IUT Lyon 1

l'excellence technologique

Prog IHM - Cours 1
Développement interfaces utilisateurs en Java

V. DESLANDRES, I. GUIDARA

veronique.deslandres@univ-lyon1.fr

Avril 2020

Sommaire de ce cours

Organisation du cours

- Volume Horaire: 28h
- 7 séances de cours + TP

- Modalités d'évaluation (cours à distance, COVID-19):

  - Note de TP (moyenne des 4 TP, 70%)
  - DM final sur les concepts vus en TPs (30%)
  - Bonus/malus sur l'implication (fréquence / pertinence des interactions, 2 pts)

- Contenu:

  - Bases d'IHM, Interface utilisateur, Gestion événementielle
  - Lien avec les BD (JDBC)

Programme du Module

- Introduction sur les interfaces utilisateurs
- Bases de l'API Swing : **Conteneurs, Composants**
- **Création** des interfaces
- Gestionnaire de **dispositions**, mise en page :
  - **`layout`**
- Gestion des **événements**
- Focus spécifiques :
  - Présentation **table (`JTable`)**, fenêtres de **dialogue**
  - **`JDBC`** : `API` liens avec les bases de données

Introduction : les interfaces utilisateur (IHM, GUI)

Principes de base

- **Interface Console**
  - C'est le programme qui pilote l'utilisateur, en le sollicitant quand nécessaire pour qu'il fournisse des données

    ➔ Dialogue **en mode texte et séquentiel**

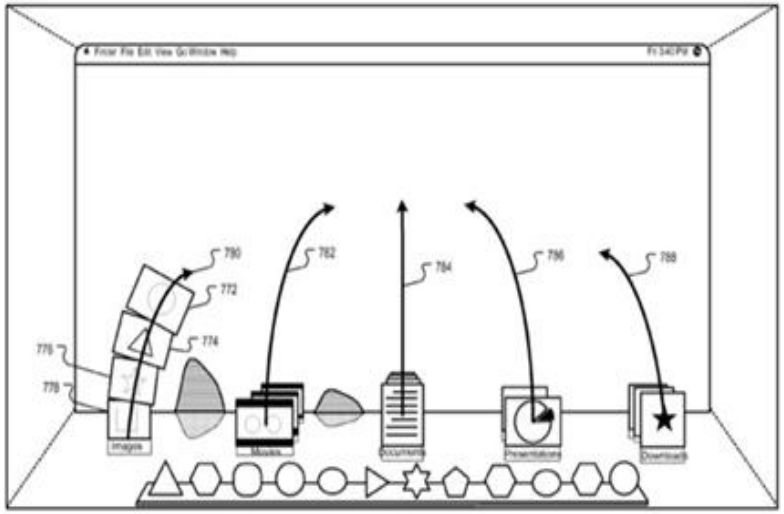
    dans une fenêtre appelée « Console »

- **Interface graphique (GUI – Graphical User Interface)**
  - L'utilisateur pilote le programme, qui réagit à ses demandes (sélection d'articles, d'item de menu, clic bouton,…)

- Chaque action de l'user = événement
- Programmation « **évenementielle** »

Fondamentaux de l'IHM

- L'interface visible d'une application est constituée de 2 éléments :



(1) Les **conteneurs** :

Contiennent des objets graphiques qui peuvent être des composants ou d'autres conteneurs

(2) Les **composants atomiques** : boutons, cases à cocher, zone de texte, slider, etc.

Généralités

- Une interface utilisateur se compose de :
  - Une **fenêtre** de travail
  - Une **zone** où afficher les composants graphiques dans cette fenêtre de travail :
    - Un panneau (**Panel)**
  - Des **composants** à insérer dans cette fenêtre
    - Boutons, cases à cocher, menus, barre de tâches,…
  - Une **mise en page** des composants
    - Le **Layout**: mis bout à bout, centrés, en tableau,…
  - La **représentation graphique** des composants
    - Couleur, forme, image,…
  - Des gestionnaires **d'évènements**
    - Répondre aux actions de l'utilisateur

Java propose

- Des **composants graphiques**
  - `Widgets`

- Des classes de **gestion de la position** des composants sur la fenêtre
  - `LayoutManager`
- Des éléments de **représentation graphique**
  - `Color, Font, Graphics, Point, Rectangle, Image, Icon...`
- Des mécanismes de **gestion d'événements**
  - `java.awt.events`

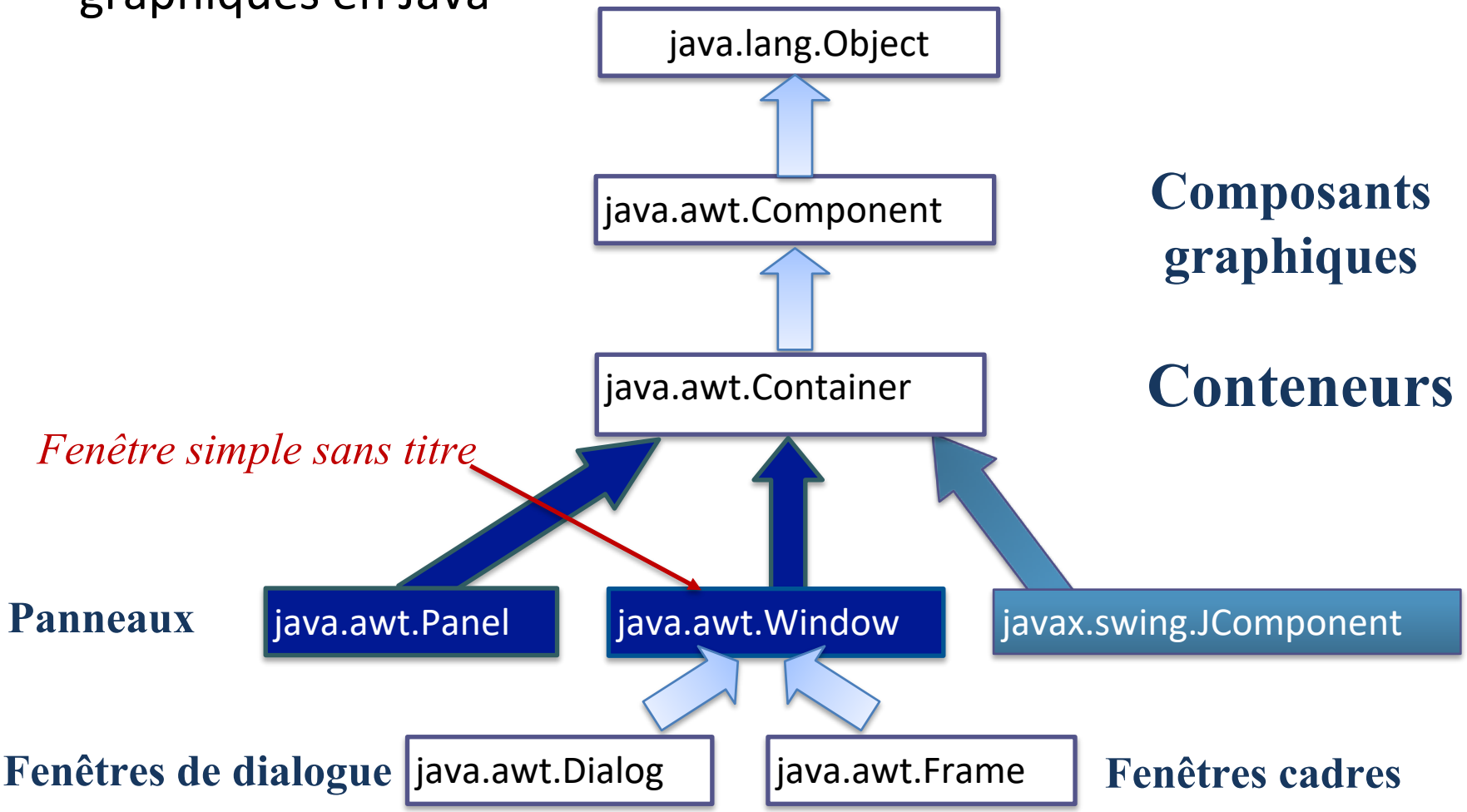
AWT, SWING

- Première bibliothèque graphique JAVA: **AWT**
  - Package: `java.awt`
  - Utilisation du code qui dépend du système d'exploitation
  - Une gestion des événements
  - Composants limités

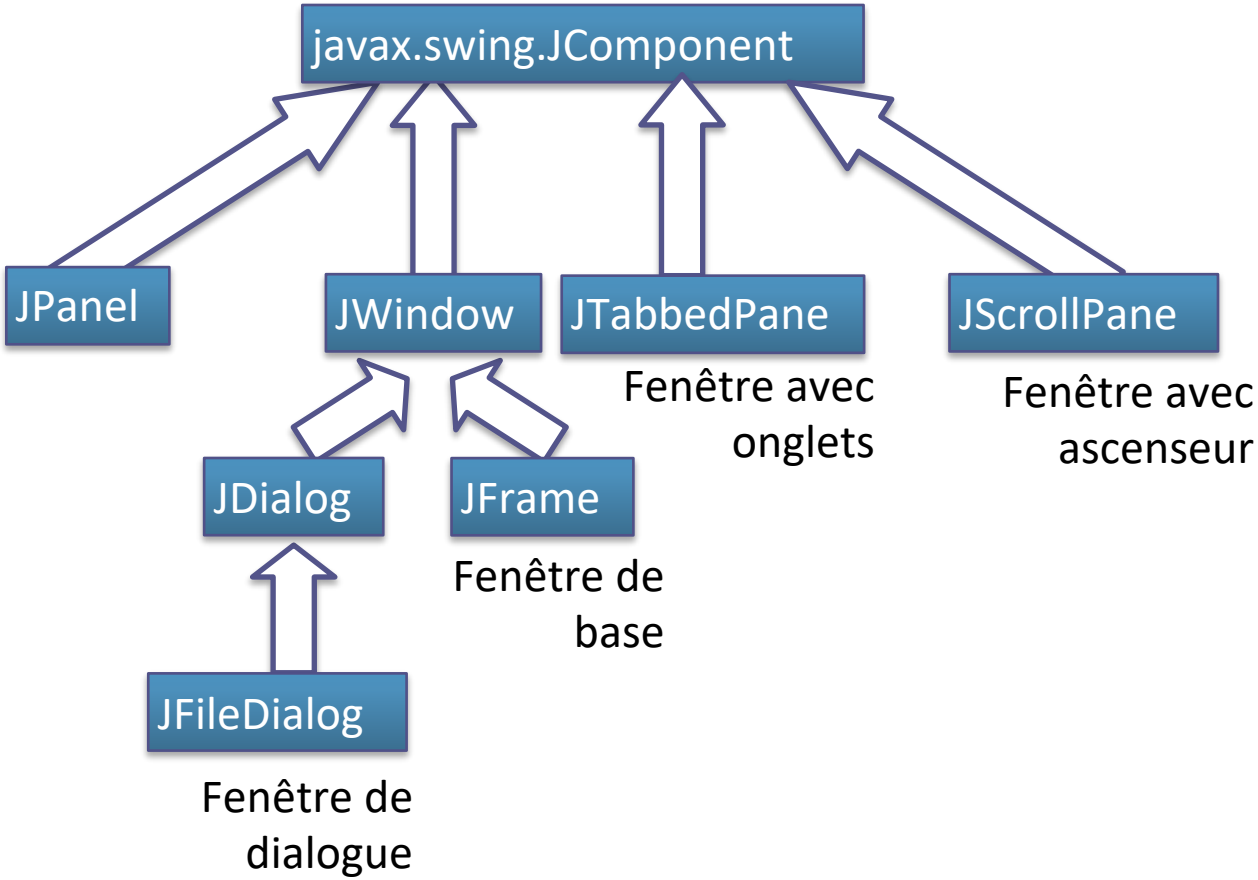
- Nouvelle bibliothèque graphique JAVA: **SWING**
  - Package: `javax.swing`
  - Plus riche et plus personnalisable
  - Construite sur AWT, fournit des composants plus performants

Arborescence des packages AWT/SWING

- Hiérarchie d'héritage des principaux éléments des interfaces graphiques en Java



Arborescence des packages SWING

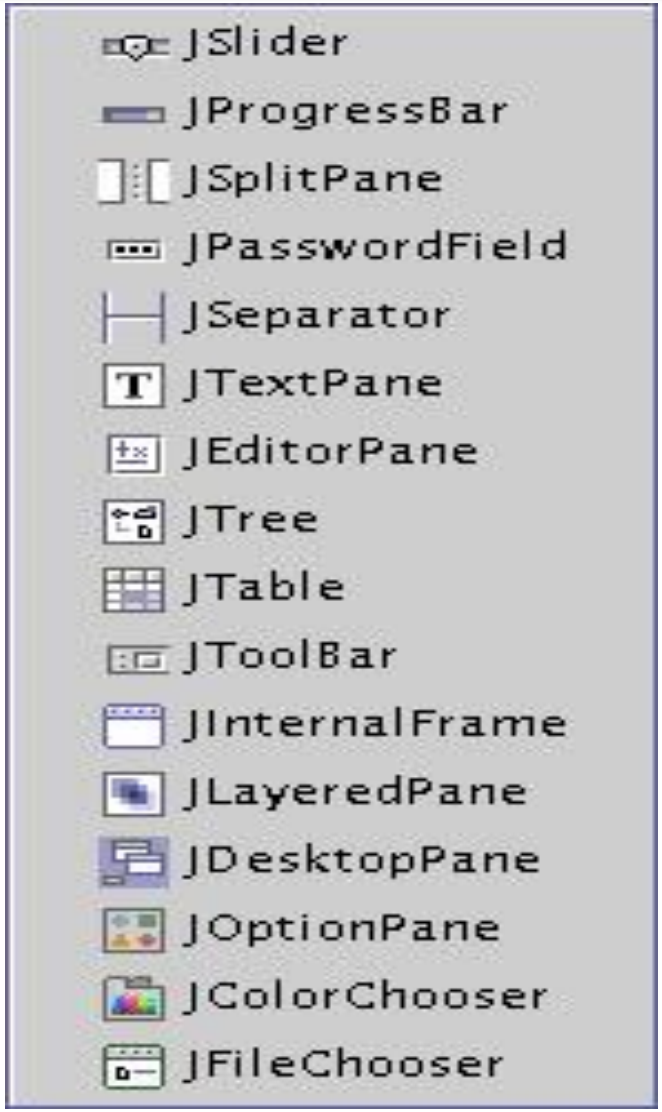


Bases de Swing Composants, Conteneurs

Composants graphiques (SWING)

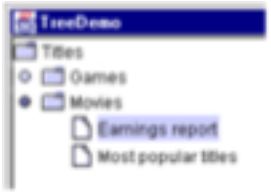
- Les **widgets**
  - JLabel
  - JButton
  - JToggleButton
  - JCheckbox
  - JRadioButton
  - ButtonGroup
  - JComboBox
  - JList
  - JTextField
  - JTextArea
  - JScrollBar
  - JMenuBar
  - JPopupMenu
- Les **containers**
  - JWindow
    - JFrame
    - JDialog
      - JFileDialog
  - JPanel
    - Applet
  - JTabbedPane
  - JScrollPane

Composants graphiques (SWING)

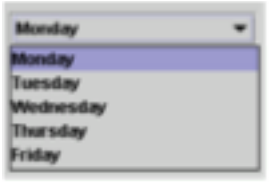


JButton

JSlider



JTree

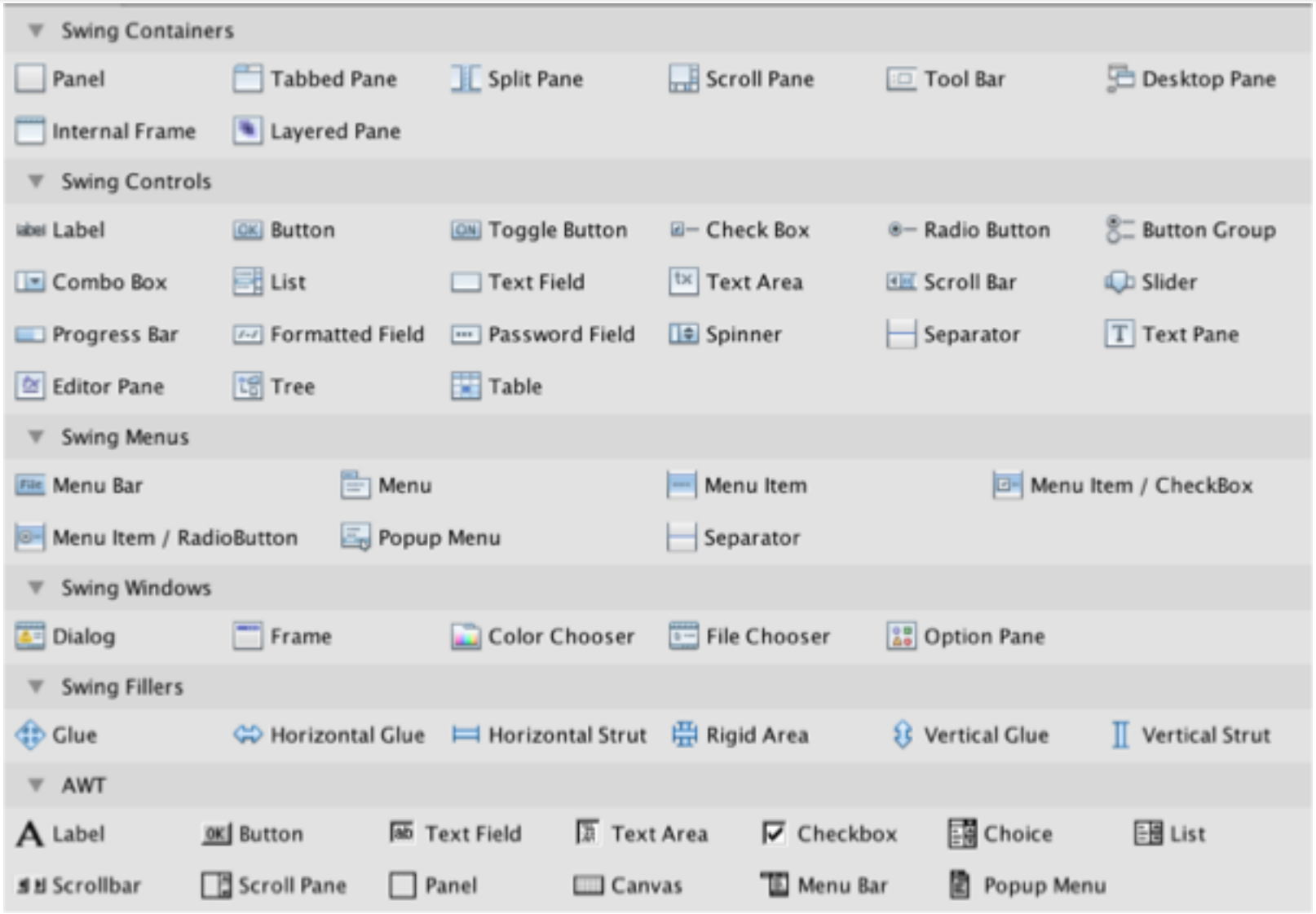


JComboBox

JTextField

JProgressBar

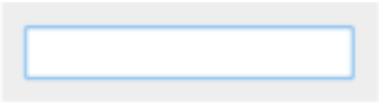
Composants graphiques (SWING)



Composants Texte de la Swing

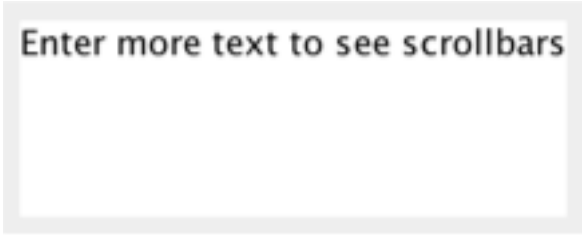
JTextField

```
JTextField numTel = new JTextField("",10);
```



JTextArea

```
JTextArea commentaire=new JTextArea("Enter more text to see scrollbars", 10, 10);
```



JLabel et JButton

JLabel

```
JLabel texte=new JLabel("Texte");
```

Texte

JButton

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

OK

JCheckBox

```
JCheckBox box1=new JCheckBox("BOX1", true);
JCheckBox box2=new JCheckBox("BOX2");
```

BOX1 BOX2

JRadioButton

pour cocher un élément par défaut (facultatif)

```
JRadioButton radio1=new JRadioButton("Radio1", true);
JRadioButton radio2=new JRadioButton("Radio2");
ButtonGroup groupRadio=new ButtonGroup();
groupRadio.add(radio1);
groupRadio.add(radio2);
```

Radio1 Radio2

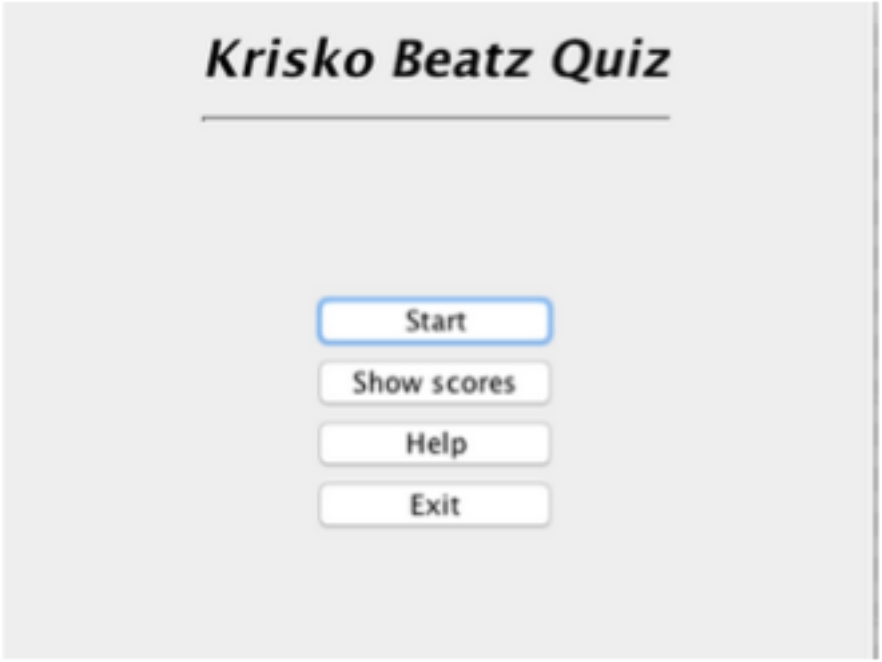
Groupe : pour gérer choix exclusif (facultatif)

Différentes catégories de composants (1/2)

Composants atomiques	JLabel, JButton, JCheckBox, JRadioButton, JToggleButton, JComboBox, JScrollBar, JSeparator, JSlider, JSpinner, JProgressBar
Composants complexes	JTable, JTree, JList, JFileChooser, JColorChooser, JOptionPane
Composants textuels	JTextField, JFormattedTextField, JPasswordField, JTextArea, JTextPane, JEditorPane

Différentes catégories de composants (2/2)

Composants de menus	JMenuBar, JMenu, JPopupMenu, JMenuItem, JCheckboxMenuItem, JRadioButtonMenuItem
Composants Conteneurs	JPanel, JScrollPane, JSplitPane, JTabbedPane, JDesktopPane, JToolBar
Composants de haut niveau	JFrame, JDialog, JWindow, JInternalFrame, JApplet



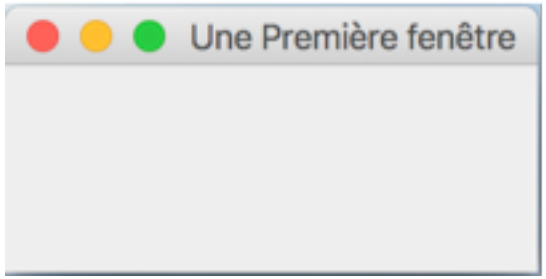
Création d’une fenêtre

Création + appel dans le *main*

```
public class MaFenetre extends JFrame {
    private JButton btVoir, btDebut, btPrecedent, btSuivant, btFin;

    // Constructeur
    public MaFenetre() {
     initComponents();
     this.setTitle("Une Première fenêtre");
     this.setResizable(false);
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

→ *Fenêtre non redimensionnable*

→ *Quitter qd on ferme la fenêtre*

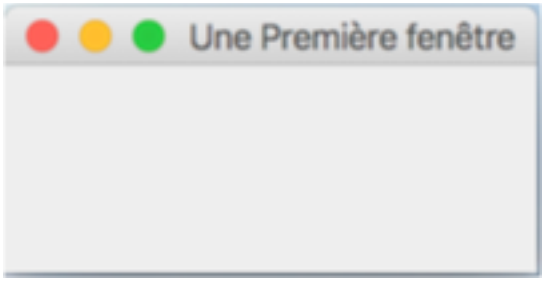


Création + appel dans le *main*

```
public class MaFenetre extends JFrame {
    private JButton btVoir, btDebut, btPrecedent, btSuivant, btFin;

    // Constructeur
    public MaFenetre() {
     initComponents();
     this.setTitle("Une Première fenêtre");
     this.setResizable(false);
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private void initComponents() {
    btVoir = new javax.swing.JButton("Voir");
    btDebut = new javax.swing.JButton("Début");
    btPrecedent = new javax.swing.JButton("Précédent");
    btSuivant = new javax.swing.JButton("Suivant");
    btFin = new javax.swing.JButton("Fin");
```

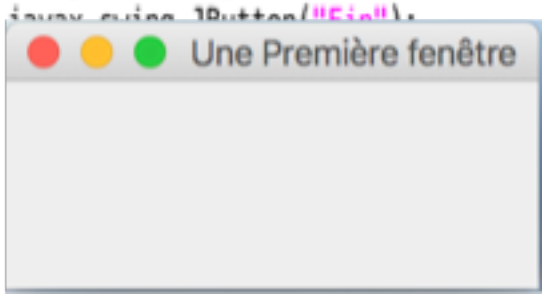
```
// Appel
public static void main(String args[]) {

 MaFenetre premFen = new MaFenetre();
 premFen.pack();
 premFen.setVisible(true);

} // fin du main
```

Adapter la taille de la fenêtre à ses composants



Création + appel dans le *main*

```
public class MaFenetre extends JFrame {
    private JButton btVoir, btDebut, btPrecedent, btSuivant, btFin;

    // Constructeur
    public MaFenetre() {
     initComponents();
     this.setTitle("Une Première fenêtre");
     this.setResizable(false);
     this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

private void initComponents() {
    btVoir = new javax.swing.JButton("Voir");
    btDebut = new javax.swing.JButton("Début");
    btPrecedent = new javax.swing.JButton("Précédent");
    btSuivant = new javax.swing.JButton("Suivant");
    btFin = new
```




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

 MaFenetre premFen = new MaFenetre();
 premFen.pack();
 premFen.setVisible(true);

} // fin du main
```

Écriture du *main* par l'IDE

```
public static void main(String args[]) {

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new MaFenetre().setVisible(true);
    }
});
```

Création de fenêtre

`setDefaultCloseOperation()` de `JFrame`

JFrame.EXIT_ON_CLOSE
— quitte l'application

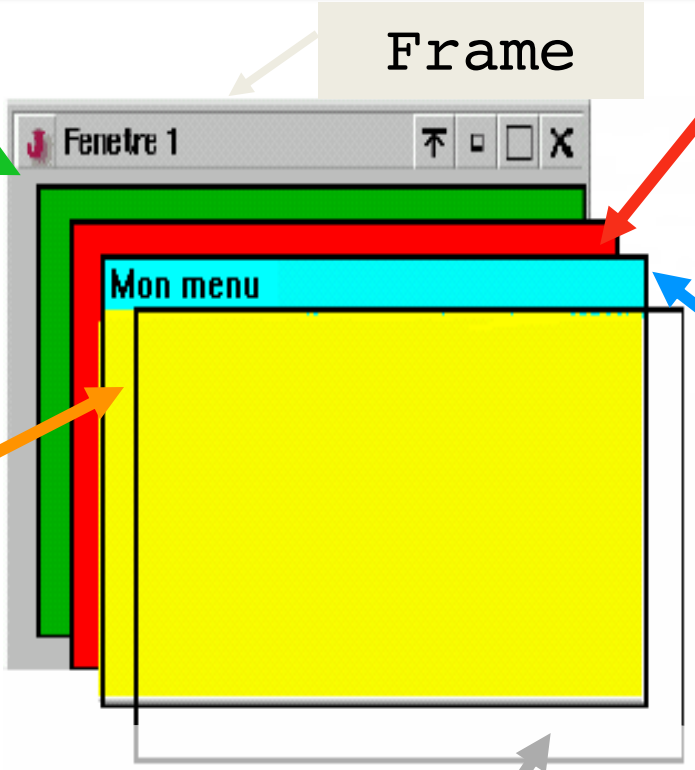
JFrame.HIDE_ON_CLOSE (choix par défaut)
— Cache la fenêtre sans quitter l'application
— *Quand on a plusieurs fenêtres, on peut souhaiter fermer une fenêtre et continuer l'application*

JFrame.DISPOSE_ON_CLOSE
— Rend la main à une fenêtre **parent** tout en fermant la fenêtre courante

JFrame.DO_NOTHING_ON_CLOSE
— Ignore la demande de fermeture

Construction en couches

Panneau racine
(root pane)
un seul dans chaque conteneur racine, il contient les autres éléments



Panneau couche
(layered pane)
conteneur permettant de positionner la barre de menu et le panneau contenu

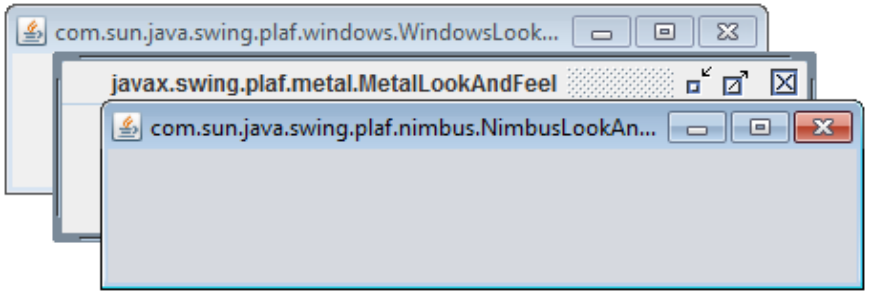
Panneau contenu
(content pane)
conteneur pour les composants visibles

Barre de menu
(menu bar)
optionnelle, contient les différents menus de commandes

Panneau vitre
(glass pane)
caché, il permet d'intercepter les différents événements de la fenêtre

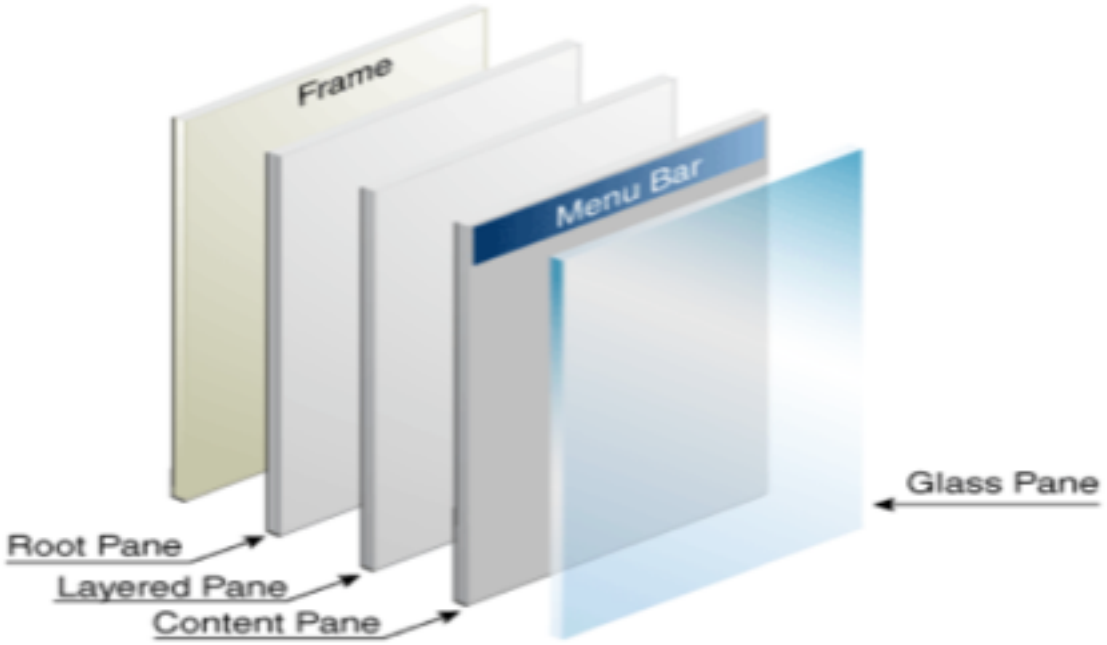
ContentPane

- Les **JRootPane**, le **JLayeredPane** et le **GlassPane** sont utilisés par Swing pour implémenter le *look and feel*
  - n'ont donc pas à être considérés dans un 1er temps par le développeur



- Le niveau qui nous intéresse pour déposer un composant sur une fenêtre *JFrame* est le **ContentPane**
  - **C'est lui qui contient les composants** (boutons et autres widgets)
  - C'est une instance de la classe Container

Accès par niveau

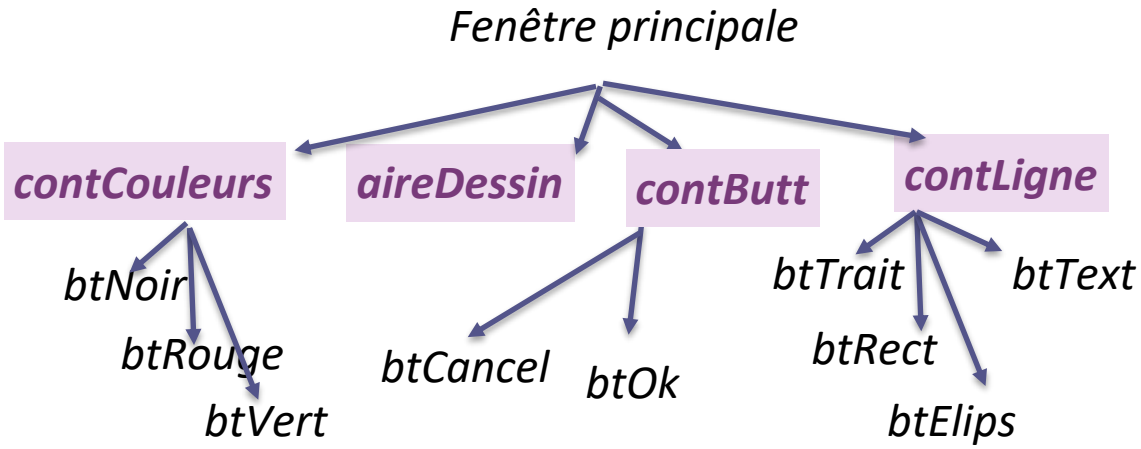


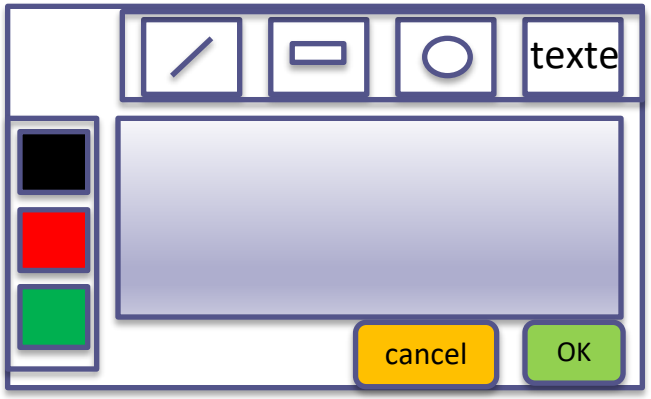
Par exemple pour ajouter un bouton sur une fenêtre Frame :

```
maFrame.getContentPane().add(new JButton("OK"));
```

La fenêtre `java.swing.JFrame`

- Construire une IHM, c'est mettre des composants les uns à l'intérieur des autres, dans le bon ordre
- Dans cet **arbre d'instanciation**, la flèche signifie « contient » :





(analogie : arborescence de fichiers et de répertoires)