

# Construction d'interfaces graphiques

Licence Informatique Semestre 3

Martine Gautier Université H. Poincaré Nancy

`Martine.Gautier@loria.fr`



# Chapitre 3

## Gestionnaires et composants

Martine Gautier Université H. Poincaré Nancy  
Martine.Gautier@loria.fr

# 1. Composants, conteneurs et gestionnaires

- **Composant** (*component*) élément susceptible d'apparaître sur une interface utilisateur
  - ↪ bouton, liste à défilement, menu, case à cocher, champ de texte, fenêtre, curseur, *etc.*
  - ↪ cas particulier de `Component` et `JComponent`
- **Conteneur** (*container*) élément susceptible de contenir plusieurs composants / conteneurs
- **Gestionnaire de présentation** (*layout*) dispose et dimensionne les composants d'un conteneur.
  - ↪ appelé aussi *gestionnaire de mise en page*, *gestionnaire de géométrie*
- Gestion arborescente des composants
  - La mise en place de l'arborescence des composants est dynamique.
  - L'apparence de l'interface peut évoluer au cours de l'exécution : ajouter/enlever des composants

## 2. Gestionnaires de présentation

- Positionnement des différents composants au niveau du pixel de l'écran
  - ↪ calculs délicats et fastidieux
  - ↪ recalculs indispensables en cas de redimensionnement de la fenêtre.
- Utilisation de gestionnaires de mise en page
  - ↪ calculs et recalculs automatiques
  - ↪ plusieurs sortes de mises en pages
- Tous les conteneurs ont un gestionnaire par défaut.
  - ↪ `setLayout(LayoutManager l)` pour en changer

## Fonctions communes à tous les gestionnaires

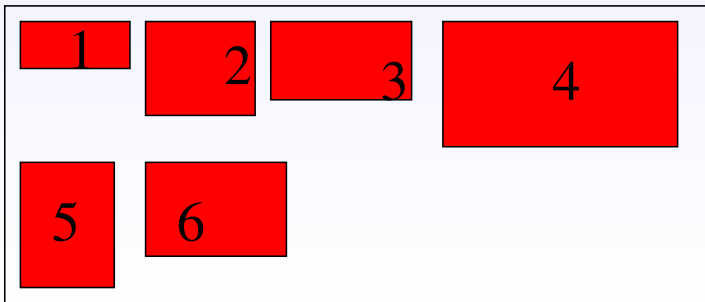
- Un gestionnaire place au mieux les composants en utilisant
  - l'algorithme de placement qui lui est propre,
  - les indications de positionnement des composants,
  - la taille du conteneur,
  - les tailles préférées des composants.
- Dimensionnement des composants en pixels :  
`java.awt.Dimension`
  - ↪ préférée : `setPreferredSize()`
  - ↪ maximale : `setMaximalSize()`
  - ↪ minimale : `setMinimalSize()`
- En règle générale, les gestionnaires ne tiennent pas trop compte des tailles imposées.
- Recalcul et actualisation de l'affichage
  - ↪ agrandissement/réduction provoqué(e) par l'utilisateur
  - ↪ appel à la fonction `layoutContainer()`
  - ↪ appel à la fonction `revalidate()`

## Différents gestionnaires

- Gestionnaires sans contrainte
  - ↪ `FlowLayout`
  - ↪ `GridLayout`
  - ↪ `BoxLayout`
  - ↪ `CardLayout` (plus utilisé)
- Gestionnaires avec contraintes
  - ↪ `BorderLayout`
  - ↪ `GridBagLayout`
  - ↪ `GroupLayout` (une nouveauté du JDK 6)
  - ↪ `SpringLayout` (utilisé essentiellement par des builders d'interface)
- Possibilité de se construire son propre gestionnaire (un peu compliqué ...)

## Gestionnaire FlowLayout

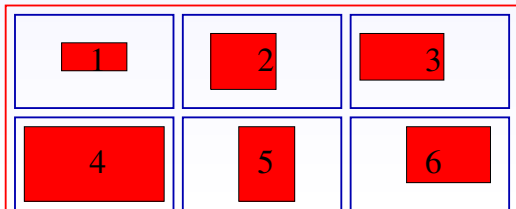
- C'est le gestionnaire par défaut de JPanel et JApplet
- La gestion de l'espace est rudimentaire : chaque composant est ajouté sur la même ligne que la précédente, sauf s'il ne reste plus assez de place.
- Les composants sont affichés avec leur taille préférée.
- L'alignement ainsi que l'espacement horizontal/vertical entre les composants peuvent être fixés lors de l'instanciation du gestionnaire et peuvent évoluer dynamiquement.





## Gestionnaire GridLayout

- L'espace est géré en grille, les nombres de lignes et de colonnes étant fixés lors de l'instanciation du gestionnaire.
  - ↪ si le nombre de lignes est fixé ( $>0$ ), le nombre de colonnes est ignoré
  - ↪ si le nombre de lignes est nul, il est ignoré
- Une cellule de la grille ne contient qu'un seul composant.
- Toutes les cellules de la grille ont la même taille, celle du plus grand des composants rangés dans la grille.
- La grille est remplie ligne par ligne.
- L'espacement entre les lignes et les colonnes peut être fixé à la création et peut varier dynamiquement.



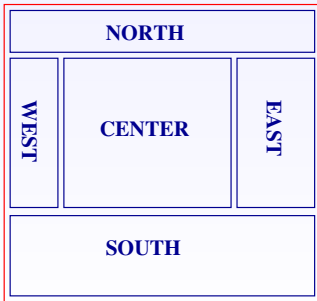
## Gestionnaire BoxLayout

- L'espace est géré, au choix, verticalement ou horizontalement.
- Chaque composant est ajouté après le précédent selon l'axe choisi.
- On peut ajouter de la *glue* (composant élastique vide) pour forcer le placement dans la ligne/colonne.  
~> `Box.createHorizontalGlue()`



## Gestionnaire BorderLayout

- L'espace est découpé en 5 zones, appelées respectivement NORTH, SOUTH, EAST, WEST et CENTER (champs statiques et constants).
- Une zone ne contient qu'un seul composant.
- Les zones peuvent être de tailles différentes.
- La zone CENTER prend toute la place non utilisée par les 4 autres zones.



### 3. Composants racines de l'arborescence

- `JFrame` : fenêtre avec titre, boutons de fermeture et de redimensionnement
  - possibilité d'adjonction de menus
  - fenêtre principale d'une application (dans la plupart des cas)
- `JApplet` : pour développer une application téléchargée par un navigateur Web, s'exécutant dans une fenêtre de ce navigateur.
- `JWindow` : fenêtre rudimentaire, sans titre, ni bouton de fermeture et de redimensionnement
- `JDialog` : pour réaliser des boîtes de dialogue.

## 4. Caractéristiques communes des composants

- Sous-classes de `swing.JComponent`, sous-classe de `awt.Component`
- Dimensionnement et placement (pas forcément utilisé par les gestionnaires)
- Bordure (*cf.* méthodes statiques de `BorderFactory`)
- Transparence : par défaut, un composant est opaque, mais il peut être transparent (`setOpaque`)
- Hors-service : un composant peut être désactivé, c'est-à-dire ne plus répondre aux actions de l'utilisateur (`setEnabled`)
- Facilités de debugage : trace, clignotement de l'affichage  
↪ `setDebugGraphicsOptions`

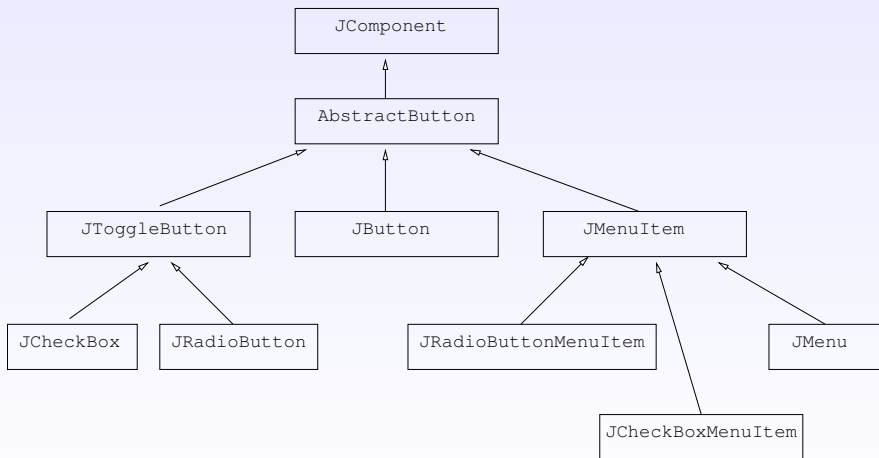
## Composants de base

- JLabel : étiquette contenant un texte et/ou une icône
- JButton : bouton
- JTextField : une ligne de texte
- JPasswordField : zone d'affichage et de saisie d'un mot de passe
- JTextArea : zone d'affichage et de saisie de texte

## Composants évolués

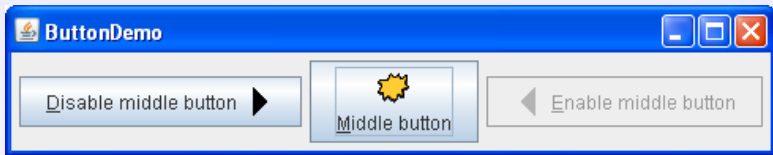
- JScrollPane : panneau avec barres de défilement si nécessaire
- JCheckBox : cases à cocher
- JRadioButton : groupes de cases à cocher
- JComboBox : liste de choix déroulante
- JOptionPane : boîtes de dialogues standards
- JTabbedPane : onglets
- JMenu : menu déroulant
- JFileChooser : pour sélectionner un fichier dans un répertoire
- JEditorPane : éditeur de textes
- JTextPane : éditeur de textes, avec paragraphes, styles

## 5. Les boutons





## Exemple de JButton



- Un bouton peut contenir du texte et/ou une image
- Raccourci clavier pour éviter le clic
- Activation/désactivation possible

```
JButton b1 = new JButton("<html>Je suis un robot  
<p><FONT COLOR=RED>Je me présente </html>");
```

```
ImageIcon im = new  
ImageIcon(getClass().getResource("r2d2.jpeg"));  
JButton b2 = new JButton(im);
```

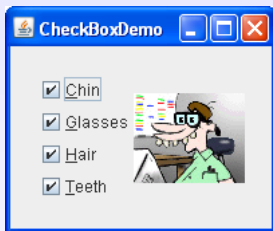
```
JButton b3 = new JButton("Mon nom est r2d2");  
b3.setMnemonic('r');  
b3.setEnabled(false);  
b3.setActionCommand("Robot");  
b3.addActionListener(ecouteur);
```

```
....
```

```
public void actionPerformed(ActionEvent a) {  
    if (a.getActionCommand().equals("Robot") ...
```

```
..  
}
```

## JCheckBox



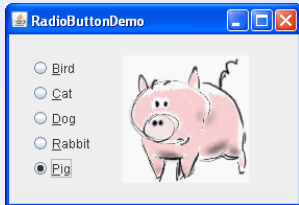
Gestion spécifique des événements (ce qui n'exclut pas la possibilité de définir un `ItemListener`)

```
// Créer une boîte à cocher
```

```
JCheckBox chinButton = new JCheckBox(" Chin");  
chinButton.setMnemonic(KeyEvent.VK_C);  
chinButton.setSelected(true);
```

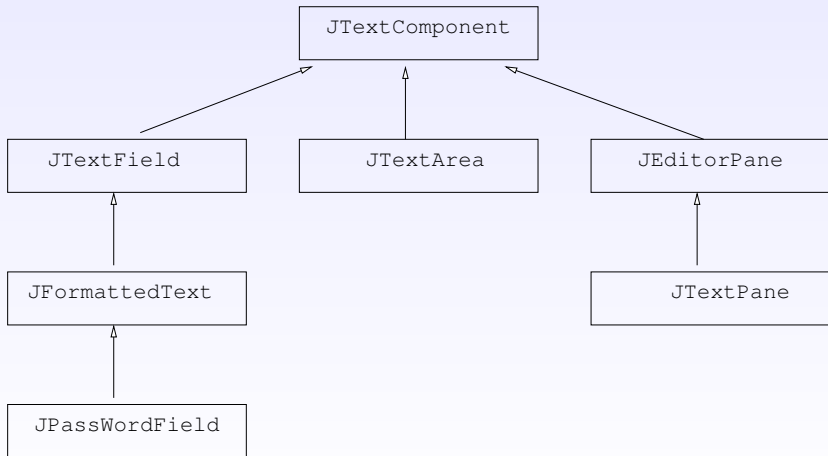
```
// Consulter l'état de la boîte à cocher quand on en a besoin  
boolean on = chinButton.isSelected();
```

## JRadioButton



```
String birdString = "Bird" ;  
JRadioButton birdButton = new JRadioButton(birdString) ;  
birdButton.setActionCommand(birdString) ;  
birdButton.addActionListener(ecouteur) ;  
// Ajouter le bouton dans un groupe de boutons  
ButtonGroup group = new ButtonGroup() ;  
group.add(birdButton) ;  
// Distinguer la source de l'événement dans l'écouteur  
public void actionPerformed(ActionEvent a) {  
    if (a.getActionCommand().equals("Robot") ...  
}
```

## 6. Les textes



**TextSamplerDemo** [minimize] [maximize] [close]

**Text Fields**

JTextField: Hello

JPasswordField: ●●●●


JFormattedTextField: Feb 20, 2007

You typed "hola"

**Plain Text**


*This is an editable JTextArea. A text area is a "plain" text component, which means that although it can display text in any font, all of the text is in the same font.*


**Styled Text**

 This is an uneditable JEditorPane, which was *initialized* with **HTML** text from a **URL**.

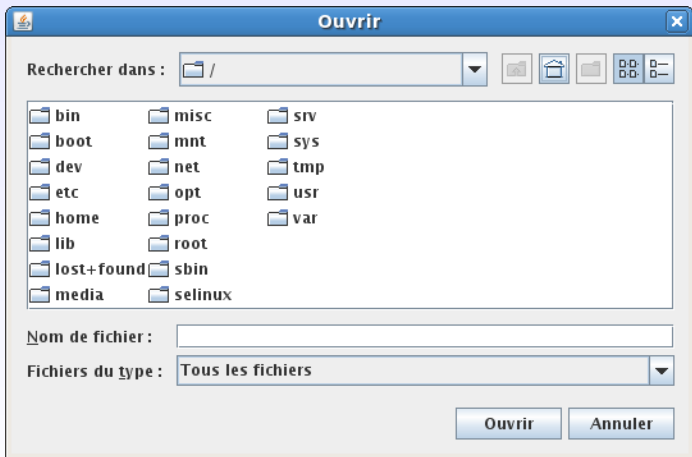
An editor pane uses specialized editor

This is an editable JTextPane, another **styled** text component, which supports embedded components...

 ...and embedded icons...



## 7.Composant JFileChooser ... l'exemple



## Composant JFileChooser ... le programme

```
try {  
    JFileChooser jf = new JFileChooser();  
    int reponse = jf.showOpenDialog(parent);  
    // parent : composant dont dépend jf  
    if (reponse == JFileChooser.APPROVE_OPTION) {  
        File fichier = jf.getSelectedFile();  
  
        ...  
    } // if  
} catch (Exception ex) ex.printStackTrace();
```



## Compléments sur JFileChooser

- Possible de fixer le répertoire sur lequel on ouvre le `FileChooser`, lors de la création.
- Créer un seul `FileChooser` dans une application  
↪ chaque appel de `showOpenDialog` ouvre le répertoire utilisé la fois précédente
- Possible de choisir des fichiers et/ou des répertoires (`setFileSelectionMode`)
- Possible de changer les différents intitulés
- Sélection simple ou multiple (`setMultiSelectionEnabled`)

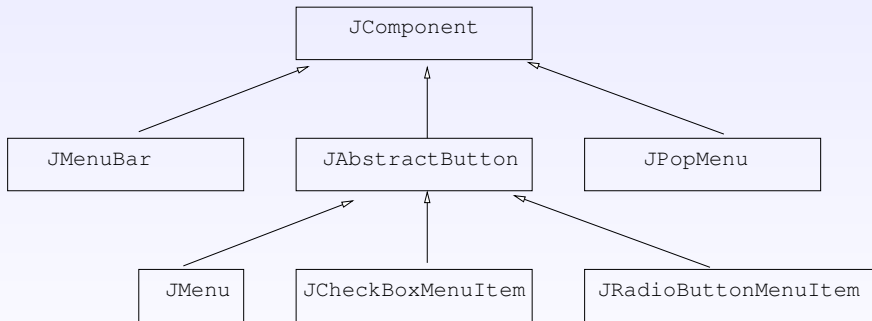
## 8.Composant JTabbedPane ... un exemple



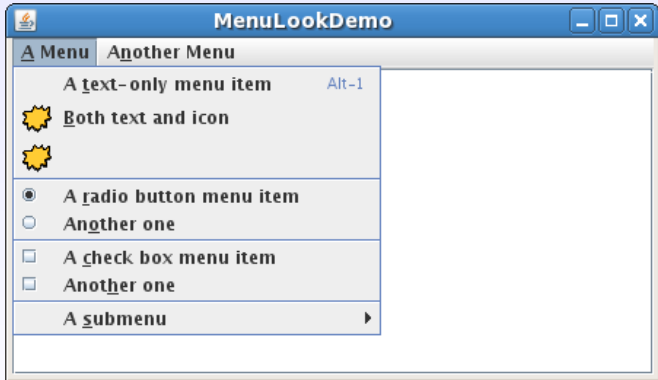
## Composant JTabbedPane ... quelques fonctions

```
int position = JTabbedPane.BOTTOM;  
JTabbedPane onglets = new JTabbedPane(position); // pour  
fixer la place des onglets  
String tip = "Cliquez pour sélectionner";  
ImageIcon icone = new ImageIcon("icomes/lune.gif");  
onglets.addTab("Nom de l'onglet", icone, panneau, tip);  
  
int index = onglets.getSelectedIndex();  
onglets.remove(index);  
  
onglets.setIconAt(3, nouvelleImage);
```

## 9. Les menus



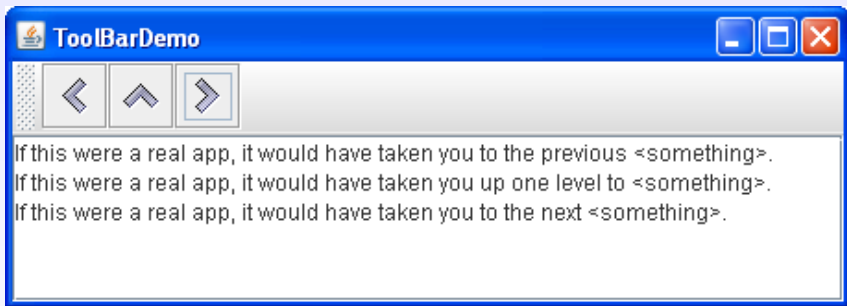
# Exemple de menu



## Composant JMenu ... quelques fonctions

```
JFrame fenetre= new JFrame (" Exemple de JFrame" );  
JMenuBar barreMenus = new JMenuBar();  
// Un menu  
JMenu menu = new JMenu(" Un menu" );  
menu.setMnemonic(KeyEvent.VK_M);  
// Un item de menu  
JMenuItem jmi = new JMenuItem(" Un item", image);  
jmi.setMnemonic(KeyEvent.VK_B);  
jmi.setActionCommand(" Un item" );  
menu.add(jmi);  
barreMenus.add(menu);  
  
jmi.addActionListener(ecouteur);  
fenetre.setJMenuBar(barreMenus);
```

## 10. Les barres d'outil



```
JToolBar tb = new JToolBar();  
JButton b1 = new JButton(image);  
b1.addActionListener(ecouteur);  
tb.add(b1);
```

## 11. Le dialogue avec l'utilisateur

- JOptionPane : boîtes de dialogue *modales*  
↪ l'utilisation de tout composant de l'interface graphique est impossible tant que l'utilisateur n'a pas cliqué sur l'un des boutons de la boîte
- Utilisées pour afficher des messages d'erreurs, des avertissements ou pour effectuer des saisies.
- La boîte de dialogue la plus complète a la forme suivante :

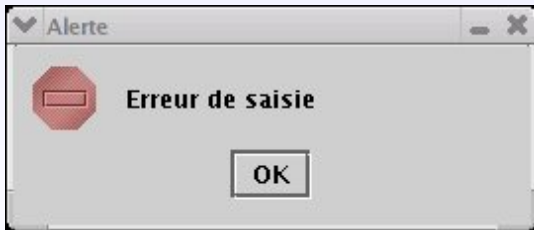


**forme de boîte affichée par showDialog(..)**



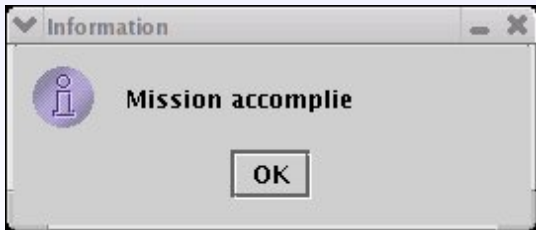
## Composant JOptionPane ... exemple

```
// Message d'erreur  
JOptionPane.showMessageDialog(null,  
    "Erreur de saisie", "Alerte",  
    JOptionPane.ERROR_MESSAGE);
```



## Composant JOptionPane ... exemple

```
// Information  
JOptionPane.showMessageDialog(null,"Mission accomplie",  
    "Information",  
    JOptionPane.INFORMATION_MESSAGE);
```



## Composant JOptionPane ... exemple

```
// Information et demande de réponse, avec 2 choix possibles
int rep1 = JOptionPane.showConfirmDialog(null,
    "sauvegarder avant de quitter?",
    "Il faut choisir ...",
    JOptionPane.YES_NO_OPTION);
if (rep1==JOptionPane.OK_OPTION) ...
```



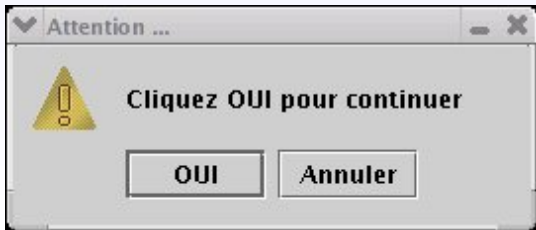
## Composant JOptionPane ... exemple

```
// Information et demande de réponse, avec 3 choix possibles
int rep2 = JOptionPane.showConfirmDialog(null,
    "sauvegarder avant de quitter?",
    "Encore choisir ... ",
    JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.INFORMATION_MESSAGE);
if (rep2==JOptionPane.CANCEL_OPTION) ...
else if (rep2==JOptionPane.NO_OPTION) ... else
```



## Composant JOptionPane ... exemple

```
// Avertissement et demande de réponse (2 choix)
Object[] options = {" OUI", " Annuler" };
int rep3 = JOptionPane.showOptionDialog(null,
    " Cliquez OUI pour continuer", " Attention ... ",
    JOptionPane.DEFAULT_OPTION,
    JOptionPane.WARNING_MESSAGE,
    null, options, options[0]);
if (rep3==JOptionPane.CANCEL_OPTION) ...
if (rep3==JOptionPane.NO_OPTION) ... else
```



## Composant JOptionPane ... exemple

```
// Message et saisie  
String s = JOptionPane.showInputDialog(" Entrez un texte ");
```



## Composant JOptionPane ... exemple

```
// Message et saisie
Object[] choix = {"Manger", "Dormir", "Travailler", "Courir"};
Object s = JOptionPane.showInputDialog(null, "Choisissez",
    "Boîte de choix",
    JOptionPane.INFORMATION_MESSAGE,
    null, choix, choix[2]);
```

