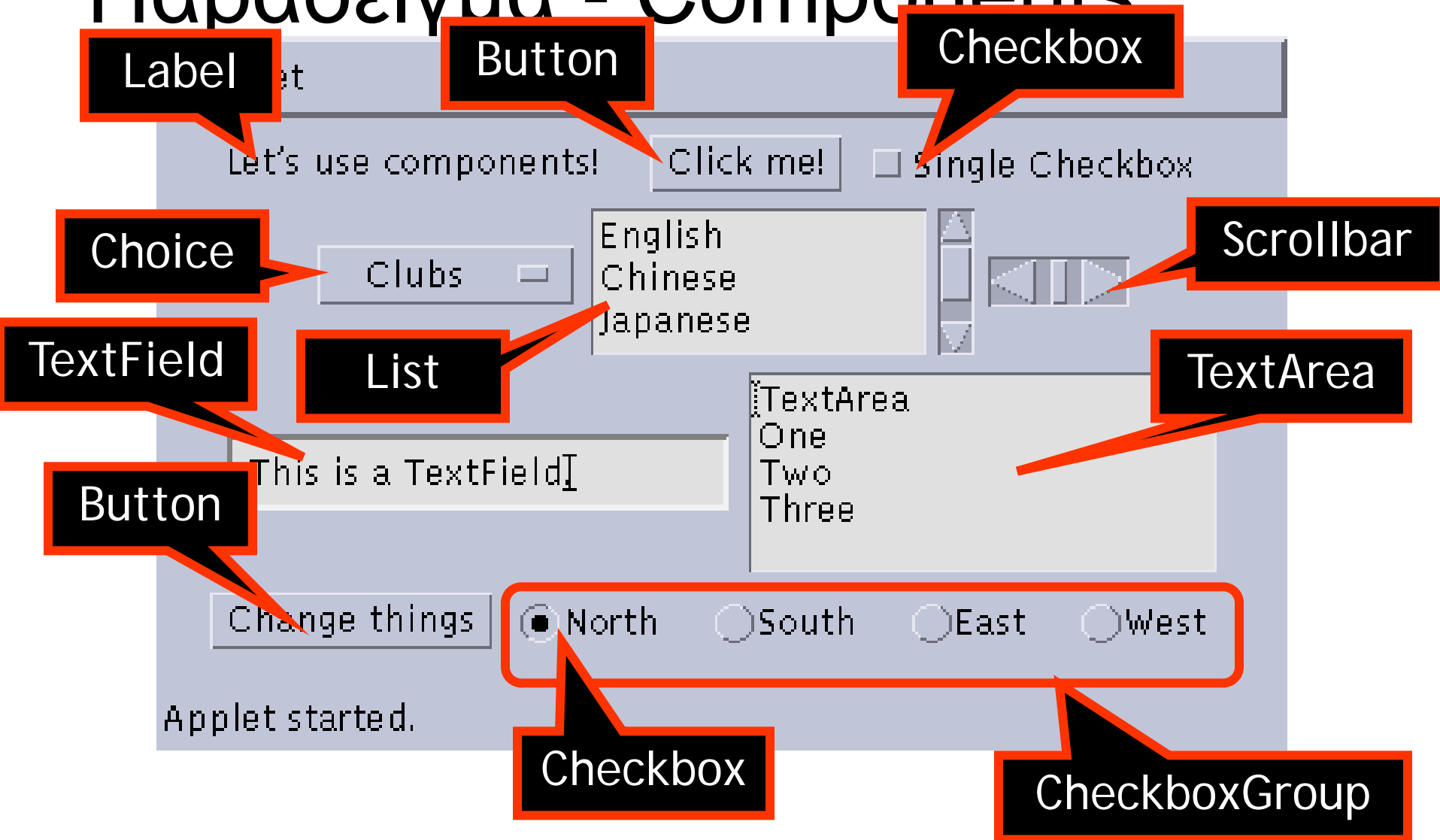
Component (Canvas)

Components (Buttons)

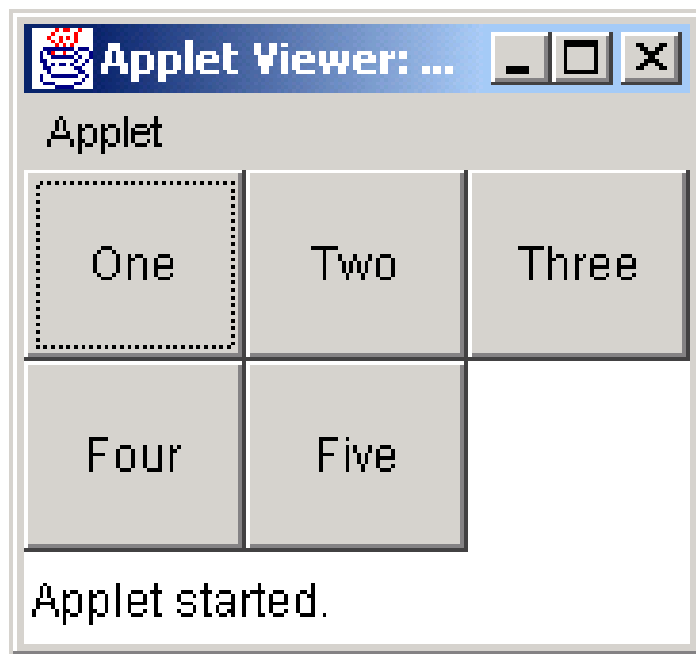
Components (TextFields)

Components (Labels)

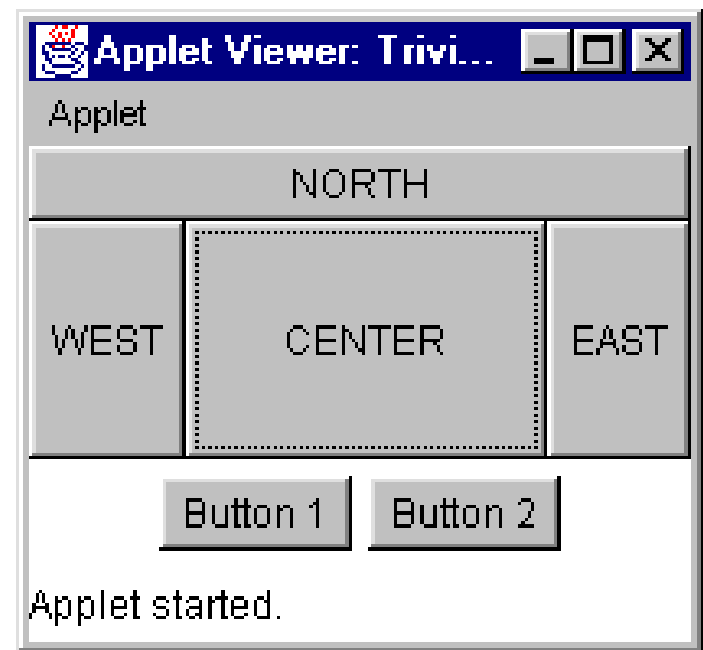
Παράδειγμα - Components



Παράδειγμα - Layout Managers



FlowLayout



BorderLayout

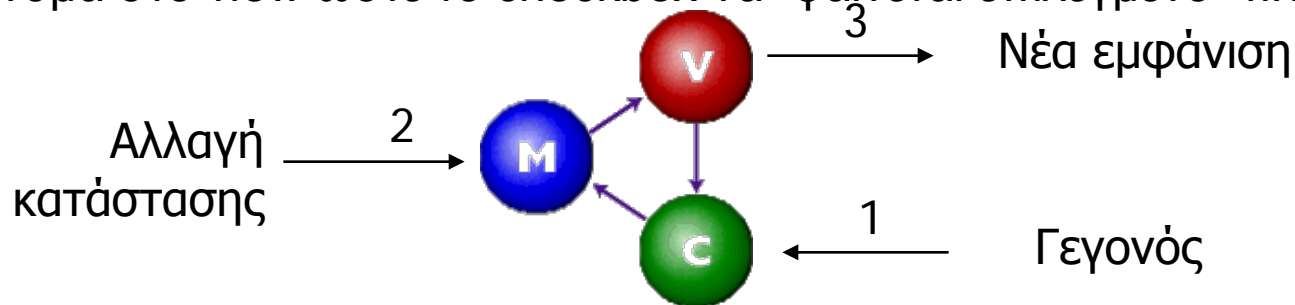
Swing

- Επεκτείνει το AWT με περισσότερα στοιχεία (components)
- Τα στοιχεία είναι 100% υλοποιημένα σε Java
javax.swing.*
- Τα στοιχεία είναι "ελαφρά" δεν επικοινωνούν απευθείας με το γραφικό περιβάλλον του λειτουργικού, παρά μόνο μέσω των βασικών παραθύρων
- Η τάξη JFrame (του Swing) κληρονομεί την Frame και αυτή την Window (του AWT) συνδέοντας έτσι το JFrame με το λειτουργικό σύστημα
- Προσφέρει τη δυνατότητα για διαφορετική εμφάνιση του GUI (look and feel)
- Μπορεί να συνδυαστεί με το AWT αλλά θέλει προσοχή

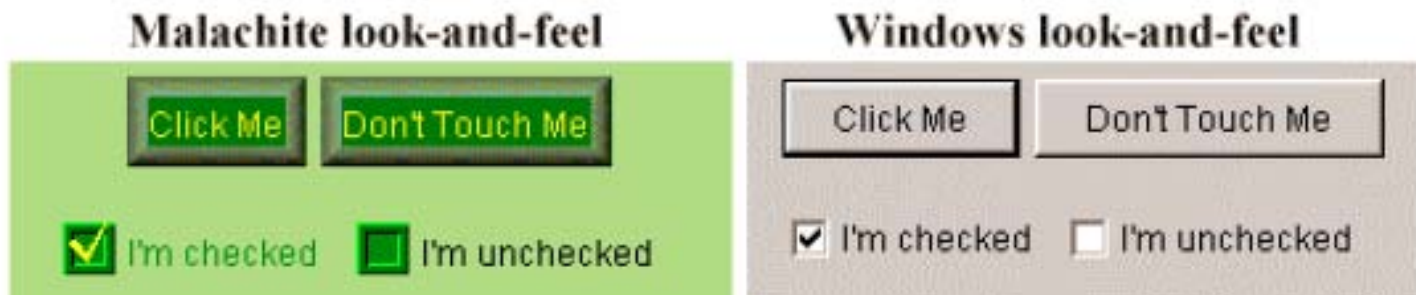
Model-View-Controller

- Ο controller ανιχνεύει γεγονότα, ενημερώνει το model που αποθηκεύει την κατάσταση του στοιχείου.

π.χ. Όταν επιλέξουμε ένα checkbox, ο controller ενημερώνει το model ώστε να ξέρει ότι το checkbox “είναι επιλεγμένο” και αυτό στέλνει ένα μήνυμα στο view ώστε το checkbox να “φαίνεται επιλεγμένο” πλέον.



- Μπορούμε έτσι να έχουμε για την ίδια λογική (Model) πάνω από διαφορετικά look (View) και feel (Controller)



Με ποια σειρά φτιάχνουμε το GUI

■ Δημιουργούμε

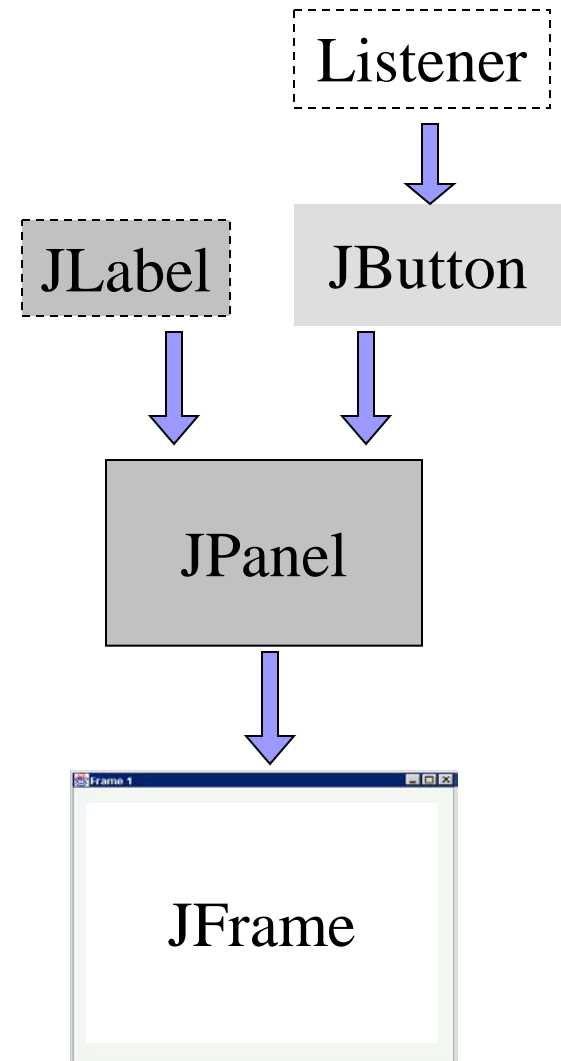
- ☐ To JFrame
- ☐ To JPanel
- ☐ Τα Components (JButton, JLabel)
- ☐ To Listener για το JButton

■ Προσθέτουμε (μέθοδος add)

- ☐ Τον Listener στο JButton
- ☐ Τα components στο JPanel
- ☐ Το JPanel στο JFrame

■ Εμφανίζουμε

- ☐ Το JFrame (μέθοδος show)

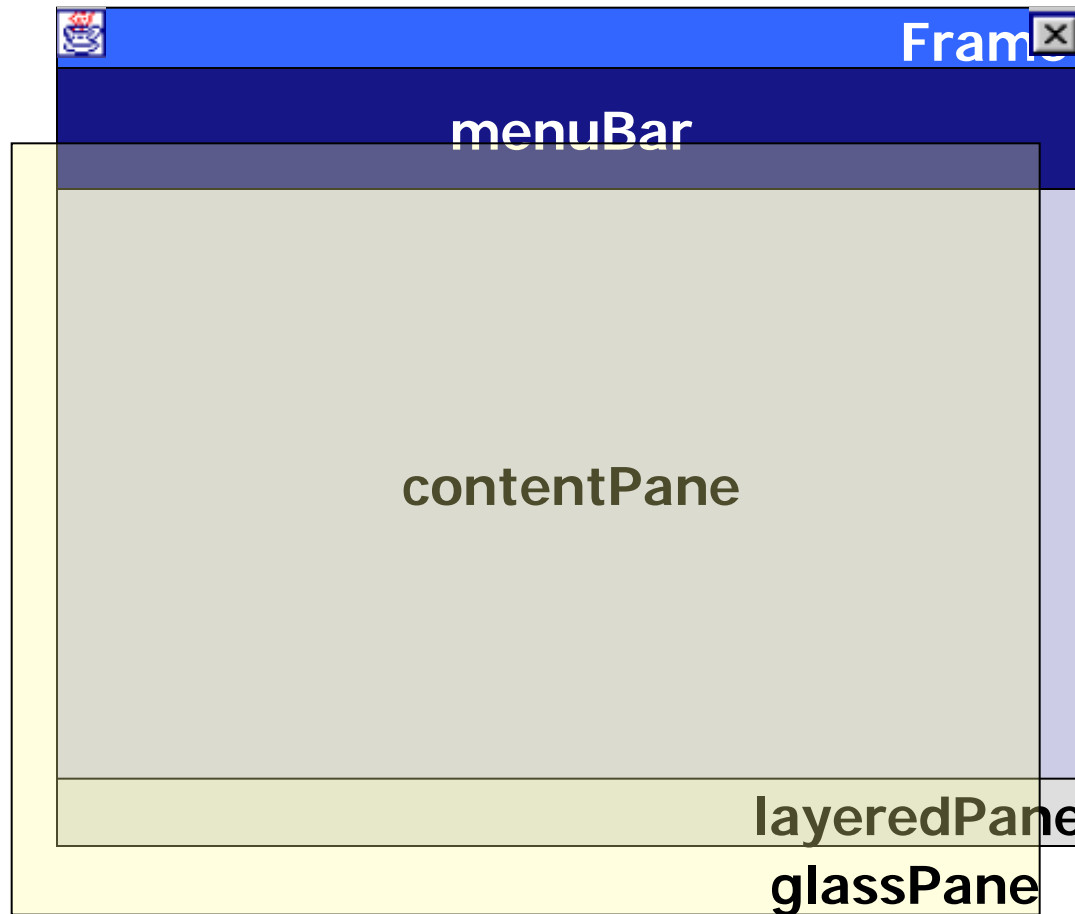


JFrame

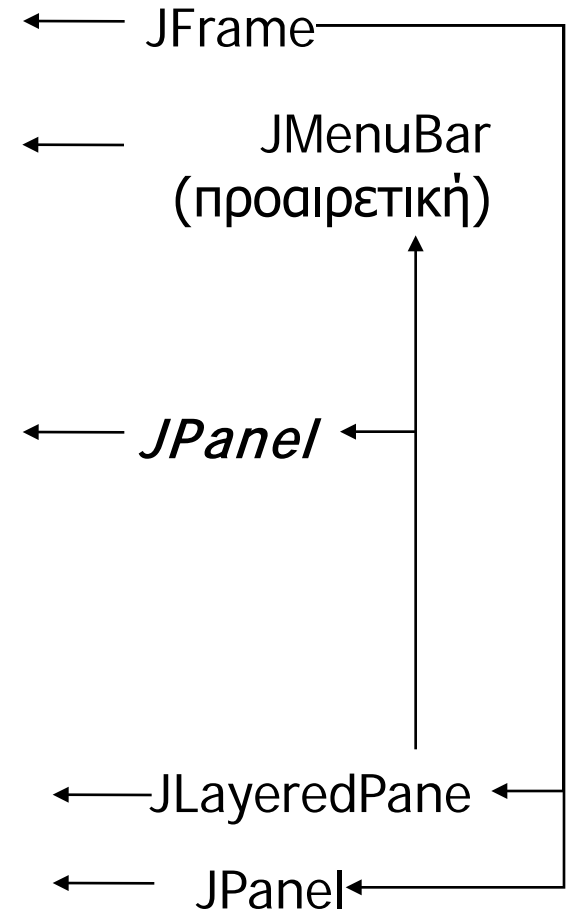
- Το βασικό container του swing
- Αποτελείται εξ' ορισμού από ένα container JRootPane
- Το JRootPane αποτελείται από το διάφανο glassPane και το ορατό layeredPane
- Το layeredPane περιέχει προαιρετικά ένα menuBar και το contentPane πάνω στο οποίο προστίθενται όλα τα υπόλοιπα στοιχεία.
- Συνεπώς, αφού φτιάξουμε το JFrame προσθέτουμε στο contentPane τα διάφορα components

```
JFrame f=new JFrame("title");  
f.getContentPane().add(myComponent);
```

Αντικείμενα



- Τάξεις



Παράδειγμα

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

```
...
```

```
JFrame f = new JFrame("title");
```

```
JButton b = new JButton("press me");
```

```
f.getContentPane() .add(b);
```

```
f.show();
```

press me

■ Παρόμοια

```
JFrame f = new JFrame("title");
```

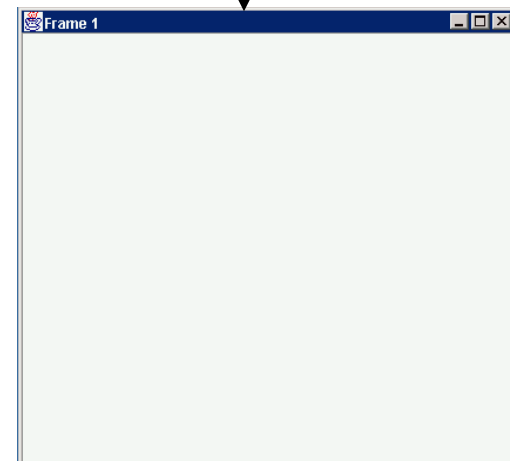
```
JPanel p = new JPanel( );
```

```
JButton b = new JButton("press me");
```

```
p.add(b);
```

```
f.setContentPane(p);
```

```
f.show();
```



To JMenuBar

- Η μπάρα μενού δεν είναι υποχρεωτική για ένα JFrame
- Μπορούμε να την προσθέσουμε φτιάχνοντας ένα JMenuBar αντικείμενο
JMenuBar menu = new JMenuBar();
- και αναθέτοντάς το στο JFrame
f.setJMenuBar(menu);

Άλλες παράμετροι ενός JFrame

- Εικόνα

```
ImageIcon image = new ImageIcon("spiral.gif");  
f.setIconImage(image.getImage());
```

- Αρχικό μέγεθος

```
f.setSize(100,100);
```

- Θέση εξ ορισμού (0,0) = πάνω αριστερά στην οθόνη

```
f.setLocation(50, 100);
```


- Μέγεθος και Θέση

```
f.setBounds(120,120,300,300);
```

- Ανάλυση οθόνης

```
Dimension dim = f.getToolkit().getScreenSize();  
int screenwidth=dim.width;  
int screenlength=dim.length;
```


Πώς κλείνουμε ένα JFrame

- Τι θα γίνεται όταν πατήσουμε το ;
- Ο πιο απλός τρόπος είναι να χρησιμοποιήσουμε τη μέθοδο `setDefaultCloseOperation(int)` της `JFrame` με όρισμα:
 - `WindowConstants.DISPOSE_ON_CLOSE` – κλείνει το frame
 - `WindowConstants.EXIT_ON_CLOSE` – κλείνει την εφαρμογή
 - `WindowConstants.DO_NOTHING_ON_CLOSE` – δεν κάνει τίποτε, οπότε ανιχνεύουμε διαφορετικά το γεγονός
 - `WindowConstants.HIDE_ON_CLOSE` – κρύβει το frame χωρίς να το καταστρέφει

f.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE)

Η `WindowConstants` είναι ένα interface που υλοποιεί η `JFrame`

JPanel

- Το JPanel είναι το βασικότερο container μέσα σε ένα JFrame.
- Ένα JPanel μπορεί να περιέχει components ή άλλα JPanel επιτρέποντας έτσι την καλύτερη οργάνωση των στοιχείων στο παράθυρο.
- Κάθε JPanel έχει ένα διαχειριστή τοποθέτησης στοιχείων (LayoutManager). Ο βασικός διαχειριστής προσθέτει τα στοιχεία στη σειρά το ένα δίπλα στο άλλο.
- Στα JPanel αλλά και στα υπόλοιπα components μπορούμε να ορίσουμε ένα τύπο περιγράμματος (Border)

Περιγράμματα - Borders

- Οι αντίστοιχες τάξεις βρίσκονται στο πακέτο javax.swing.border
- BevelBorder (υπερυψωμένο ή βυθισμένο), CompoundBorder (διπλό) EmptyBorder (διάφανο), TitledBorder κ.ά.

```
Panel p=new Panel();
```

```
BevelBorder bb = new BevelBorder (BevelBorder.RAISED);
```

```
p.setBorder(bb);
```



Τοποθέτηση components στο JPanel

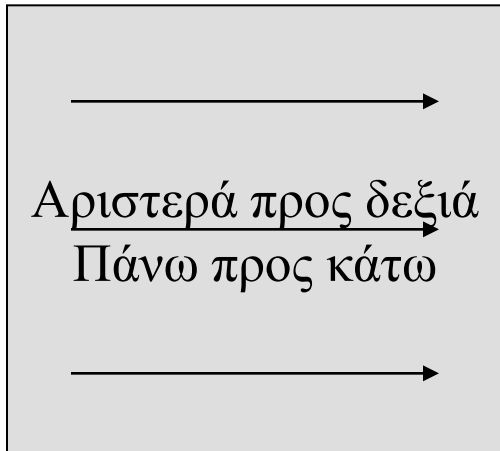
- Οι Layout Managers είναι τάξεις που διαχειρίζονται τον τρόπο τοποθέτησης των στοιχείων ενός παραθύρου
- Υλοποιούν δύο βασικά interfaces τα `LayoutManager` και `LayoutManager2`
- Κάποιοι περιέχονται στο `java.awt` ενώ κάποιοι άλλοι στο `javax.swing`
- Πρώτα ορίζουμε το layout σε ένα container και μετά προσθέτουμε στοιχεία.

Ορισμός Layouts

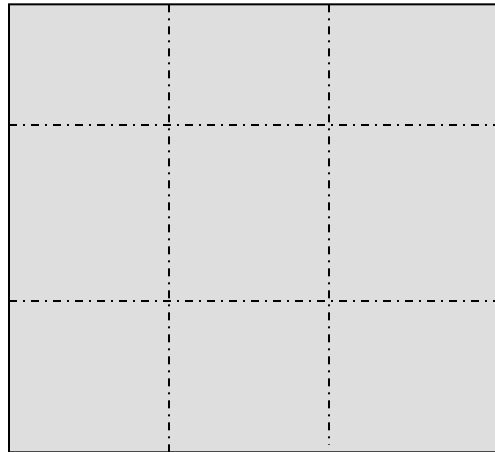
- Κάθε παράθυρο (container) έχει ένα προκαθορισμένο τρόπο τοποθέτησης στοιχείων (components)
 - Τα JPanel έχουν προκαθορισμένα FlowLayout
- Για κάθε παράθυρο ορίζουμε ένα layout με τη μέθοδο `setLayout`
myPanel.setLayout(new BorderLayout());
ή
this.getContentPane().setLayout(new GridLayout(2,2));

Ενδεικτικά layouts

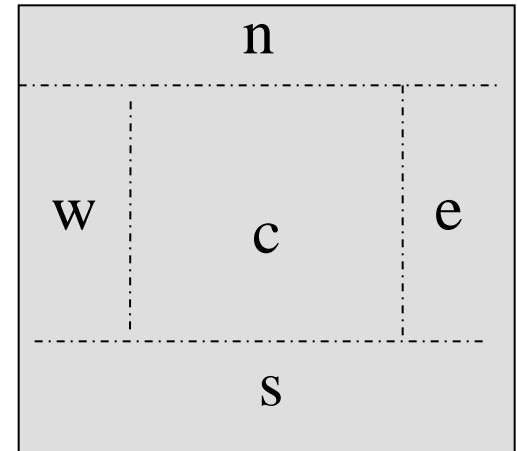
FlowLayout



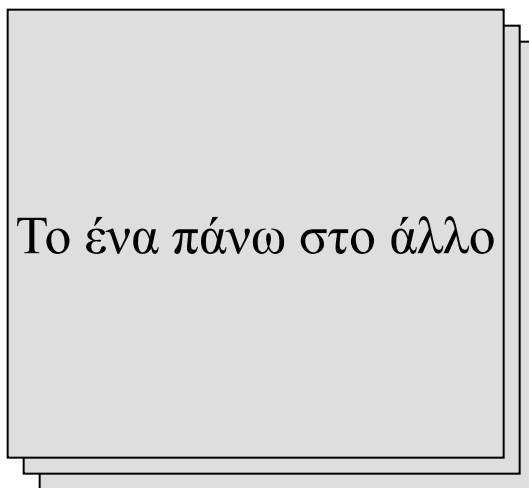
GridLayout



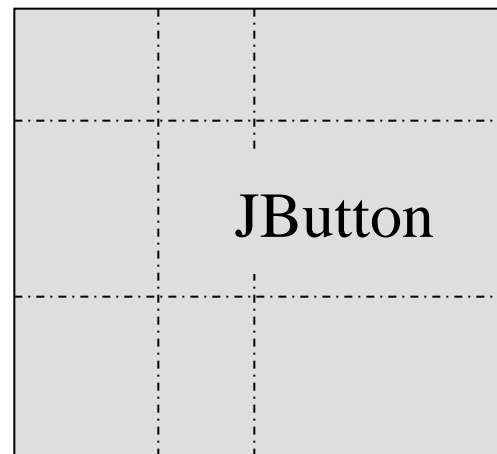
BorderLayout



CardLayout



GridBagLayout



Βασικές τάξεις τοποθέτησης

- `FlowLayout()`, τοποθετεί τα στοιχεία στη σειρά

```
myPanel.setLayout(new  
    FlowLayout(FlowLayout.CENTER));
```

```
myPanel.add(new JButton("1"));
```

- `GridLayout(int rows, int columns)`

```
myPanel.setLayout(new GridLayout(2,2));
```

```
myPanel.add(new JButton("1"));
```

- `BorderLayout()`

```
myPanel.setLayout(new BorderLayout());
```

```
add(new JButton("1"), BorderLayout.NORTH);
```



Βασικές τάξεις τοποθέτησης

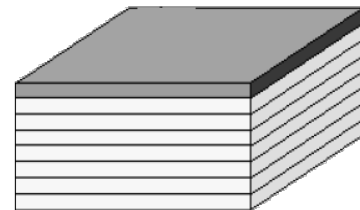
- `CardLayout()`, τοποθετεί το ένα στοιχείο πίσω από το άλλο. Ορατό είναι αυτό που προσθέτουμε πρώτο.

```
myPanel.setLayout(new CardLayout());  
myPanel.add(new JButton("1"),"first");  
myPanel.add(new JButton("2"),"second");
```

Με τις μεθόδους `first`, `last`, `next`, `previous`, `show` φέρνουμε στην επιφάνεια μια κάρτα

- `GridBagLayout()`, τοποθετεί τα στοιχεία σε θέσεις ενός πλέγματος, ώστε να γεμίζουν τη θέση ή να καταλαμβάνουν πάνω από ένα κελιά στο πλέγμα


```
GridBagLayout gbl=new GridBagLayout();  
myPanel.setLayout(gbl);  
GridBagConstraints c = new GridBagConstraints();  
c.fill=GridBagConstraints.BOTH;  
c.weightx=1.0; c.weighty=1.0;  
c.gridx=0; c.gridy=0;  
c.gridheight=2;  
myPanel.add(new JButton("one",c);
```



Τάξεις στο πακέτο swing

- **BoxLayout**, τοποθετεί τα στοιχεία σε μια σειρά (ή στήλη) χωρίς αναδίπλωση. Χρησιμοποιείται κυρίως για μπάρες εργαλείων.
- Συνδυάζεται με την τάξη Box που είναι Container

```
Container toolbar = Box.createHorizontalBox( );  
toolbar.add(new JButton("1"));  
myPanel.add(toolbar);
```

A horizontal toolbar with two buttons labeled '1' and '2'. The buttons are light gray with black text and are separated by a small gap. The toolbar itself is a light gray bar with a thin border.
- **SpringLayout**, καθορίζει την απόσταση κάθε στοιχείου από τα γειτονικά στοιχεία
- **OverlayLayout**, **ScrollPaneLayout**

Ειδικά πλαίσια (Panels)

- **JScrollPane**: Περιέχει ένα στοιχείο μόνο
- Εμφανίζει αυτόματα μπάρες κύλισης αν το στοιχείο δεν χωρά στο πλαίσιο

```
JScrollPane sp=new JScrollPane(new JButton("A button with a long label on it"));  
myPanel.add(sp);
```
- **JSplitPane**: Περιέχει δύο στοιχεία με ένα διαχωριστικό που μετακινείται.

```
JSplitPane split =new JSplitPane(  
    JSplitPane.HORIZONTAL_SPLIT, button1, button2);  
myPanel.add(split);
```



Ειδικά πλαίσια (Panels)

- ***JTabbedPane***: Τοποθετεί κάθε στοιχείο πίσω από τα υπόλοιπα με τη μορφή καρτελών
- Κάθε καρτέλα έχει ένα όνομα και ένα στοιχείο ή πλαίσιο

```
JPanel config=new JPanel();
```

```
JPanel options=new JPanel();
```

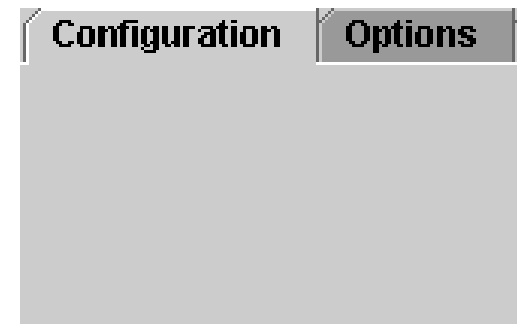
```
...
```

```
JTabbedPane tabby = new JTabbedPane( );
```

```
tabby.addTab("Configuration", config);
```

```
tabby.addTab("Options", options);
```

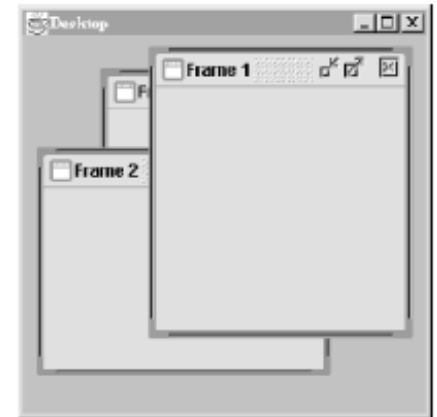
```
myPanel.add(tabby);
```



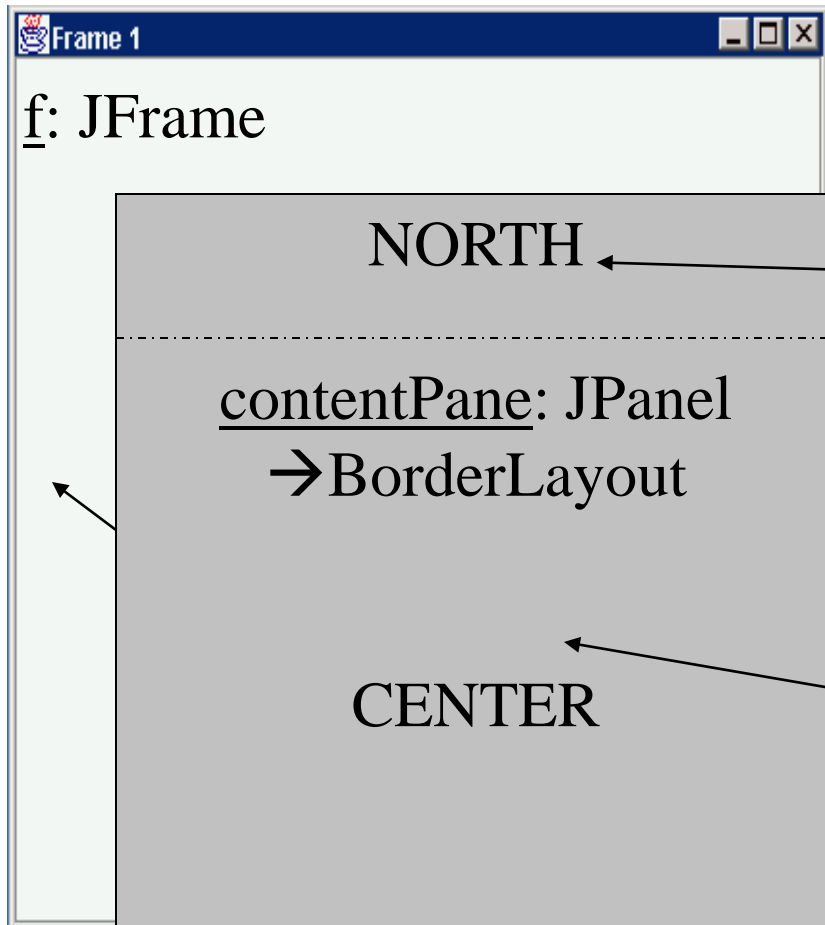
Εσωτερικά παράθυρα

- Η τάξη JDesktopPane μας επιτρέπει να εμφανίσουμε ολόκληρα παράθυρα (JInternalFrame) μέσα σε ένα JFrame
- Τα εσωτερικά παράθυρα μπορεί να είναι resizable, closable, maximizable, iconifiable, όπως ακριβώς συμβαίνει με τα παράθυρα των Windows

```
JDesktopPane desktop = new JDesktopPane( );  
JInternalFrame internal =  
new JInternalFrame("Frame 1", true, true, true, true);  
internal.setSize(180, 180);  
internal.setLocation(20,20);  
internal.setVisible(true);  
desktop.add(internal);  
f.setContentPane(desktop);
```



Συνδυασμοί panels και layouts



JButton

JButton

p1: JPanel → FlowLayout

t: JTextArea

Συνδυασμός

```
JFrame f = new JFrame("Frame1");
```

```
JPanel p1 = new JPanel( );
```

```
JButton b1 = new JButton("OK");
```

```
JButton b2 = new JButton("Cancel");
```

```
p1.add(b1);
```

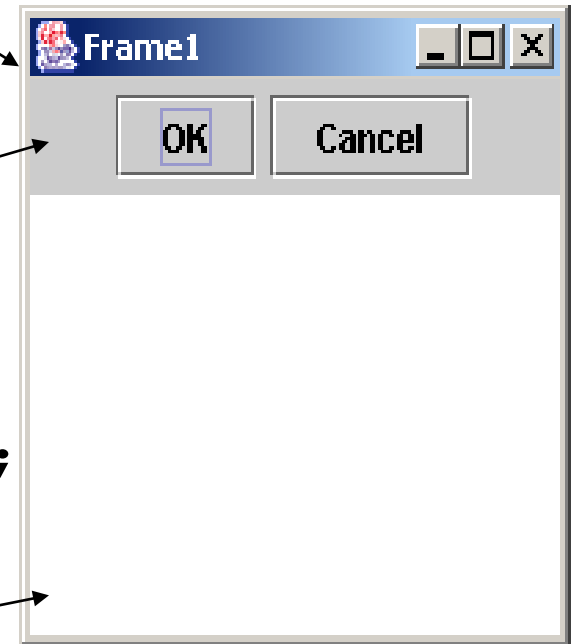
```
p1.add(b2);
```

```
JTextArea t=new JTextArea("");
```

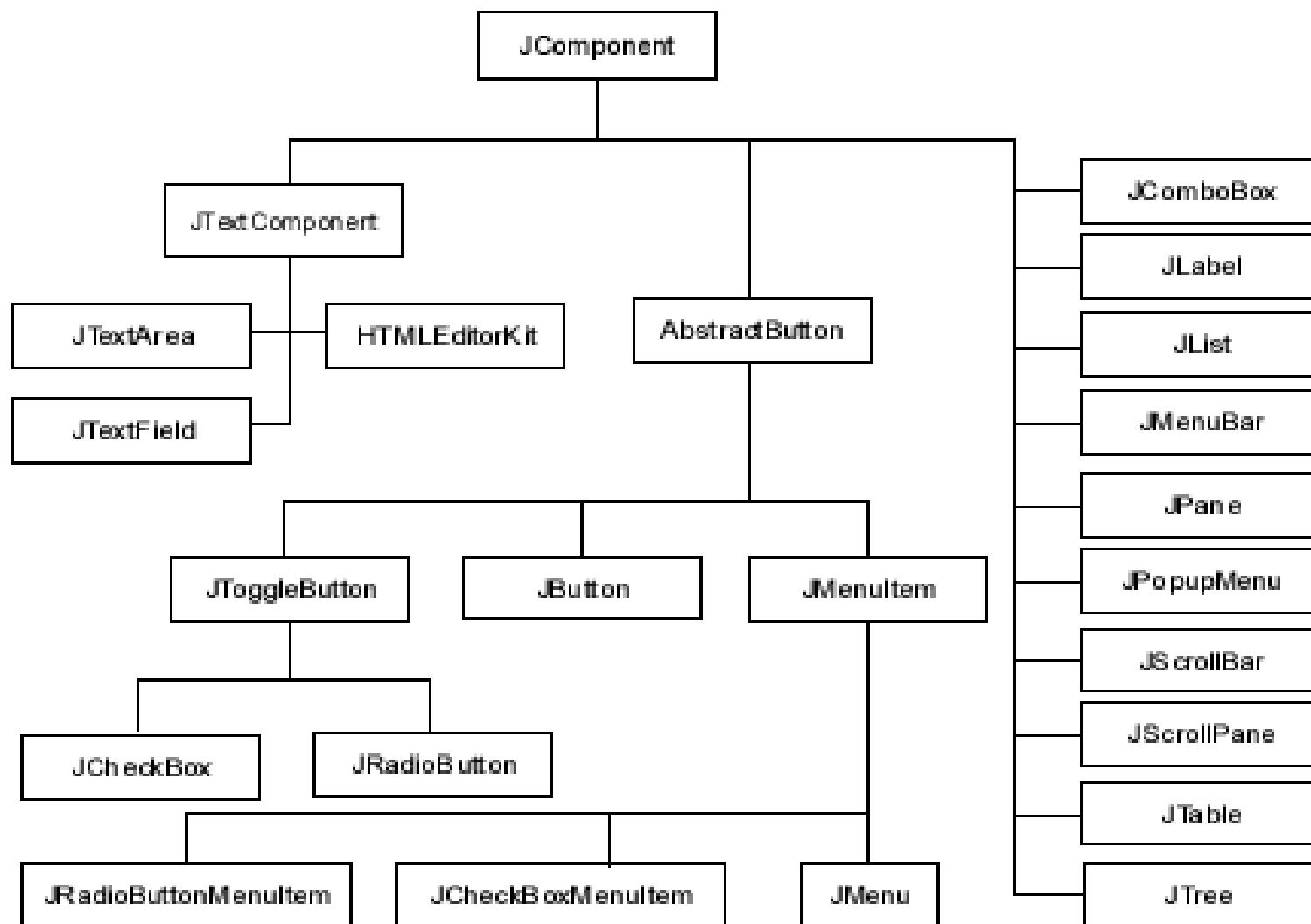
```
f.getContentPane().setLayout(new BorderLayout());
```

```
f.getContentPane().add(p1,BorderLayout.NORTH);
```

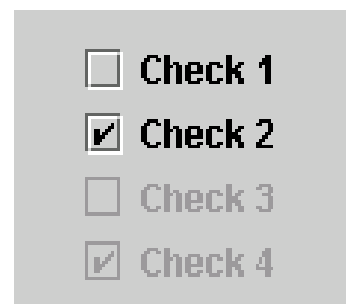
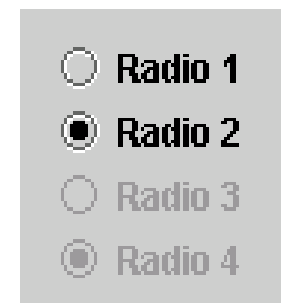
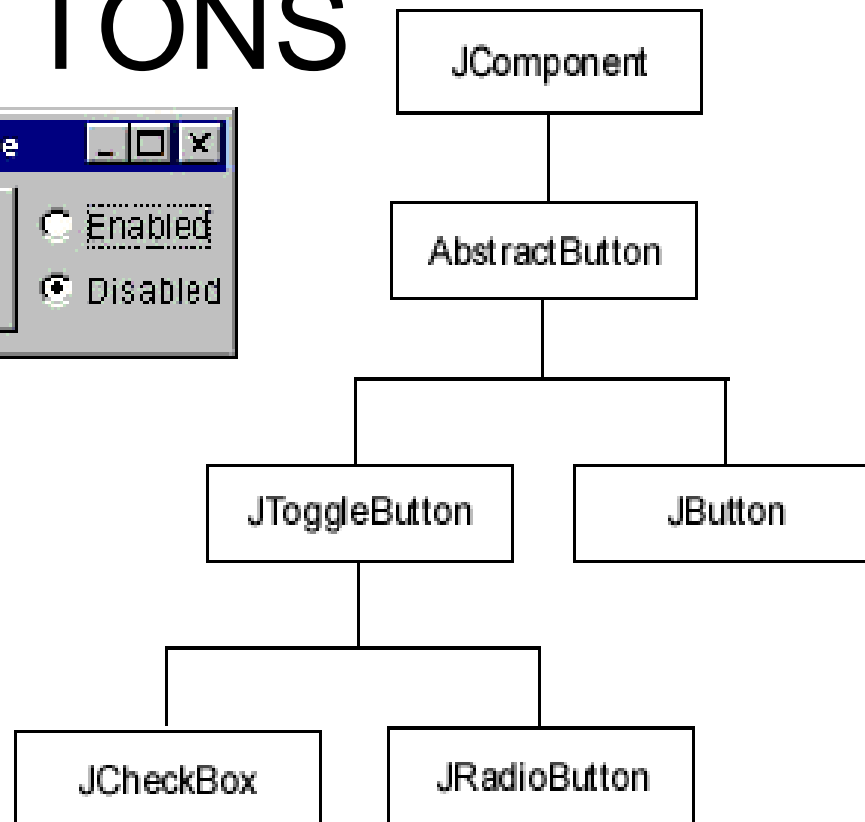
```
f.getContentPane().add(t,BorderLayout.CENTER);
```



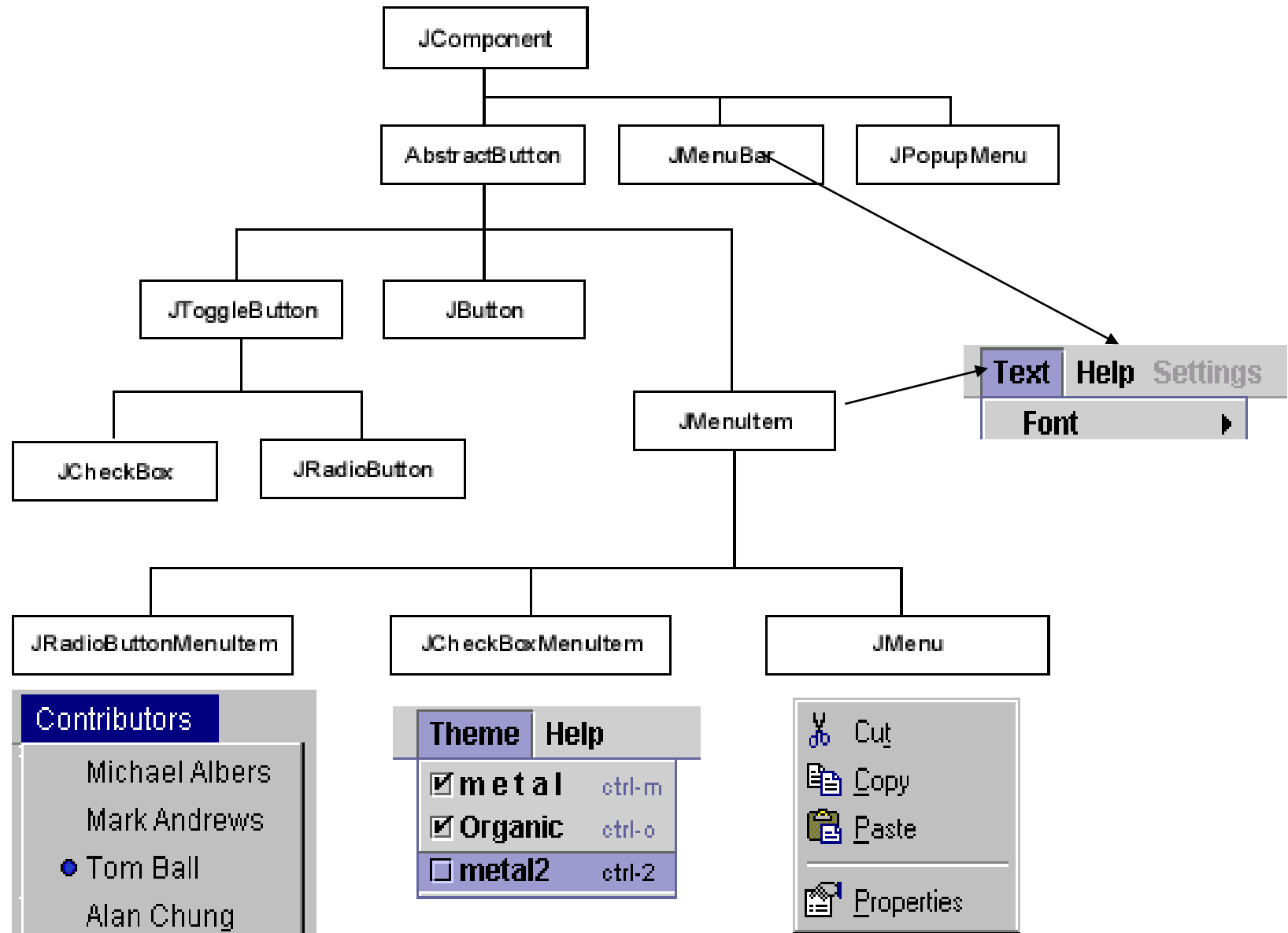
Στοιχεία του Swing



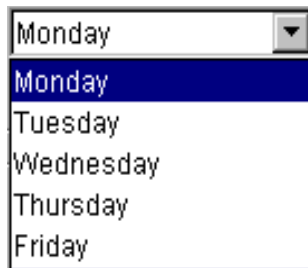
BUTTONS



MENUS



Άλλα COMPONENTS



JComboBox



JApplet



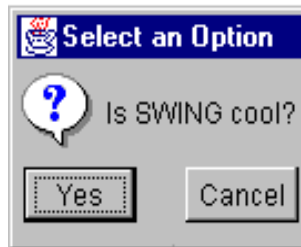
Border Interface



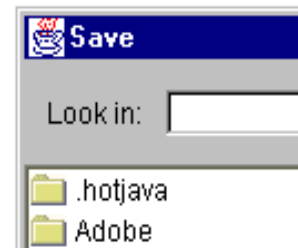
JColorChooser



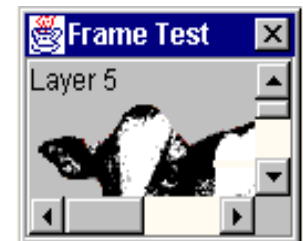
ImageIcon



JDialog



JFileChooser

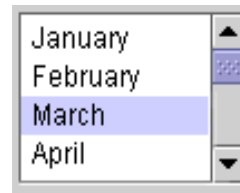


JInternalFrame

Άλλα COMPONENTS



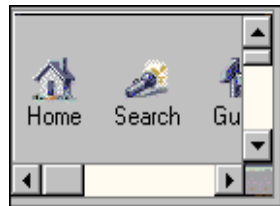
JLabel



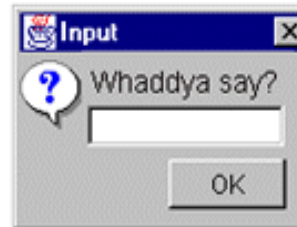
JList



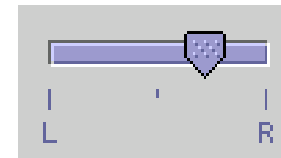
JScrollBar



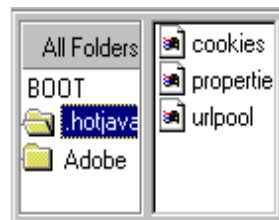
JScrollPane



JOptionPane



JSlider



JSplitPane



JTabbedPane

Άλλα COMPONENTS

First Na...	Last Name
Mark	Andrews
Tom	Ball
Alan	Chung
Jeff	Dinkins

JTable

Verify that the RJ45 cable is connected to the WAN plug on the back of the Pipeline unit.

JTextArea



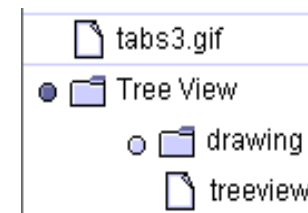
JToolBar



JToolTip

George Washington
Thomas Jefferson
Benjamin Franklin
Thomas Paine

JTextField



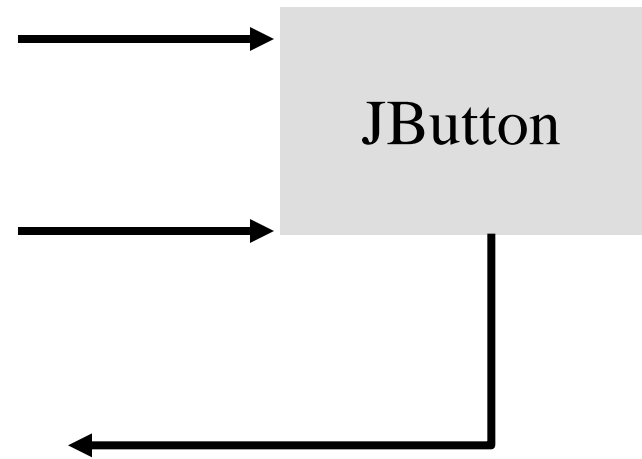
JTree

JComponent

- Κάθε JComponent είναι αντικείμενο μιας τάξης.

Συνεπώς:

- Έχει κατάσταση (ιδιότητες)
(π.χ. ενεργό, ορατό, επιλεγμένο, θέση, κείμενο, εικόνα κλπ.)
- Έχει μεθόδους
(π.χ. Όρισε κείμενο/εικόνα, φόντο)
- **Ανταποκρίνεται σε γεγονότα**
(π.χ. mouseClicked, mouseEntered, keyTyped, componentMoved)



Πώς χρησιμοποιούμε ένα Component

1. Το δημιουργούμε
 - `JButton b = new JButton("press me");`
2. Το ρυθμίζουμε
 - Ιδιότητες: `b.text = "press me";`
 - Μέθοδοι: `b.setText("press me");`
3. Του προσθέτουμε components (αν είναι container)
4. Τα προσθέτουμε σε ένα container (εκτός αν είναι JFrame)
 - `panel.add(b);`
5. Παρακολουθούμε τα γεγονότα σ' αυτό
 - Events: Listeners

ΕΤΙΚΕΤΕΣ

```
JLabel label1 = new JLabel("JLabel");  
JLabel label2 = new JLabel("JLabel", SwingConstants.CENTER);  
label2.setOpaque(true); label2.setBackground(Color.white);
```

```
JLabel label3 = new JLabel("JLabel", SwingConstants.CENTER);  
label3.setFont(new Font("Helvetica", Font.BOLD, 18));  
label3.setOpaque(true); label3.setBackground(Color.white);
```

```
ImageIcon icon = new ImageIcon("image.gif");  
JLabel label4 = new JLabel("JLabel", image, SwingConstants.RIGHT);  
label4.setVerticalTextPosition(SwingConstants.TOP);  
label4.setOpaque(true); label4.setBackground(Color.white);
```

```
myPanel.add(label); myPanel.add(label2);  
myPanel.add(label3); myPanel.add(label4);
```



Περιοχές κειμένου

```
JTextField tf1 = new JTextField();  
JTextField tf2 = new JTextField("text");  
JTextField tf3 = new JTextField("more text",40);  
tf2.getText();  
tf1.setText("empty");  
JTextArea ta= new TextArea();  
ta.append("text"); ta.append("\nline2");  
JPasswordField pf=new JPasswordField("hidden");
```

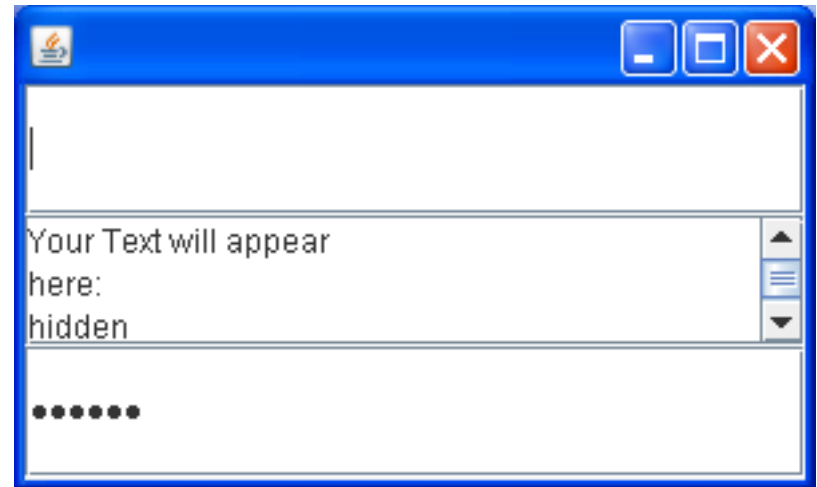


tetete text more text

text
line2

Περιοχές κειμένου

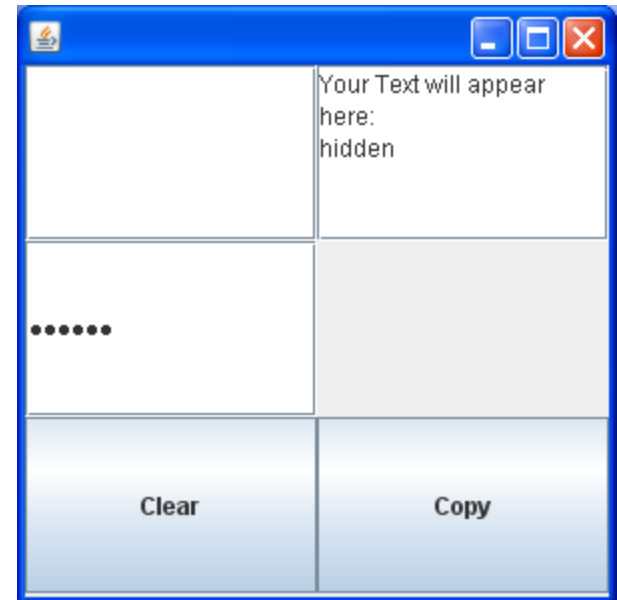
```
public class TestFrame extends JFrame {  
    JTextField tf;    JTextArea ta;  
    JPasswordField pf;    JScrollPane jsp;  
    public TestFrame() {  
        tf = new JTextField();    ta = new JTextArea();  
        ta.setText("Your Text will appear");  
        ta.append("\nhere:\n");  
        pf = new JPasswordField("hidden");  
        jsp = new JScrollPane();  
        jsp.getViewport().add(ta);  
        this.setLayout(new GridLayout(3, 3));  
        this.getContentPane().add(tf);  
        this.getContentPane().add(jsp);  
        this.getContentPane().add(pf);  
        String password=pf.getText();  
        ta.append(password);  
        this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);  
    }  
}
```



Κουμπιά

....

```
ta.append(password);  
clearButton=new JButton("Clear");  
copyButton = new JButton("Copy");  
this.getContentPane().add(new JPanel());  
this.getContentPane().add(clearButton);  
this.getContentPane().add(copyButton);
```



```
this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
```

```
}
```

Κουμπιά – Ομάδες κουμπιών

```
JButton b= new JButton("OK");
```

```
JToggleButton tb1= new JToggleButton("ON");
```

```
JToggleButton tb2= new JToggleButton("OFF");
```

```
ButtonGroup buttonGroup = new ButtonGroup();
```

```
tb1.setMnemonic('n'); // Ενεργοποιείται με ALT+n
```

```
tb1.setToolTipText("This is the ON button");
```

```
tb2.setMnemonic('f'); // Ενεργοποιείται με ALT+f
```

```
tb1.setToolTipText("This is the OF
```

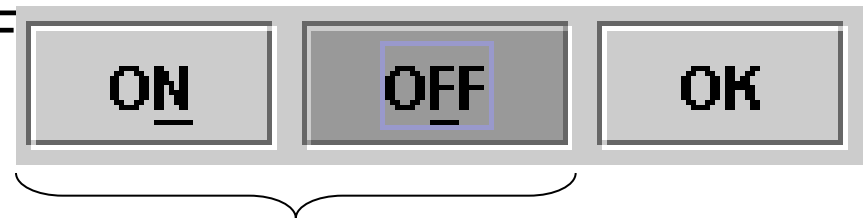
```
buttonGroup.add(tb1);
```

```
buttonGroup.add(tb2);
```

```
myPanel.add(tb1);
```

```
myPanel.add(tb2);
```

```
myPanel.add(b);
```



Ενεργοποιούνται εναλλάξ

JCheckBox και JRadioButton

- Τα JCheckBox λειτουργούν ανεξάρτητα και μπορούν να επιλεγούν όλα μαζί

```
myPanel.add(new JCheckBox("case 1"));
```

```
myPanel.add(new JCheckBox("case 2"));
```

Τα JRadioButton λειτουργούν σε ομάδες και μόνο ένα επιλέγεται κάθε φορά.

```
ButtonGroup options = new ButtonGroup( );
```

```
JRadioButton rb1= new JRadioButton("Option 1");
```

```
JRadioButton rb2= new JRadioButton("Option 2");
```

```
options.add(rb1); options.add(rb2);
```

```
myPanel.add(rb1); myPanel.add(rb2);
```



Λίστες επιλογών

- JComboBox: Περιέχει ένα πίνακα από επιλογές, εμφανίζει στο χρήστη μόνο μία και επιτρέπει μονές ή πολλαπλές επιλογές.

```
String [] items = { "uno", "due", "tre", "quattro", "cinque", "sei", "sette",  
    "otto", "nove", "deici", "undici" };
```

```
JComboBox comboBox = new JComboBox(items);  
comboBox.addItem("dodici");  
comboBox.getSelectedItem() //επιστρέφει Object  
comboBox.getSelectedObjects() //επιστρέφει Object[]
```
- JList: Περιέχει ένα πίνακα από επιλογές, εμφανίζει στο χρήστη ορισμένες από αυτές (ανάλογα με το ύψος της) και επιτρέπει μονές ή πολλαπλές επιλογές.

```
JList list = new JList(comboBox.getModel( ));  
list.getSelectedValues() //επιστρέφει Object[]
```
- Μοιράζονται το ίδιο μοντέλο δεδομένων



Παράδειγμα

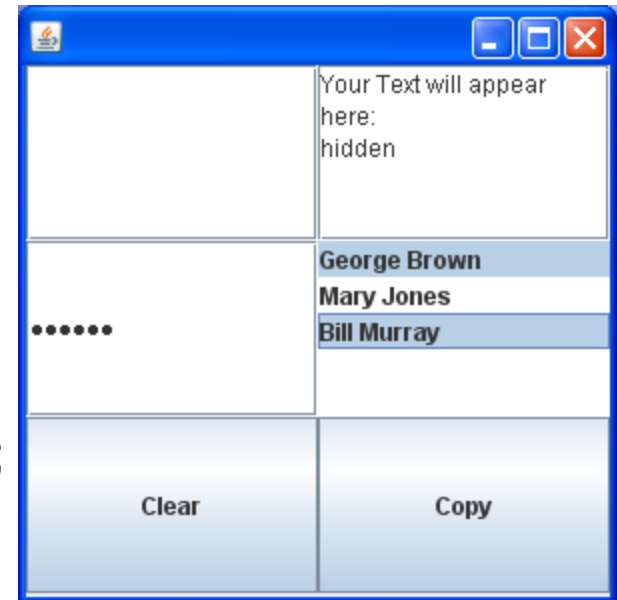
```
public class Human {  
    String name;  
    String surname;  
    int age;  
    String address;  
    public Human(String name, String surname, int age, String address) {  
        this.name = name;  
        this.surname = surname;  
        this.age = age;  
        this.address = address;  
    }  
    @Override  
    public String toString() {  
        return this.name+" "+this.surname;  
    }  
}
```

Μοντέλα Δεδομένων για Λίστες

```
dml = new DefaultListModel();  
dml.addElement(new Human("George", "Brown", 22, "6th Avenue"));  
dml.addElement(new Human("Mary", "Jones", 18, "5th Avenue"));  
dml.addElement(new Human("Bill", "Murray", 19, "Madison Avenue"));  
students = new JList(dml);  
this.getContentPane().add(students);  
this.getContentPane().add(clearButton);  
this.getContentPane().add(copyButton);
```

- Φαίνεται ότι επιστρέφει η toString()
- Επιστρέφει όλο το Object

```
Object[] selected=students.getSelectedValues();  
for (int i=0;i<selected.length;i++)  
    Human h=(Human)selected[i];
```



Πίνακες - JTable

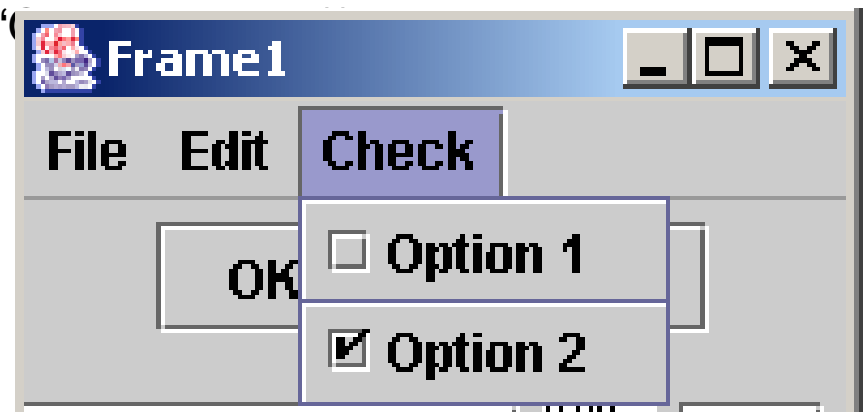
```
String[] columnNames = {"First Name", "Last Name", "Age", "Address"};
Object[][] data = {{"George", "Brown", new Integer(22), "6th Avenue"},
                  {"Mary", "Jones", new Integer(18), "5th Avenue"},
                  {"Bill", "Murray", new Integer(19), "Madison Avenue"}};
dtm = new DefaultTableModel(data, columnNames);
studentsTable = new JTable(dtm);
JScrollPane jsp2 = new JScrollPane();
jsp2.getViewport().add(studentsTable);
this.getContentPane().add(jsp2);
```

First Name	Last Name	Age	Address
George	Brown	22	6th Avenue
Mary	Jones	18	5th Avenue
Bill	Murray	19	Madison Av...

```
JTable(Object[][] rowData, Object[] columnNames);
JTable(Vector rowData, Vector columnNames);
JTable(DefaultTableModel dtm);
```


Μενού επιλογών

```
JMenu file = new JMenu("File");  
file.add(new JMenuItem("Open"));  
file.add(new JMenuItem("Close"));  
JMenu edit = new JMenu("Edit");  
edit.add(new JMenuItem("Copy"));  
edit.add(new JMenuItem("Paste"));  
JMenu check = new JMenu("Check");  
check.add(new JCheckBoxMenuItem("Option 1"));  
check.add(new JSeparator());  
check.add(new JCheckBoxMenuItem("Option 2"));  
MenuBar mb = new MenuBar();  
mb.add(file);  
mb.add(edit);  
mb.add(check);  
myFrame.setJMenuBar(mb);
```



Μελέτη Περίπτωσης

1	2	3	4	5	
Short Description	<input type="text"/>			<input type="button" value="submit"/>	1
Description	<input type="text"/>			<input type="button" value="Cancel"/>	2
				<input type="button" value="Help"/>	3
Severity	<input type="text"/>	Priority	<input type="text"/>		4
Name	<input type="text"/>				5
Telephone	<input type="text"/>				6
Sex	<input type="radio"/> Male	<input type="radio"/> Female			7
ID Number	<input type="text"/>				8

Μελέτη Περίπτωσης



A screenshot of a software window titled "Simple Complaints Dialog". The window has a standard Windows-style title bar with a close button (X) in the top right corner. The main area is divided into several sections. On the left, there are labels for "Short Description", "Description", "Severity", "Name", "Telephone", "Sex", and "ID Number". To the right of these labels are input fields: a single-line text box for "Short Description", a large multi-line text area for "Description", a dropdown menu for "Severity" showing "A", a dropdown menu for "Priority" showing "1", a single-line text box for "Name", a single-line text box for "Telephone", radio buttons for "Sex" with "Male" selected and "Female" unselected, and a single-line text box for "ID Number". On the far right, there are three buttons stacked vertically: "Submit", "Cancel", and "Help".

Simple Complaints Dialog

Short Description

Description

Severity **A** ▼ Priority **1** ▼

Name

Telephone

Sex ☒ Male ☐ Female

ID Number

Submit

Cancel

Help



Παράθυρα διαλόγου

Παράθυρα διαλόγου

- Τα παράθυρα διαλόγου χρησιμοποιούνται για να συλλέξουμε πληροφορίες από το χρήστη
 - Τραβούν το ενδιαφέρον του χρήστη καθώς εμφανίζονται πάνω από την εφαρμογή (π.χ. εισαγωγή κωδικού)
 - Απαιτούν εισαγωγή δεδομένων από το χρήστη και αποδοχή της επιλογής
 - Επιτρέπουν στο χρήστη να κάνει σύνθετες επιλογές και επιστρέφουν στην εφαρμογή το αποτέλεσμα της επιλογής (π.χ. επιλογή μιας ομάδας αρχείων και τις δράσης σε αυτά, επιλογή ενός χρώματος για το φόντο ενός πλαισίου κλπ)

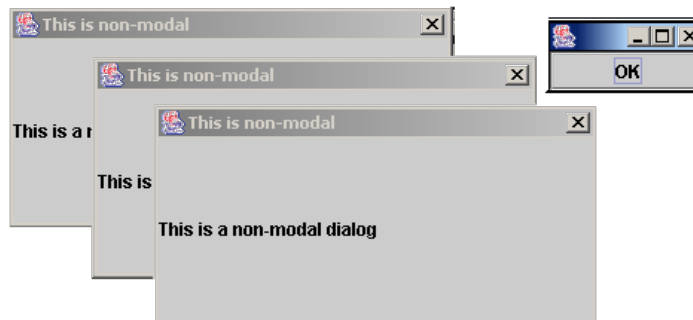
Το παράθυρο JDialog

- Συμπεριφέρεται όπως και το JFrame

```
JDialog dialog = new JDialog(myFrame, "Dialog Frame");  
JLabel label = new JLabel("This is a message");  
dialog.getContentPane().add(label);  
dialog.setVisible(true);
```

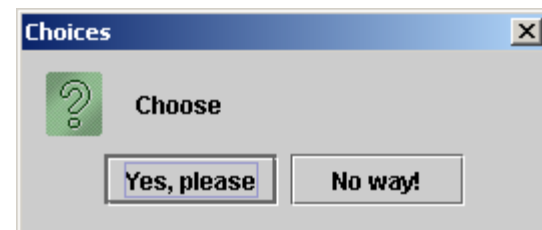
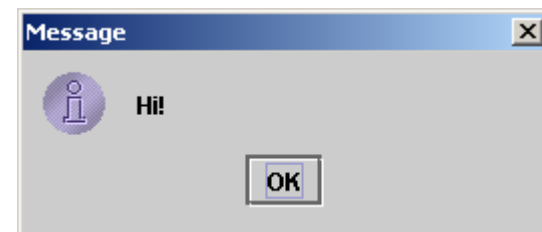
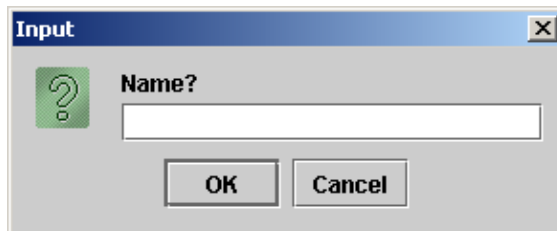
- Μπορεί να μπλοκάρει τη συνέχιση του προγράμματος (modal) ή όχι (non-modal)

```
JDialog dialog1 = new JDialog(myFrame, "This is modal", true);  
JDialog dialog2 = new JDialog(myFrame, "This is non modal", false);
```



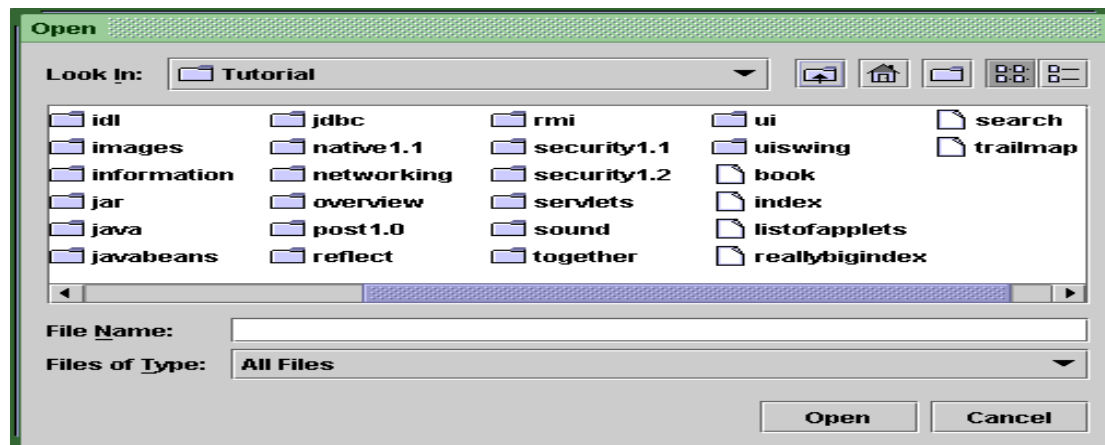
Έτοιμα πλαίσια διαλόγου

- Φτιάχνονται με κλήση των static μεθόδων της JOptionPane
 - `int n = JOptionPane.showConfirmDialog(myFrame, "Is it OK?", "ConfirmDialog", JOptionPane.YES_NO_OPTION);`
 - `String s = (String)JOptionPane.showInputDialog(myFrame, "Name?");`
 - `JOptionPane.showMessageDialog(myFrame, "Hi!");`
 - `Object[] options = {"Yes, please", "No way!"};`
`int n = JOptionPane.showOptionDialog(myFrame, "Choose", "Choices", JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE, null, options, options[0]);`



Παράθυρο επιλογής αρχείου

- Δημιουργία ενός JFileChooser:
`JFileChooser fc = new JFileChooser();`
- Εμφάνισή του (π.χ. με το πάτημα ενός JButton)
`int returnVal = fc.showOpenDialog(aComponent);`



Παράθυρο επιλογής αρχείου

- Στη συνέχεια μπορούμε να διαβάσουμε τις ενέργειες του χρήστη στο παράθυρο:

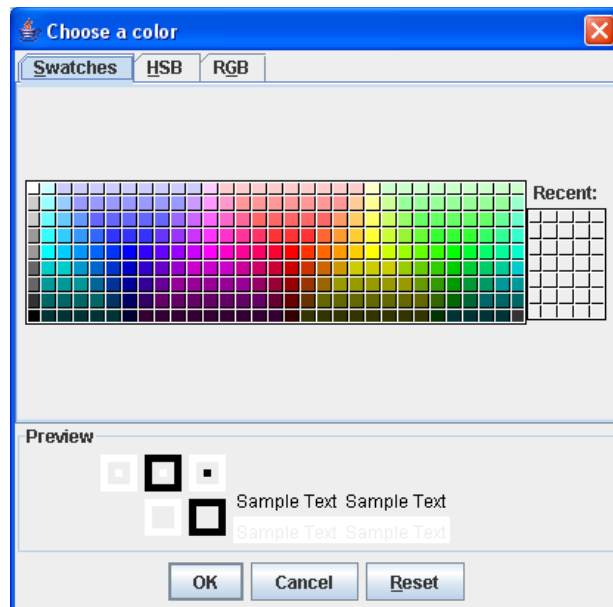
```
if (returnVal == JFileChooser.APPROVE_OPTION) {  
    // αν ο χρήστης επέλεξε Open  
    File file = fc.getSelectedFile(); //το αρχείο που επέλεξε  
    log.append("Opening: " + file.getName() + "." + newline);  
}  
else {  
    log.append("Open command cancelled by user." + newline);  
}
```
- `fc.showDialog(aComponent, "Save");`
//αλλάζει το open button σε Save

Παράθυρο επιλογής χρώματος

- Δημιουργία και χρήση ενός JColorChooser

```
Color c = JColorChooser.showDialog(myFrame, "Choose a  
color", myFrame.getContentPane().getBackground( ));
```

```
if (c != null) myFrame.getContentPane().setBackground(c);
```





Διαχείριση γεγονότων

Περιεχόμενα

- Διαχείριση γεγονότων
 - Δημιουργία γεγονότος
 - Ακρόαση (ανίχνευση) γεγονότος
 - Δημιουργία ακροατή – Κατηγορίες ακροατών
 - Σύνδεση ακροατή με στοιχείο
 - Πρακτικές χρήσης ακροατών

Διαχείριση γεγονότων

- *Το στοιχείο του GUI* (π.χ. ένα JButton)
 - Παράγει γεγονότα σε συγκεκριμένες συνθήκες
 - Ανάλογα με τις δραστηριότητες του χρήστη
- *Το γεγονός* (π.χ. ένα MouseEvent)
 - Ένα αντικείμενο που περιέχει πληροφορίες για το γεγονός
 - Ποιος το προκάλεσε, τι ακριβώς προκάλεσε, σε ποιο στοιχείο του GUI, κλπ
- *Οι ακροατές* (π.χ. ένας MouseListener)
 - Καταλαβαίνουν ένα γεγονός
 - Έχουν μεθόδους που παίρνουν γεγονότα ως όρισμα
 - Διάφορες κατηγορίες ακροατών (για γεγονότα σχετικά με τα mouse, keyboard, window, components κλπ)

Δημιουργία γεγονότων

- Τα αντικείμενα επικοινωνούν μεταξύ τους δημιουργώντας γεγονότα
- Τα γεγονότα είναι αντικείμενα της τάξης `java.util.EventObject` και των απογόνων της (π.χ. `AWTEvent`)
- Μεταφέρουν πληροφορία για το είδος του γεγονότος, τη δράση που το προκάλεσε κλπ. Για παράδειγμα
 - ένα `MouseEvent` παράγεται από ένα στοιχείο, όταν ο χρήστης κινήσει το ποντίκι στην περιοχή του στοιχείου (π.χ. ενός `JButton`). Το αντικείμενο `MouseEvent` περιλαμβάνει πληροφορία για τις συντεταγμένες του ποντικιού (x,y), για την κατάσταση των κουμπιών του κλπ.
 - Ένα `ActionEvent` μπορεί να παράγεται από το ίδιο στοιχείο (π.χ. `JButton`) ενημερώνοντας ότι κάτι συνέβη με το στοιχείο
- **Δεν παράγουν όλα τα στοιχεία τα ίδια γεγονότα**

Ακρόαση γεγονότων

- Το πάτημα ενός JButton δημιουργεί ένα γεγονός (αντικείμενο ActionEvent).
- Για να το χειριστούμε, θα πρέπει στο JButton να προσθέσουμε ένα τρόπο διαχείρισης γεγονότων.
- Οι μέθοδοι που διαχειρίζονται κάθε γεγονός περιέχονται στο αντίστοιχο interface. Τα interfaces αυτά ονομάζονται ακροατές – *listeners*.

```
public interface ActionListener extends java.util.EventListener {  
    public void actionPerformed( ActionEvent e );  
}
```

- Ένα στοιχείο μπορεί να έχει πολλούς ακροατές. Κάθε ακροατής μπορεί να παρακολουθεί πολλά στοιχεία.
- Συνοψίζοντας: α) Φτιάχνουμε την τάξη που υλοποιεί το κατάλληλο interface ακροατή, β) προσθέτουμε στο στοιχείο ένα αντικείμενο της τάξης αυτής.

A) Δημιουργία ακροατή

- Τα listener interfaces έχουν ένα κοινό πρόγονο το interface `java.util.EventListener` που δεν δηλώνει καμία μέθοδο.
- Ένα γεγονός έχει μοναδική πηγή αλλά μπορεί να ανιχνευτεί από πολλούς ακροατές (π.χ. ο ακροατής του στοιχείου, ο ακροατής του component που περιέχει το στοιχείο κλπ)
- Κάθε τάξη ακροατής γεγονότων θα πρέπει να υλοποιεί ένα ή περισσότερα listener interfaces.

```
public class myActionListener implements ActionListener{  
    public void actionPerformed( ActionEvent e ){  
        myTextArea.append("Something happened");  
    }  
}
```

- Εκτός από τα Listener interfaces υπάρχουν και οι αντίστοιχες Adapter τάξεις (είναι abstract)

```
public class myMouseListener implements MouseListener  
public class myMouseListener extends MouseAdapter
```


Τύποι ακροατών

Ακροατής	Μέθοδοι
ActionListener	actionPerformed(ActionEvent)
MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
ItemListener	itemStateChanged(ItemEvent)

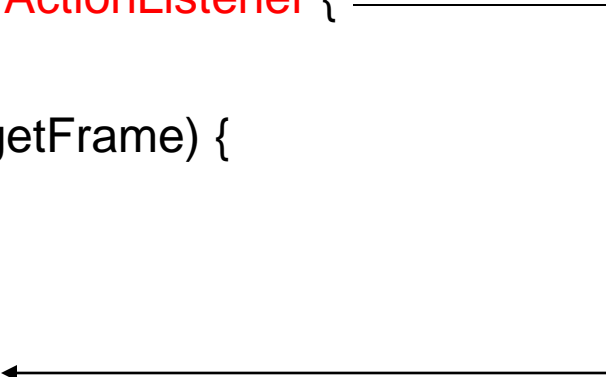
Τύποι ακροατών

ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentShown(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
WindowListener WindowAdapter	windowActivated(WindowEvent) , windowClosed windowClosing, windowDeactivated, windowDeiconified, windowGainedFocus windowIconified, windowLostFocus windowOpened, windowStateChanged

Παράδειγμα - ActionListener

```
public class CopyButtonActionListener implements ActionListener {
    TestFrame targetFrame;
    public CopyButtonActionListener(TestFrame targetFrame) {
        this.targetFrame=targetFrame;
    }
    public void actionPerformed(ActionEvent e) {
        String password = targetFrame.pf.getText();
        targetFrame.ta.append(password);
    }
}
```

ΥΠΟΧΡΕΩΤΙΚΑ



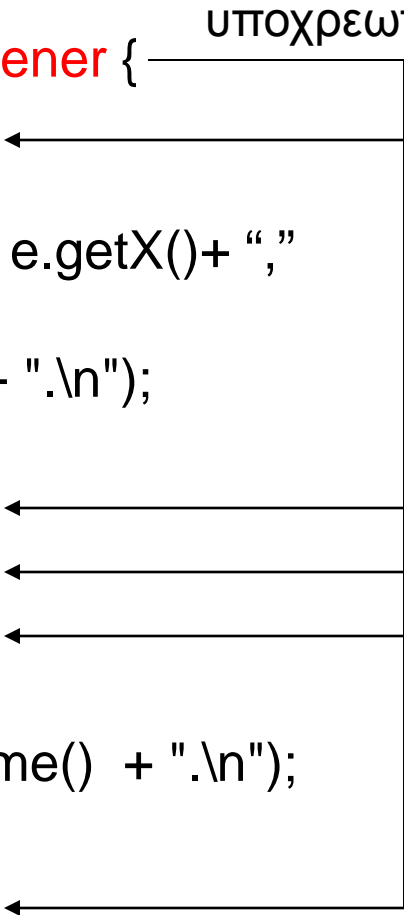
Και στην TestFrame ...

```
copyButton.addActionListener(new CopyButtonActionListener(this));
```

Παράδειγμα - mouseListener

```
public class myMouseListener implements MouseListener {
    public void mousePressed(MouseEvent e) {
        textArea.append ("Mouse pressed: # of clicks: "
            + e.getClickCount(), " detected at position " + e.getX()+ ",",
            + e.getY() + " of component"
            + e.getComponent().getClass().getName() + ".\n");
    }
    public void mouseReleased(MouseEvent e) {...}
    public void mouseClicked(MouseEvent e) { ...}
    public void mouseEntered(MouseEvent e) {
        textArea.append ("Mouse entered detected on "
            + e.getComponent().getClass().getName() + ".\n");
    }
    public void mouseExited(MouseEvent e) { ...}
}
```

ΥΠΟΧΡΕΩΤΙΚΑ



B) Ανίχνευση γεγονότων

- Για να μπορεί ένα στοιχείο να ανιχνεύει γεγονότα θα πρέπει να του προσθέσουμε τον αντίστοιχο ακροατή
`button1.addActionListener(new myActionListener());`
//έτσι θα ανιχνεύουμε πάτημα του κουμπιού
`button1.addMouseListener(new myMouseListener());`
//έτσι θα ανιχνεύουμε κινήσεις του ποντικιού πάνω στο κουμπί
- Τα `FocusEvent`, `KeyEvent`, `MouseEvent`, `ComponentEvent` είναι γεγονότα που προκαλούνται από όλα τα `components` του `swing`
- Το `ContainerEvent` προκαλείται από όλα τα `containers` του `Swing`

Γεγονότα - Στοιχεία που τα παράγουν

ActionEvent	JButton, JCheckBoxMenuItem, JComboBox, JFileChooser, JList, RadioButtonMenuItem, JTextField, JToggleButton
ListSelectionEvent	JList, ListSelectionModel
ItemEvent	JCheckBoxMenuItem, ItemListener, JComboBox, JRadioButtonMenuItem, JToggleButton
MenuEvent	JMenu
MenuKeyEvent	JMenuItem
WindowEvent	JDialog, JFrame, JWindow



Πρακτικές δήλωσης και χρήσης ακροατών

Υλοποίηση με interface

■ Υλοποίηση με interface

- Η τάξη μας είναι απόγονος της JFrame (περιέχει components κλπ)
- Μπορεί να υλοποιεί όλα τα interfaces που χρειάζεται για να ανιχνεύει γεγονότα
- Θα πρέπει να δηλώνει όλες τις μεθόδους, όλων των interfaces, ακόμη και αν υλοποιεί ορισμένες μόνο από αυτές.
- Με τον τρόπο αυτό έχουμε **ένα μόνο ακροατή για όλα** τα στοιχεία που βάζουμε στο πλαίσιο.

■ Υλοποίηση με inner class και adapter

- Η τάξη κληρονομεί μόνο έναν adapter και ορίζει μόνο τις μεθόδους που σχετίζονται με τα γεγονότα που ανιχνεύει
- Θα πρέπει να δηλωθεί ως εσωτερική τάξη ώστε να έχει πρόσβαση στα μέλη της κύριας τάξης που είναι απόγονος της JFrame
- Έχουμε έτσι **ένα ακροατή για κάθε** στοιχείο στο πλαίσιο.

Παράδειγμα χρήσης interface

```
import javax.swing.*;
import java.awt.event.*;
public class MyClass extends JFrame implements MouseListener {
    JButton button1;
    void init(){
        button1=new JButton("OK");
        this.add(button1);
        button1.addMouseListener(this);
        this.setSize(200,200);
    }
    public void mouseClicked(MouseEvent e) { }
    //παρόμοια οι mousePressed και mouseReleased
    public void mouseExited(MouseEvent e) { button1.setText("OFF");}
    public void mouseEntered(MouseEvent e) { button1.setText("ON");}
    public static void main(String args[]){
        MyClass m=new MyClass();
        m.init();
        m.show();
    }
}
```

//είναι ταυτόχρονα πλαίσιο
//και ακροατής

Χρήση ανώνυμης εσωτερικής τάξης

- Αν η λειτουργικότητα αφορά ένα μόνο component είναι προτιμότερο να χρησιμοποιήσουμε ανώνυμη εσωτερική τάξη

```
public class MyClass extends JFrame {  
    JButton button1;  
    void init(){  
        button1=new JButton("OK");  
        this.add(button1);  
        button1.addMouseListener(new MouseAdapter(){  
            public void mouseExited(MouseEvent e) { button1.setText("OFF");}  
            public void mouseEntered(MouseEvent e) { button1.setText("ON");}  
        });  
        this.setSize(200,200);  
    }  
    public static void main(String args[]){ ...}  
}
```

Περισσότερα για τους ακροατές

- ActionListener

- Ορίζει τη μέθοδο

void actionPerformed(ActionEvent)

- Το ActionEvent έχει δύο χρήσιμες μεθόδους

- **String getActionCommand()**

Δίνει ένα μήνυμα που περιγράφει το γεγονός

- **int getModifiers()**

Μας ενημερώνει αν όταν συνέβη το γεγονός (π.χ. Mouse click) είχε πατηθεί κάποιο από τα πλήκτρα SHIFT, CTRL ή ALT

Περισσότερα για τους ακροατές

- **ComponentListener**

- ☐ **void componentHidden(ComponentEvent)**
- ☐ **void componentShown(ComponentEvent)**
- ☐ **void componentMoved(ComponentEvent)**
- ☐ **void componentResized(ComponentEvent)**

Ανάλογα με το αν το στοιχείο κρύφτηκε, εμφανίστηκε, μετακινήθηκε ως προς το πλαίσιο στο οποίο ανήκει, κλπ

- Το **ComponentEvent** έχει τη μέθοδο

- ☐ **Component getComponent()**

Επιστρέφει το στοιχείο που προκάλεσε το γεγονός

Περισσότερα για τους ακροατές

- **ItemListener**: Συνδέεται με μενού, checkboxes

- **void itemStateChanged(ItemEvent)**

- Το **ItemEvent** έχει τις εξής μεθόδους

- **Object getItem()**

- Συνήθως ένα **String** με το κείμενο του στοιχείου

- **ItemSelectable getItemSelectable()**

- Επιστρέφει το στοιχείο που προκάλεσε το γεγονός

- **int getStateChange()**

- Αν το αντικείμενο επιλέχθηκε ή όχι

Περισσότερα για τους ακροατές

- **KeyListener**: Ανιχνεύει πατήματα πλήκτρων
 - **void keyTyped(KeyEvent)** // για Unicode characters
 - **void keyPressed(KeyEvent)** // για πλήκτρα
 - **void keyReleased(KeyEvent)**
- Το **KeyEvent** έχει τις εξής μεθόδους
 - **int getKeyChar()** , **void setKeyChar(int)**
Συνδέει το event με κάποιο Unicode character
 - **int getKeyCode()** , **void setKeyCode(int)**
Επιστρέφει τον κωδικό του πλήκτρου που προκάλεσε το γεγονός

Περισσότερα για τους ακροατές

- **WindowListener**

- ☐ **void windowOpened(WindowEvent)**
- ☐ **void windowClosing(WindowEvent)**
- ☐ **void windowClosed(WindowEvent)**
- ☐ **void windowIconified(WindowEvent),
void windowDeiconified(WindowEvent)**
- ☐ **void windowActivated(WindowEvent)
void windowDeactivated(WindowEvent)**

- Το **WindowEvent** έχει τη μέθοδο

- ☐ **Window getWindow()**

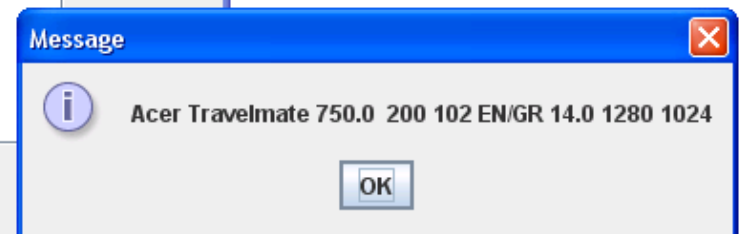
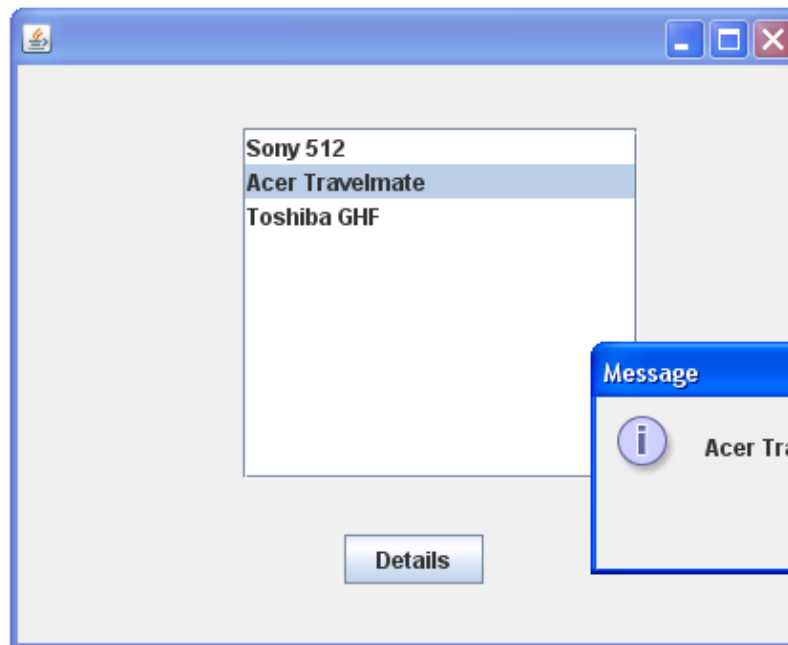
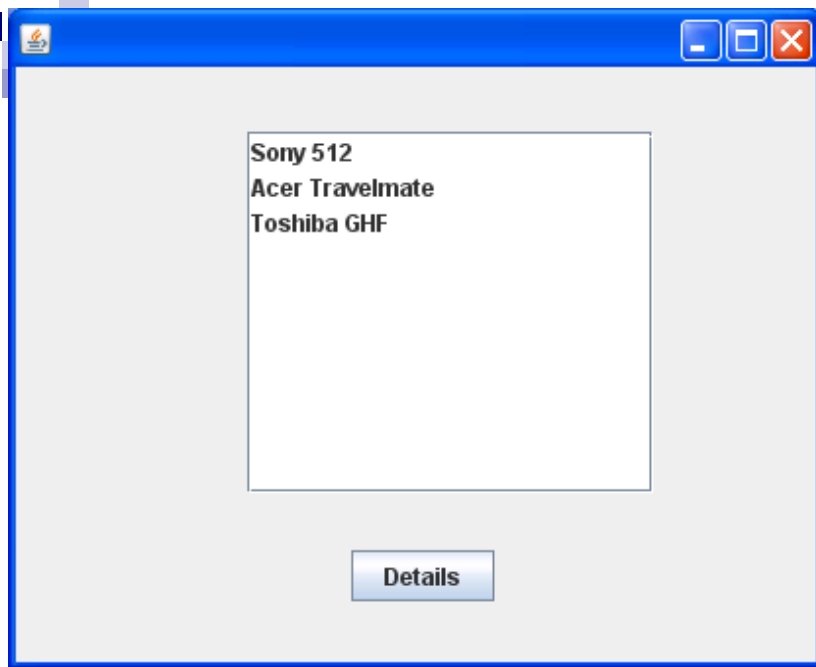
Που επιστρέφει το παράθυρο που προκάλεσε το γεγονός

- Περισσότερα:

<http://java.sun.com/docs/books/tutorial/uiswing/>

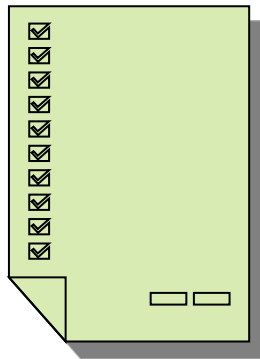
Παράδειγμα

- Στο κατάστημα με τους υπολογιστές θέλουμε να δημιουργήσουμε μια φόρμα που θα κάνει τα ακόλουθα
 - Θα διαβάσει από αρχείο όλους τους υπολογιστές και θα τους εμφανίζει σε μια λίστα, τυπώνοντας μόνο το μοντέλο του υπολογιστή
 - Θα επιτρέπει να επιλέξουμε ένα μόνο υπολογιστή και
 - Αν πατήσουμε το κουμπί Details θα μας δείχνει σε ξεχωριστό παράθυρο τις λεπτομέρειες του υπολογιστή.

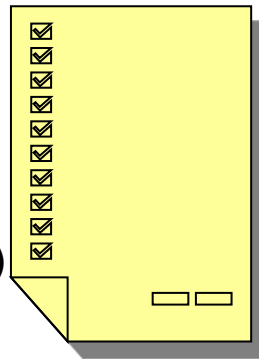


Πόσες φόρμες χρειαζόμαστε

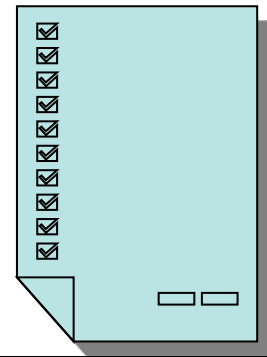
MainFrame



OrderFrame



ProductFrame



createOrder
setVisible(true)

addProduct
setVisible(true)

ArrayList<Order> orderList

ArrayList<Product> productList

Order o = new Order()

Product p = new Product()

Μια φόρμα δεν μπορεί να επιστρέψει δεδομένα

Γι' αυτό πρέπει να της περάσουμε μια αναφορά στην αντίστοιχη δομή

Τι ξέρει
η κάθε
φόρμα;

Τι
δημιουργεί
κάθε
φόρμα;

Λύση

- Στον κατασκευαστή της OrderFrame και της ProductFrame περνάμε την orderList και την ProductList αντίστοιχα
- Στο κουμπί "Καταχώρηση...", δημιουργούμε το αντίστοιχο αντικείμενο και το προσθέτουμε στη λίστα που έχουμε περάσει στη φόρμα.
- Έτσι ενημερώνουμε έμμεσα (μέσω αναφοράς) την αρχική λίστα και μπορούμε πλέον άφοβα να κλείσουμε τη φόρμα
- Η φόρμα κλείνει με τη μέθοδο dispose()