

GRAU D'ENGINYERIA INFORMÀTICA

PROGRAMACIÓ II

Bloc 3:

Programació Orientada a Events (3)

Sergio Sayago (basat en material de Laura Igual)

Departament de Matemàtiques i Informàtica

Facultat de Matemàtiques i Informàtica

Universitat de Barcelona

Exemple 4

- (continuació del bloc 3-2)
- Com gestionem events `ActionEvent` per a dos botons quan cada botó necessita fer una cosa diferent?

Exemple 4: Com gestionem events ActionEvent per a dos botons quan cada botó necessita fer una cosa diferent?



Pensar la viabilitat de les possibles implementacions i implementar-les:

1. La classe aplicació també fa el paper d'escoltador.
2. Separem l'escoltador en dues classes externes.
3. Utilitzem classes internes.

Exemple 4: Com gestionem events ActionEvent per a dos botons quan cada botó necessita fer una cosa diferent?



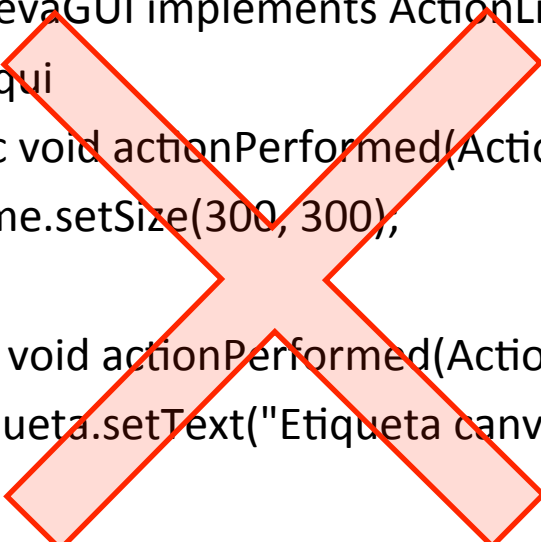
Pensar la viabilitat de les possibles implementacions i implementar-les:

1. **La classe aplicació també fa el paper d'escoltador.**
2. Separem l'escoltador en dues classes externes.
3. Utilitzem classes internes.

Exemple 4: Opció (1a)

- Si la classe fa d'escoltador, llavors haurem d'implementar dos mètodes actionPerformed()

```
class LaMevaGUI implements ActionListener {  
    // codi aquí  
    public void actionPerformed(ActionEvent ev) {  
        frame.setSize(300, 300);  
    }  
    public void actionPerformed(ActionEvent ev) {  
        etiqueta.setText("Etiqueta canviada");  
    }  
} // fi de la classe LaMevaGUI
```



Però això és impossible



Exemple 4: Opció (1b)

```
public class LaMevaGUI implements ActionListener {
    private JFrame frame;
    private JLabel etiqueta;
    private JButton boto1, boto2;
    public LaMevaGUI(){
        frame = new JFrame();
        etiqueta = new JLabel("Soc una etiqueta");
        boto1 = new JButton("Canvia Tamany");
        boto2 = new JButton("Canvia Etiqueta");
    }
    public void go() {
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().add(BorderLayout.NORTH, boto1);
        frame.getContentPane().add(BorderLayout.SOUTH, boto2);
        frame.getContentPane().add(BorderLayout.CENTER, etiqueta);
        boto1.addActionListener(this);
        boto2.addActionListener(this);

        frame.setSize(100, 100);
        frame.setVisible(true);
    }
    public void actionPerformed(ActionEvent ev) {
        if(ev.getSource().equals(boto1)) {
            frame.setSize(300, 300);
        }else{
            etiqueta.setText("Etiqueta canviada");
        }
    }
}
public static void main(String[] args) {
    LaMevaGUI gui = new LaMevaGUI();
    gui.go(); }
} // fi de la classe LaMevaGUI }
```

- O també podem registrar el mateix listener amb dos botons

Registre'm el mateix listener

Mirem quin event és

Opció correcta, però pot arribar a ser complicada de gestionar. Us imagineu que passaria si tinguéssim 5 components, o 10...?

Exemple 4: Com gestionem events ActionEvent per a dos botons quan cada botó necessita fer una cosa diferent?



Pensar la viabilitat de les possibles implementacions i implementar-les:

1. La classe aplicació també fa el paper d'escoltador.
2. **Separem l'escoltador en dues classes externes.**
3. Utilitzem classes internes.

Exemple 4: Opció (2a)

- Crear dues classes ActionListener separades

```
class LaMevaGUI {  
    private JFrame frame;  
    private JLabel etiqueta;  
    public static void main (String[] args){  
        LaMevaGUI gui = new LaMevaGUI2();  
        gui.go();  
    }  
    public void go(){  
        frame = new JFrame();  
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        JButton boto1 = new JButton("Canvia Tamany");  
        JButton boto2 = new JButton("Canvia Etiqueta");  
        etiqueta = new JLabel("Soc una etiqueta");  
        frame.getContentPane().add(BorderLayout.NORTH, boto1);  
        frame.getContentPane().add(BorderLayout.SOUTH, boto2);  
        frame.getContentPane().add(BorderLayout.CENTER, etiqueta);  
        boto1.addActionListener(new Boto1Listener());  
        boto2.addActionListener(new Boto2Listener());  
        frame.setSize(100, 100);  
        frame.setVisible(true);  
    }  
} // fi de la classe
```

```
public class Boto1Listener implements  
    ActionListener {  
    public void actionPerformed(ActionEvent ev){  
        frame.setSize(300, 300);  
    }  
} // fi de la classe
```

```
public class Boto2Listener implements  
    ActionListener {  
    public void actionPerformed(ActionEvent ev){  
        etiqueta.setText("Etiqueta canviada");  
    }  
} // fi de la classe
```

Problema! Aquesta classe no té
referència a la variable etiqueta i
frame

Exemple 4: Opció (2b)

- Crear dues classes ActionListener separades

Arreglant el problema d'abans

```
public class LaMevaGUI2 {
    private JFrame frame;
    private JLabel etiqueta;
    public static void main (String[] args){
        LaMevaGUI2 gui = new LaMevaGUI2();
        gui.go();
    }
    public void go(){
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton boto1 = new JButton("Canvia Tamany");
        JButton boto2 = new JButton("Canvia Etiqueta");
        boto1.addActionListener(new Boto1Listener(this));
        boto2.addActionListener(new Boto2Listener(this));
        etiqueta = new JLabel("Soc una etiqueta");
        frame.getContentPane().add(BorderLayout.NORTH, boto1);
        frame.getContentPane().add(BorderLayout.SOUTH, boto2);
        frame.getContentPane().add(BorderLayout.CENTER, etiqueta);
        frame.setSize(100, 100);
        frame.setVisible(true);
    }

    public JLabel getEtiqueta(){
        return etiqueta;
    }
    public void setEtiqueta(JLabel etiqueta){
        this.etiqueta = etiqueta;
    }
    public JFrame getFrame(){
        return frame;
    }
    public void setFrame(JFrame frame){
        this.frame = frame;
    }
} // fi de la classe
```

Exemple 4: Opció (2b)

- Crear dues classes ActionListener separades

```
public class Boto1Listener implements ActionListener {  
    private LaMevaGUI2 gui;  
  
    public Boto1Listener(LaMevaGUI2 gui){  
        this.gui = gui;  
    }  
  
    public void actionPerformed(ActionEvent ev) {  
        gui.getFrame().setSize(300, 300);  
    }  
} // fi de la classe
```

Atenció! Aquestes classes han de rebre l'objecte LaMevaGUI2 per poder accedir als atributs de la mateixa.

Potser fa **difícil** entendre el codi.

```
public class Boto2Listener implements ActionListener {  
    private LaMevaGUI2 gui;  
  
    public Boto2Listener(LaMevaGUI2 gui){  
        this.gui = gui;  
    }  
  
    public void actionPerformed(ActionEvent ev) {  
        gui.getEtiqueta().setText("Etiqueta canviada");  
    }  
} // fi de la classe
```

Exemple 4: Com gestionem events ActionEvent per a dos botons quan cada botó necessita fer una cosa diferent?



Pensar la viabilitat de les possibles implementacions i implementar-les:

1. La classe aplicació també fa el paper d'escoltador.
2. Separem l'escoltador en dues classes externes.
3. **Utilitzem classes internes.**

```

import java.awt.BorderLayout;
import javax.swing.*;
import java.awt.event.*;
public class LaMevaGUI {
    JFrame frame;
    JLabel etiqueta;
    public static void main(String[] args){
        LaMevaGUI gui = new LaMevaGUI();
        gui.go();
    }
    public void go() {
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JButton botoTamany = new JButton("Canvia Tamany");
        botoTamany.addActionListener(new
BotoTamanyListener());
        JButton botoEtiqueta = new JButton("Canvia Etiqueta");
        botoEtiqueta.addActionListener(new
BotoEtiquetaListener());
        etiqueta = new JLabel("Soc una etiqueta");
        frame.getContentPane().add(BorderLayout.NORTH,
botoTamany);
        frame.getContentPane().add(BorderLayout.SOUTH,
botoEtiqueta);
        frame.getContentPane().add(BorderLayout.CENTER,
etiqueta);
        frame.setSize(100, 100);
        frame.setVisible(true);
    } // Fi mètode go

```

Classes internes

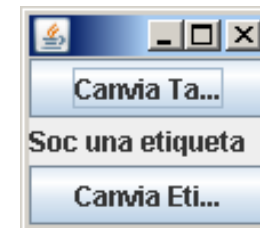
```

class BotoTamanyListener implements ActionListener
{
    public void actionPerformed(ActionEvent ev) {
        frame.setSize(300, 300);
    }
}

class BotoEtiquetaListener implements
ActionListener {
    public void actionPerformed(ActionEvent ev) {
        etiqueta.setText("Etiqueta canviada");
    }
}

} // Fi de la classe LaMevaGUI

```



Exemple 4: Opció (3)

Exemple 4: Opció (3b)

- Crear dues classes ActionListener **internes anònimes**.
- Aquesta és l'opció més comuna quan tenim moltes components en la GUI.

```

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

public class LaMevaGUI4 {
    JFrame frame;
    JLabel etiqueta;
    public static void main(String[] args){
        LaMevaGUI4 gui = new LaMevaGUI4();
        gui.go();
    }
    public void go() {
        frame = new JFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JButton botoTamany = new JButton("Canvia Tamany");
        botoTamany.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ev) {
                frame.setSize(300, 300);
            }
        });
        JButton botoEtiqueta = new JButton("Canvia Etiqueta");
        botoEtiqueta.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent ev) {
                etiqueta.setText("Etiqueta canviada");
            }
        });
    }
}

```

```
etiqueta = new JLabel("Soc una etiqueta");
```

```
frame.getContentPane().add(BorderLayout.NORTH,
    botoTamany);
```

```
frame.getContentPane().add(BorderLayout.SOUTH,
    botoEtiqueta);
```

```
frame.getContentPane().add(BorderLayout.CENT
ER, etiqueta);
```

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

```
frame.setVisible(true);
```

```
// Fi mètode go
```

```
// Fi de la classe LaMevaGUI
```

Classes internes anònimes

Fa **més fàcil** entendre el codi.



Exemple 4: Opció (3b)

Comentaris sobre les classes internes

- S'ha d'implementar tots els mètodes de la interfície
- Si el codi utilitzat per a implementar la manipulació d'events té unes poques línies es sol utilitzar una **classe interna anònima**.
- Una **classe interna amb nom** serà útil quan la volem instanciar més d'una vegada.

Índex Bloc 3:

Programació Orientada a Events

- Mecanismes d'interacció
 - Interacció mitjançant flux seqüencial
 - Interacció mitjançant programació orientada a events
- Programació d'Interfícies Gràfiques d'Usuari
- Model de gestió d'events: Exemple d'implementació d'una finestra.
- Events i Listeners
- Components i Contenedors
- Classes adapter i classes internes: Exemple d'implementació d'una finestra que es tanca.
- **Mes sobre swing components: Exemples**
- Layout manager
- Look and feel
- Panells i gràfics
- Animacions

COMPONENTS (UNA BREU INTRODUCCIÓ)

Components

Les components estan classificades com

1. Contenidor d'alt nivell

Cada contenidor d'alt nivell té un JRootPane que és l'arrel de la jerarquia de contenidors.

2. Contenidor intermedi

3. Contenidor específic

4. Control bàsic

5. Displays no editables

6. Displays interactius

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

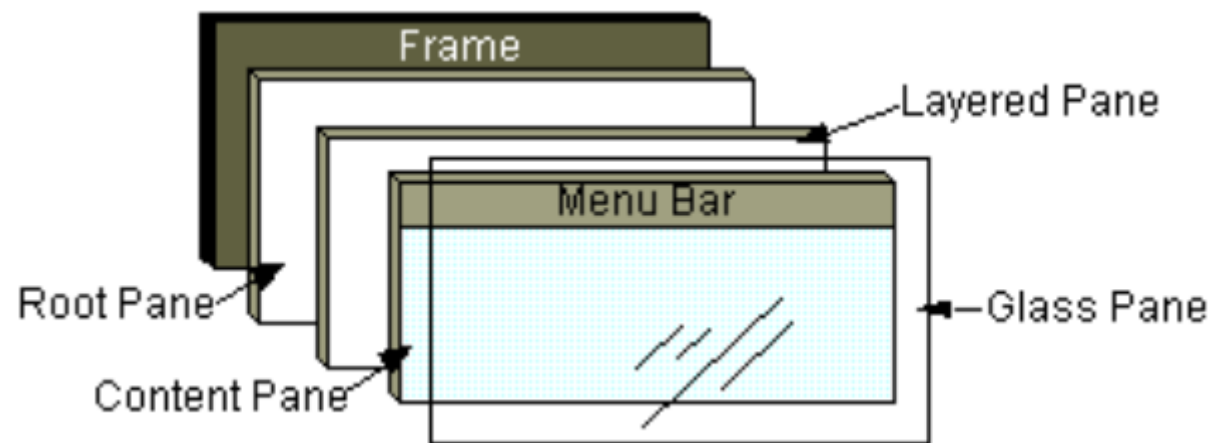
Contenidors d'alt nivell

- JFrame - Frames o Marcos (finestres)
 - Finestra principal
 - Una aplicació Java només pot tenir una única finestra principal
- JDialog – Diàlegs
 - Una finestra independent que ens servirà per a proporcionar informació temporal a la finestra principal de l'aplicació. La majoria serveixen per mostrar un missatge d'error o warnings als usuaris, però poden mostrar imatges, arbres de directoris, etc.
- JApplet – Applets
 - Una component d'una aplicació que s'executa en el context d'un altre programa, per exemple un navegador web.

(<http://docs.oracle.com/javase/tutorial/deployment/TOC.html>)

JRootPane

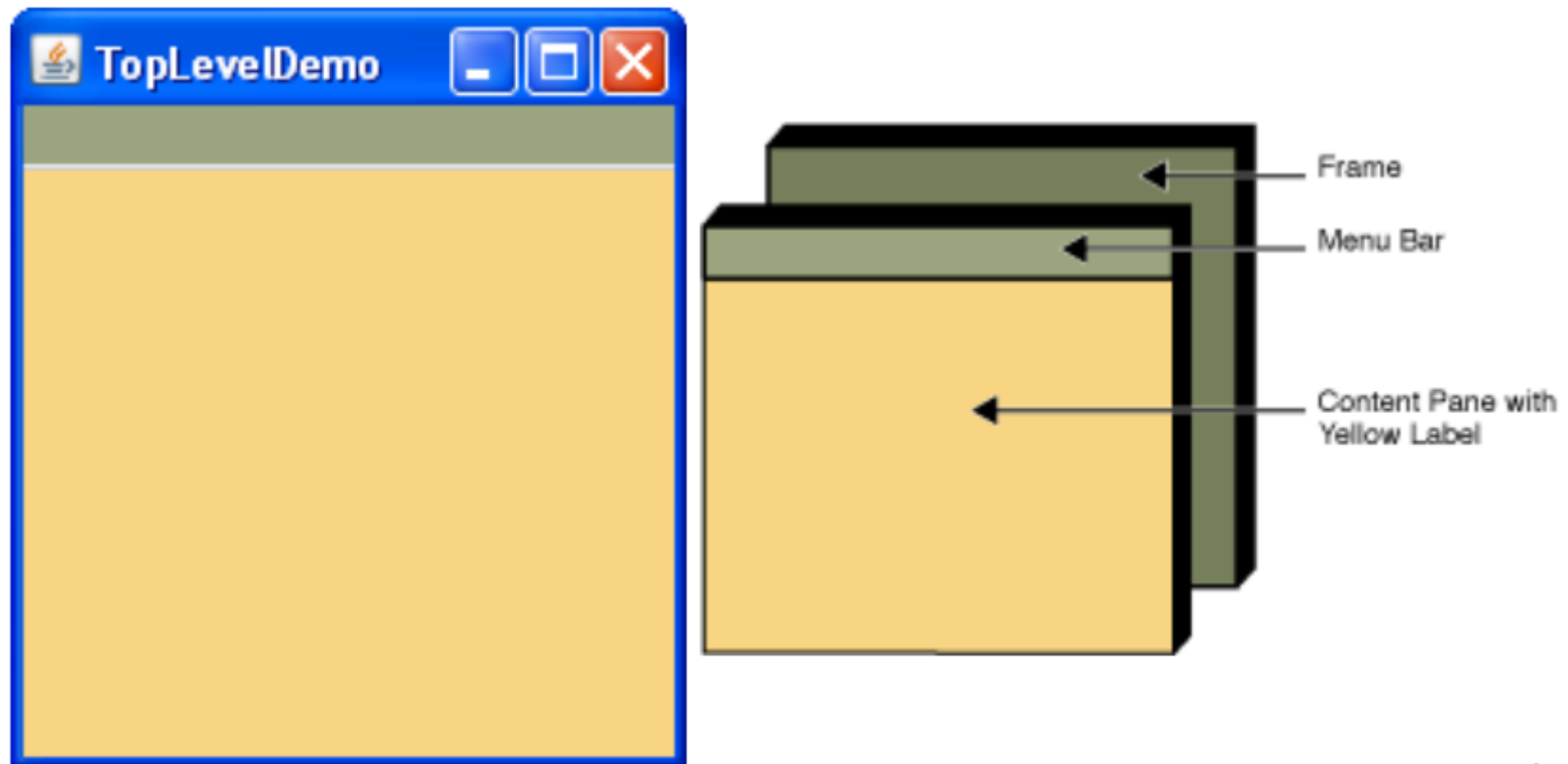
- El JRootPane es crea quan s'instancia un dels contenidors d'alt nivell de Swing.



<http://docs.oracle.com/javase/tutorial/uiswing/components/rootpane.html>

JRootPane

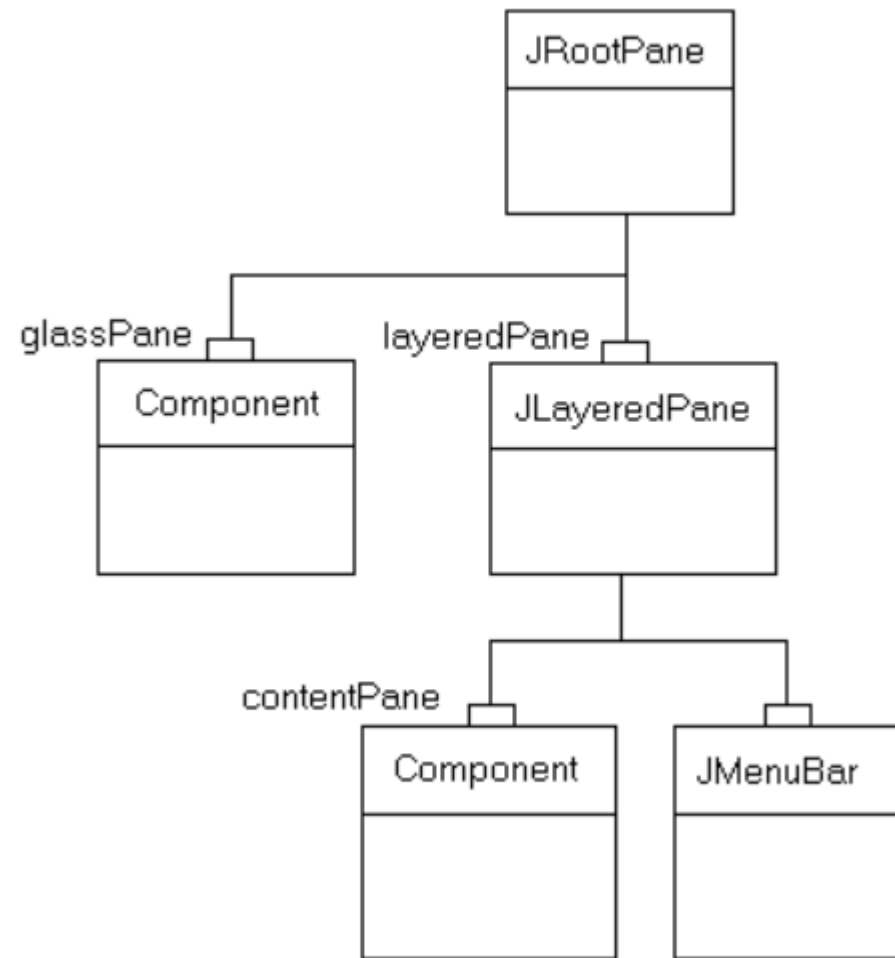
- El JRootPane es crea quan s'instancia un dels contenidors d'alt nivell de Swing.



Jerarquia de JRootPane

- Tots els contenidors d'alt nivell deleguen les seves operacions a un JRootPane
- Per afegir components al JRootPane, s'afegeix l'objecte al contentPane del JRootPane de la següent manera:

`rootPane.getContentPane().add(child)`



Contenidors intermedis

- Normalment, s'utilitzen per a agrupar components, perquè les components estan relacionades o només perquè agrupar-les fa que la distribució sigui més senzilla.
- Un panell pot fer servir qualsevol controlador de distribució (layout manager), ho veurem després.
 - Panell (panel)
 - Panell lliscants (scroll pane)
 - Panell dividit (split pane)
 - Panell amb solapes (tabbed pane)
 - Barra d'eines (tool bar)

Control bàsic

- Botons (Buttons),
 - Caixes combo (Combo boxes),
 - Barra lliscant (Sliders),...
-
- Elements d'interfície que els usuaris poden manipular per prémer un botó, seleccionar una opció o fixar un valor.

Displays no editables

- Etiquetes (labels)
- Barres de progrés (progress bars)
- Pistes d'eines (tool tips)

Per afegir una pista d'eina a un botó:

```
b1.setToolTipText("Clica aquest botó per desactivar el botó del mig.");
```

Displays interactius

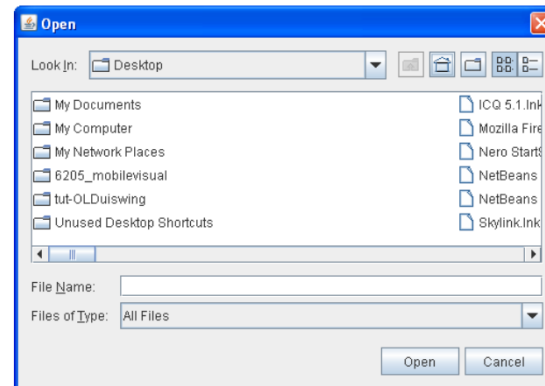
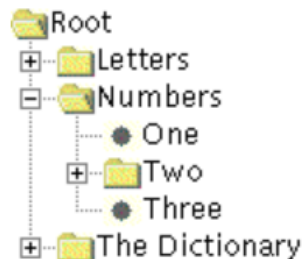
- Selector de colors (JColorChooser)
- Taula (JTable)
- Text (JTextComponent)
- Selector de fitxers (JFileChooser)
- Arbres (JTree)

The Header contains Column labels

First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	<input type="checkbox"/>
John	Doe	Rowing	3	<input type="checkbox"/>
Sue	Black	Knitting	2	<input type="checkbox"/>
Jane	White	Speed reading	20	<input checked="" type="checkbox"/>

Each Cell displays a data item

Each Column displays one type of data

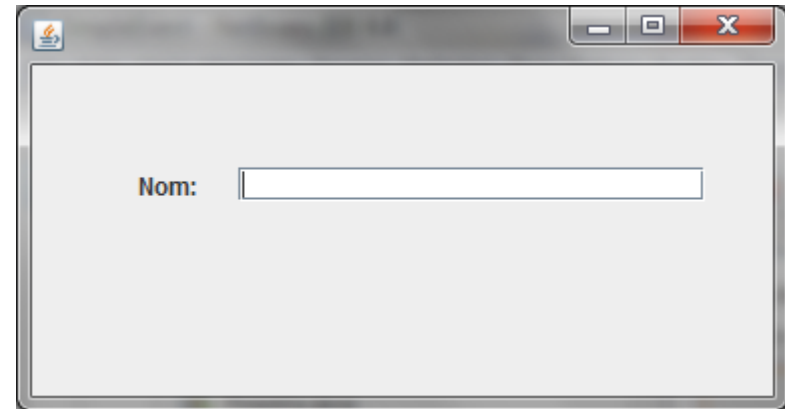


TextField

```
TextField camp = new TextField(20);  
TextField camp = new TextField("El meu missatge");
```

← 20 columnes

- Mètodes importants:
- `String getText()` i `setText(String str)`
Permeten establir o obtenir el text del component
- `addActionListener(ActionListener al)`,
- `removeActionListener(ActionListener al)`
Permet registrar o borrar l'objecte que gestionarà l'event
- `selectAll()`, `select(int start, int end)`
Selecciona tot o part del text
- `requestFocus()` (de la classe `Component`)
Permet fer des del programa que un component obtinga el Focus.



JTextArea

- Pot contenir més d'una línia de text
- Per afegir barres lliscants s'ha d'afegir un ScrollPane

- Constructor:

```
JTextArea text = new JTextArea(10, 20);
```

10 files 20 columnes

```
JScrollPane scroller= new JScrollPane(text);
```

```
text.setLineWrap(true);
```

Estableix la política d'ajustament de línies de l'àrea de text.

```
scroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
```

```
O ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER,  
  ScrollPaneConstants.VERTICAL_SCROLLBAR_AS_NEEDED
```

```
panel.add(scroller);
```

```
text.setText("Canviem el text");
```

```
text.append("botó apretat");
```

```
text.selectAll();
```

```
text.requestFocus();
```

L'ample d'una columna és igual a l'ample d'un caràcter, en la font particular que s'està utilitzant.

JTextArea

- Afegir-li una scrollbar vertical:

```
JScrollPane scroller = new JScrollPane (list);
```

```
scroller.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);
```

```
scroller.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER)
```

```
;
```

```
panel.add(scroller);
```

Exemple 1

- Implementeu una interfície gràfica d'usuari que contingui un botó i una àrea de text. Cada vegada que es prem el botó s'ha d'escriure el text “botó apretat” a l'àrea de text.
- Per fer!

Exemple 2

- Implementeu una interfície gràfica d'usuari que contingui un botó i una àrea de text. Quan es prem el botó s'ha d'escriure el text contingut en ell a l'àrea de text.
- Per fer!