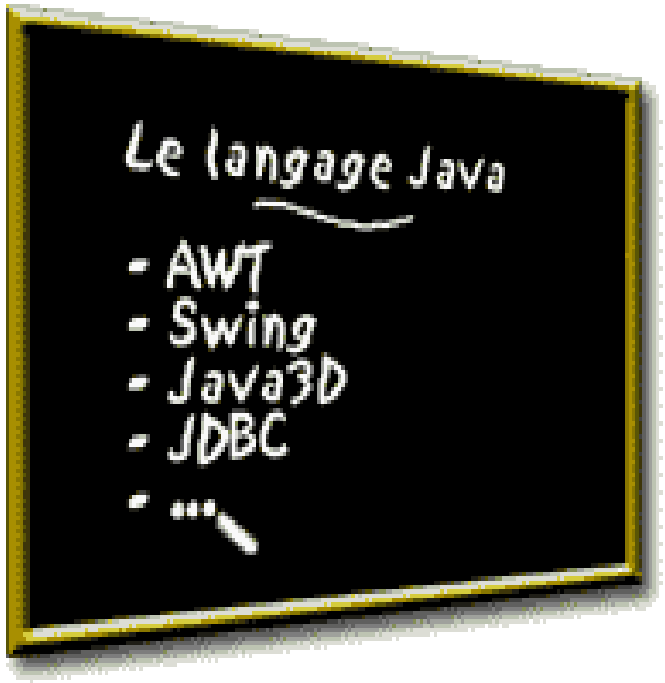
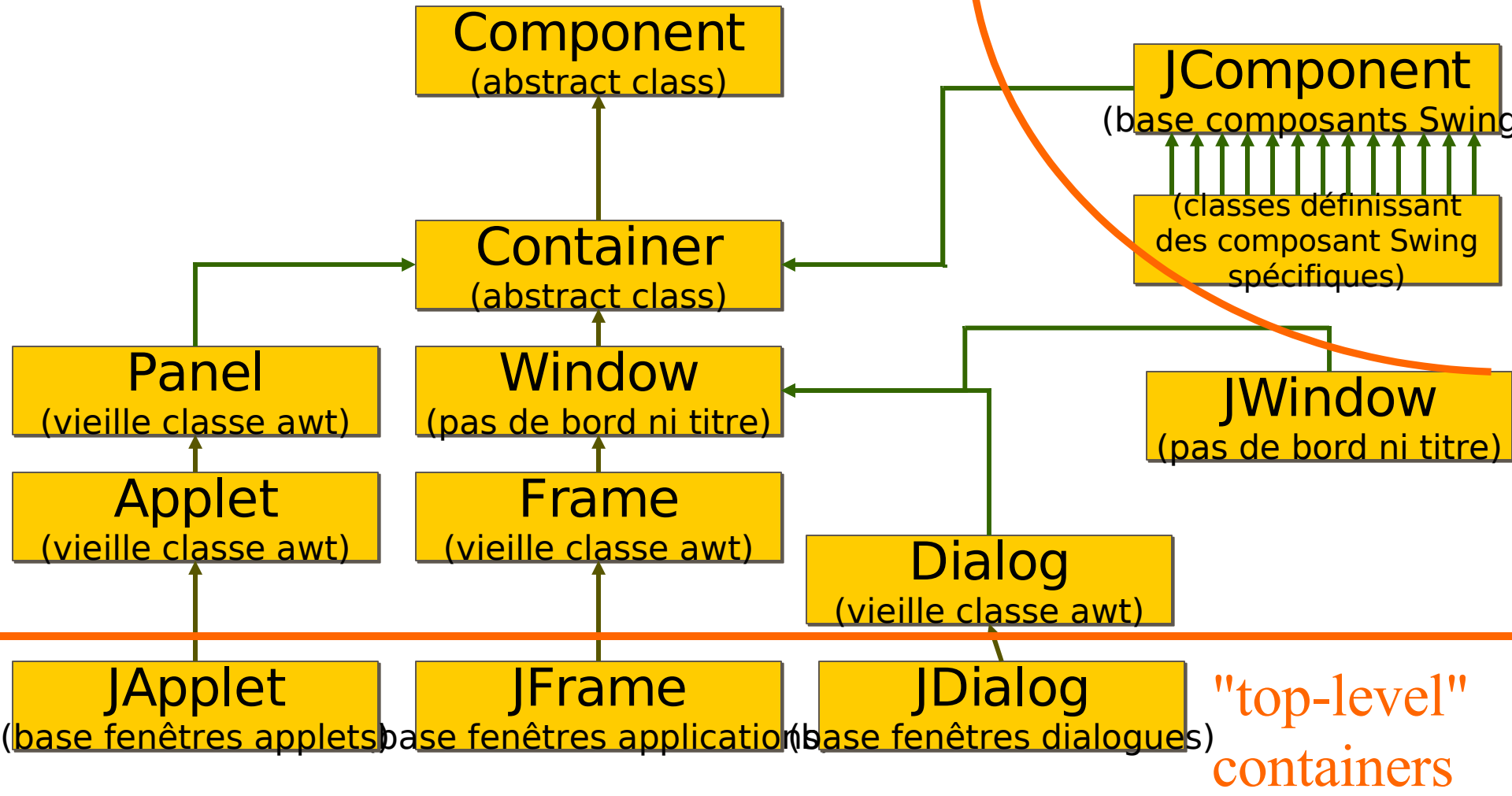
Cours 2 : Placer des composants dans une fenêtre



- Les composants graphiques
- Les gestionnaires de mise en page : layout
- Un exemple complet
- Les menus

Les composants graphiques : Hiérarchie



Commentaires sur le haut de la hiérarchie

- Ce qui commence par J : swing, le reste : awt
- Classe Component : fournit les méthodes d'affichage et de traitement des événements
- Classe Container :
  - Un conteneur permet d'inclure d'autres composants
  - Les composants graphiques se mettent dans un container

Les composants de haut niveau ou "top-level" containers : de 3 types

- JFrame : fenêtre principale d'une application.
  - Peut contenir des menus, et d'autres composants.
  - Utilisé par héritage pour créer des classes de fenêtres spécifiques à une application
- JDialog: fenêtre de dialogue avec l'utilisateur
  - Utilisé par héritage pour créer des boîtes de dialogues spécifiques
- JApplet : pour les applications web téléchargées sur le poste client à partir d 'un serveur web (vu plus tard)
  - Utilisé par héritage pour créer des applets spécifiques

Les composants à placer dans les "top-level" containers : la classe JComponent

- Classe de base de tous les composants Swing utilisés pour placer à l'intérieur d'une fenêtre de haut niveau

- Sous-classes = composants standard =
  - Menus (JMenu, JMenubar, JMenuItem)
  - Boutons (JButton), checkboxes (JCheckBox),
  - Zones de textes (JLabel, JTextField, JTextArea),
  - etc.

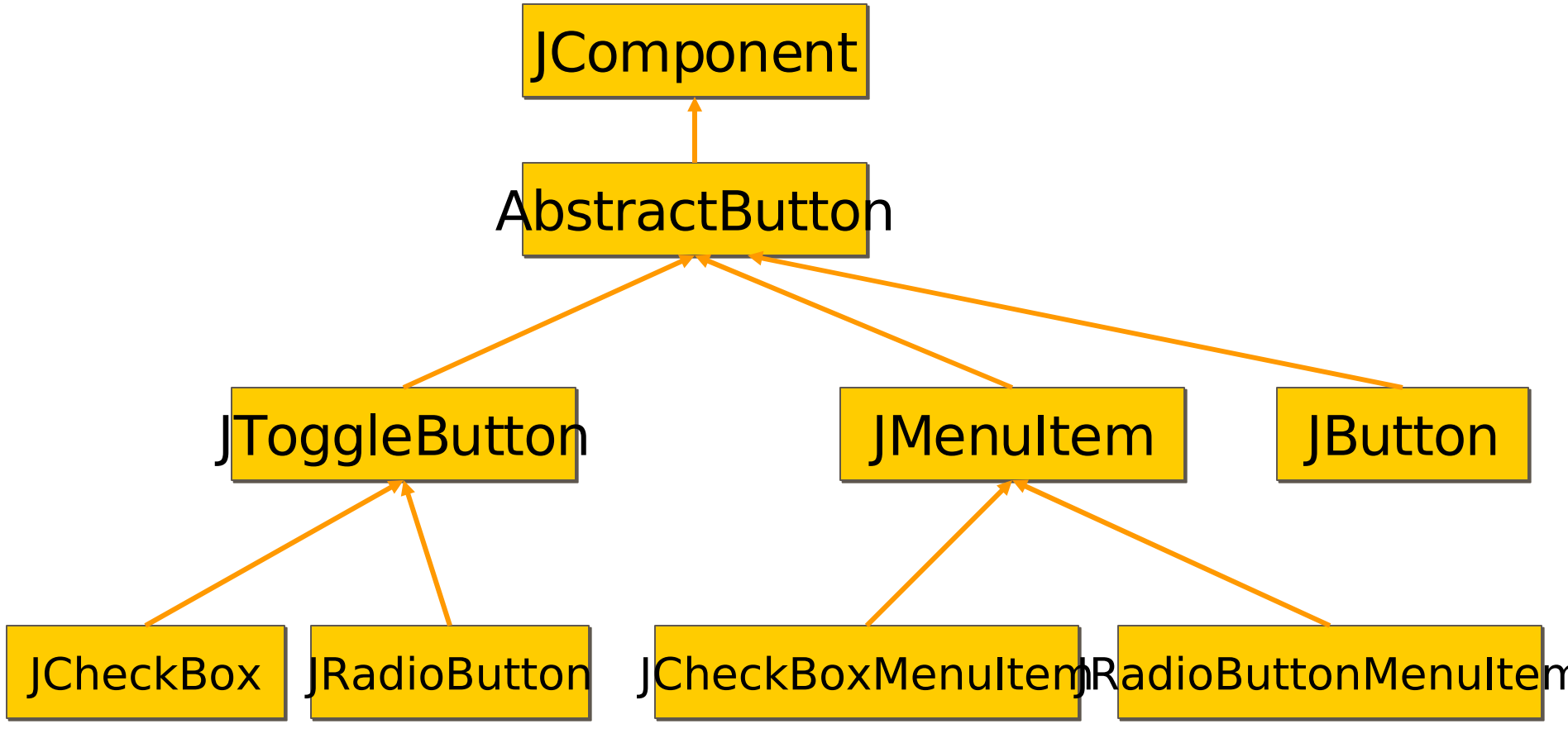
Attributs standards des composants

- Un composant peut être :
  - enabled : quand le composant l'est, il est actif (~ peut répondre à des actions de l'utilisateur)
  - visible ou non à l'écran
  - opaque : un composant est opaque s 'il est rempli avec sa couleur de fond
- Tous ces attributs sont private. On y accède ou on les change par les accesseurs habituels (setXXX, getXXX).

Méthodes standards des composants : exemples

- **Les méthodes générales :** JComponent comp=new ...
  - comp.setBackground(Color.yellow);
  - comp.setOpaque(true);
  - comp.setFont(new Font("Serif", Font.BOLD, 24));
  - comp.setEnabled(false);
- **Les méthodes spécifiques :**
  - **Les constructeurs :** plusieurs, dépendent de la classe
  - méthodes spécifiques à chaque classe de composant

Une branche héritant de JComponent les boutons et les items de menus



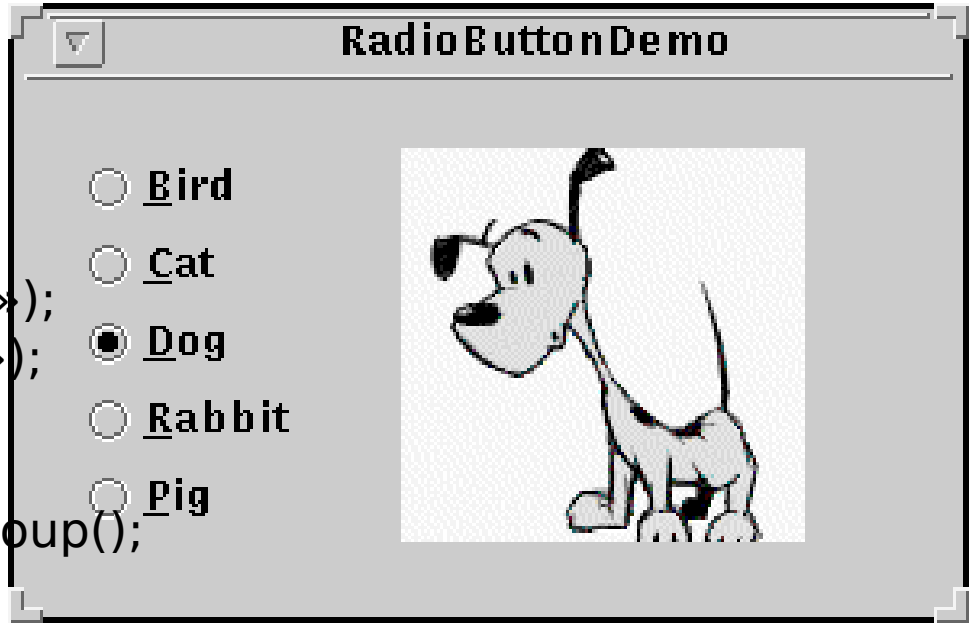
Les boutons

- JButton définit un bouton standard

```
JButton b=new JButton(«OK  »);
```

- JToggleButton définit un bouton à deux états. Il est en deux versions :
  - JCheckBox
  - JRadioButton : nécessairement en groupe et un seul du groupe est dans l'état «pressé»


```
JCheckBox bChin = new JCheckBox(«  Chi
JCheckBox bGlasses = new JCheckBox(«
bGlasses.setSelected(true);
```



1. Création des boutons

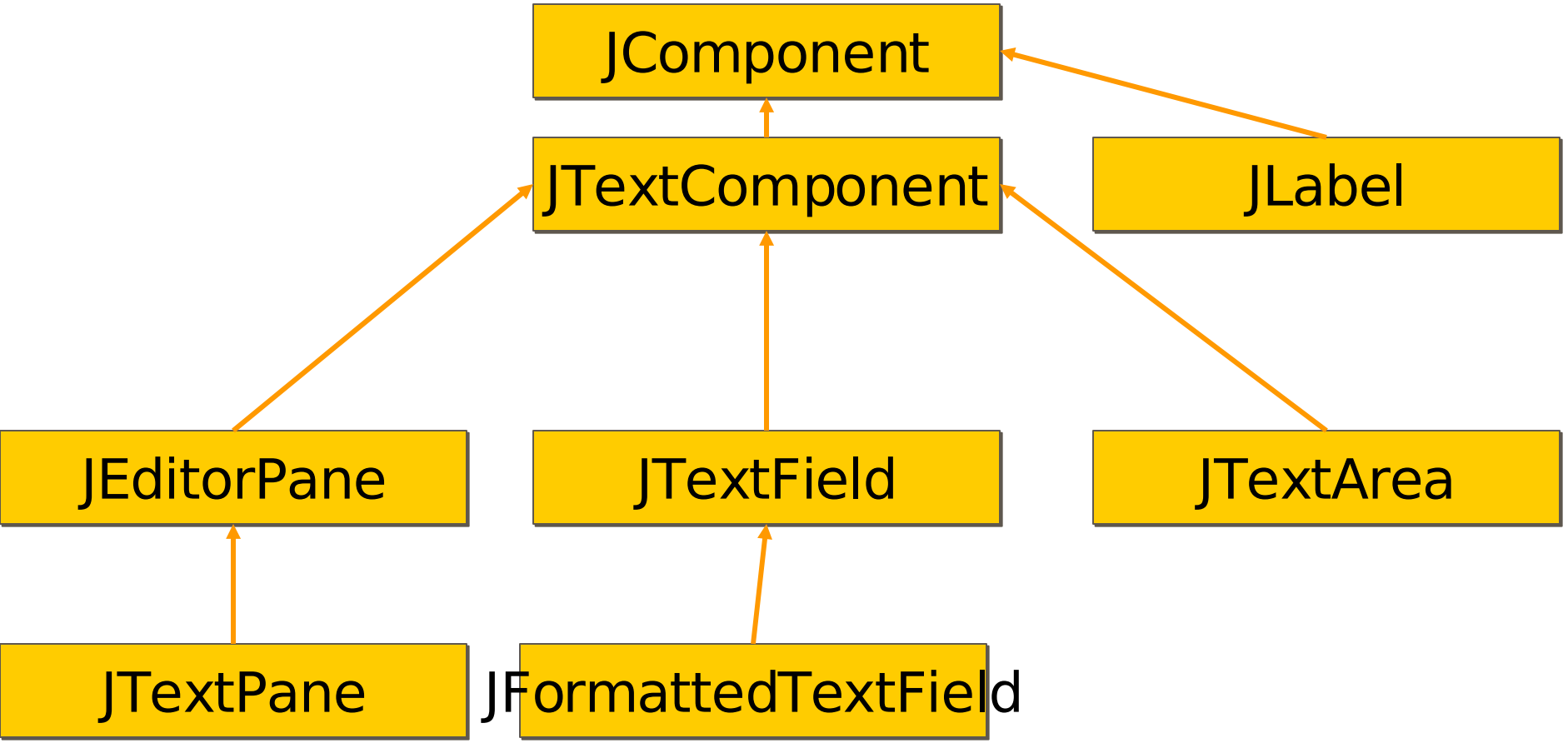
```
JRadioButton bBird, bCat;

bBird= new JRadioButton («  Bird  »);
bCat = new JRadioButton («  Cat  »);
....
```

2. Création du groupe

```
ButtonGroup group=new ButtonGroup();
group.add(bBird);
group.add(bCat);
...
```

Quelques sous-classes de JComponent
les composants de texte

Grande variété de ces composants dans Swing



Les composants de texte

- JLabel : le plus simple. Complètement passif. Non éditable
- JTextField : idem, une seule ligne de texte (pas de retour chariot), éditable
- JTextArea : plusieurs lignes de texte, présence d 'un ascenseur, éditable
- JEditorPane et JTextPane : composants plus complexes permettant d'implémenter des fonctions d'édition élaborées , et gestion de html, rtf. JTextPane permet d'avoir en plus des images.

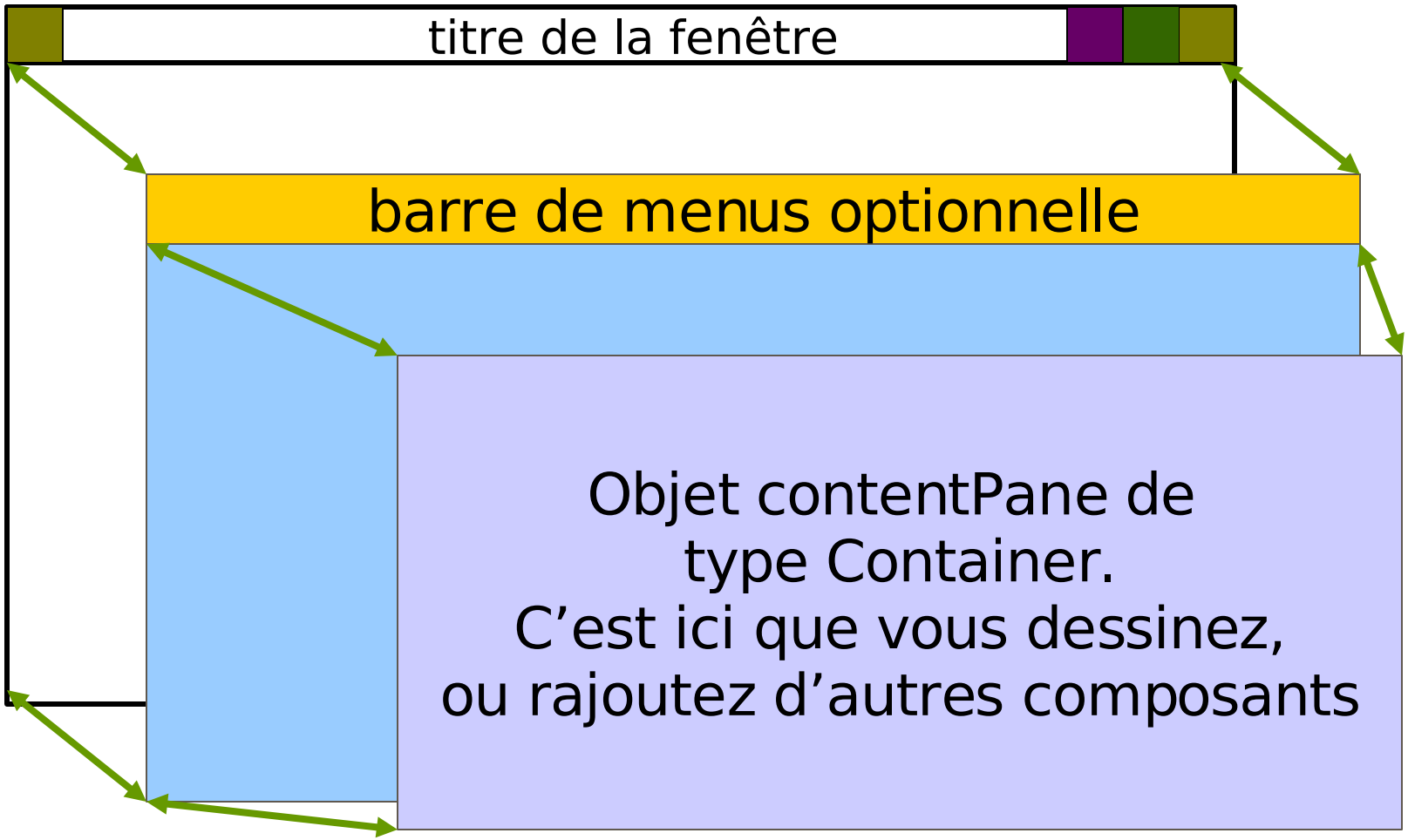
Méthodes des composants de texte
sous-classe de JTextComponent

- Méthodes :
  - String getText() : récupère le texte présent dans la zone de texte
  - void setText(String s) : remplace le texte présent dans la zone de texte par s
- Exemples :

```
JTextField tNom= new JTextField("Récupère",20);
String s = tNom.getText();

JTextArea jta = new JTextArea();
jta.setText("bonjour\nsalut\nau revoir\n");
jta.append("ciao\n");
JScrollPane scrollpane=new JScrollpane(jta);
```

Ajout d'un composant à une fenêtre de haut niveau

- Constitution d'un "top-level" container



Ajouter un composant à un "top-level" container : quelques règles

- Chaque "top-level" container possède une aire de travail, une instance de la classe Container.
- On n'ajoute pas un composant directement à un top-level container (JFrame, JApplet ou JDialog) mais à son aire de travail
- On accède à cette aire de travail en adressant la méthode getContentPane() (présente dans chacune des classes "top-level") à la fenêtre

```
JFrame fen=new JFrame("ajout");
Container c=fen.getContentPar
c.add(.....)
```

Une instance de JComponent créée avec un constructeur

Disposition et taille des composants : les Layout manager

- La disposition et la taille des composants se fait grâce à un gestionnaire de mise en page (layout) associé à tout container
- Ce gestionnaire de mise en page place le composant en suivant des règles spécifiques et relativement à la taille de la fenêtre
- Il y a plusieurs classes de layout managers dans awt et swing

Classes de Layout

- FlowLayout : place les objets dans des rangées successives.

- BorderLayout : place les objets contre les quatre bords et au centre. Placement géographique

- GridLayout : place les objets dans une grille dont vous donnez le nombre de lignes et de colonnes

- BoxLayout : les composants sont mis dans une ligne ou une colonne

Changer de gestionnaire de mise en page

- **Certains composants ont des gestionnaires de mise en page par défaut**
  - JPanel : FlowLayout
  - contentPane d'une JFrame : BorderLayout
- S 'il ne convient pas, on peut en changer en adressant au composant la méthode :

setLayout(LayoutManager l)

Exemple : changer le layout du contentPane d'une JFrame :

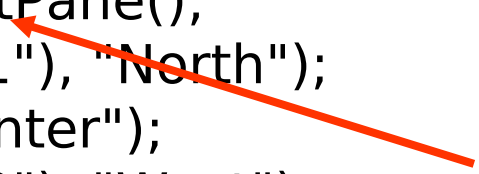
```
JFrame fen=new JFrame("changement de mise en page");
Container c=fen.getContentPane();
FlowLayout flow = new FlowLayout();
c.setLayout(flow);
```

La classe BorderLayout :

place les objets contre les quatre bords et au centre.
Placement géographique (N/S/W/E/C).

```
public class DemoBorderLayout extends JFrame {
public DemoBorderLayout(String titre, int w, int h)  {
  super(titre);
  this.initialise(w,h);
  this.show();  }

public void initialise(int w, int h) {
  this.setSize(w,h);
  Container c=this.getContentPane();
  c.add(new JButton("Button 1"), "North");
  c.add(new JButton("2"), "Center");
  c.add(new JButton("Button 3"), "West");
  c.add(new JButton("Long-Named Button 4"),
"South");
  c.add(new JButton("Button 5"), "East");
}
```

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

{new DemoBorderLayout ("Essai
BorderLayout",
600, 400);  }
```

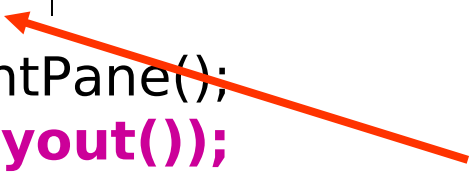


La classe FlowLayout :

de gauche à droite sur des lignes successives, positionnement des composants s'adapte si on modifie la taille de la fenêtre

```
public class DemoFlowLayout extends
JFrame {
public DemoFlowLayout(String titre, int
w, int h)  {
  super(titre);
  this.initialise(w,h);
  this.show();  }

public void initialise(int w, int h)  {
  this.setSize(w,h);
  Container c=this.getContentPane();
  c.setLayout(new FlowLayout());
  c.add(new JButton("Button 1"));
  c.add(new JButton("2"));
  c.add(new JButton("Button 3"));
  c.add(new JButton("Long-Named
```

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

new DemoFlowLayout ("Essai de
FlowLayout",
600, 400);
    }
}
```



GridLayout : arrange les composants dans une grille rectangulaire avec le nombre de lignes et de colonnes déclarées

Composants placés de haut en bas et de gauche à droite.

Différents constructeurs possibles

```
...
c.setLayout(new GridLayout(3,2));
...
```

```
...
c.setLayout(new GridLayout(3, 2, 20, 60));
c.add(new JButton("Button 1"));      // range par ligne d'abord
c.add(new JButton("2"));
....
```

Le gestionnaire BoxLayout pour mettre les composants (vertical ou horizontal)

Le constructeur exige le composant cible et l 'axe :

public BoxLayout(Container target, int axis)

axis est défini par une constante de classe de BoxLayout :

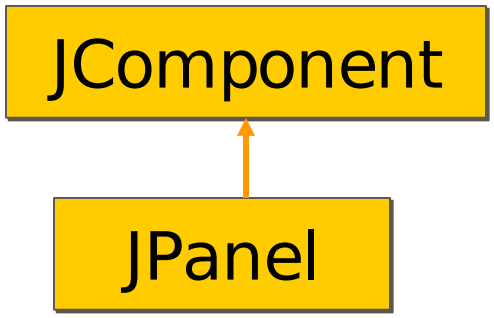
```
public void init     {
        Container c=this.getContentPane();
        c.setBackground (Color.white);
        c.setLayout(new BoxLayout(c, BoxLayout.Y_AXIS)
        ....
```

Y_AXIS et X_AXIS

Composants qui aident à organiser d 'autres composants : la classe JPanel (1)

- Permet de :
  - subdiviser un container en plusieurs zones
  - chacune de ces zones pouvant :
    - adopter un gestionnaire de mise en page différent
    - contenir plusieurs composants ou d'autres sous-panneaux

Démo



Gestionnaire de mise en page d'un panel

- Par défaut le gestionnaire de mise en page d'un panel est de type FlowLayout
- 2 constructeurs de JPanel
  - JPanel() : crée un panneau muni du gestionnaire par défaut (FlowLayout)
  - JPanel(LayoutManager) : crée un panneau avec le gestionnaire précisé en paramètre
- L'ajout d'un composant au panel se fait grâce à la méthode add()

Analyse de l ’exemple

- Une fenêtre contenant deux panels :
  - 1 panel au centre muni d ’un gestionnaire de type Grille 2*2 (GridLayout)
  - 1 panel au Sud muni du FlowLayout par défaut

⇒ Pour la modularité, définition d ’une méthode initialise() de la fenêtre qui dispose les composants graphiques, elle-même décomposée en :

- une méthode creePanelCentre() renvoyant un JPanel
- une méthode creePanelSud() renvoyant un JPanel

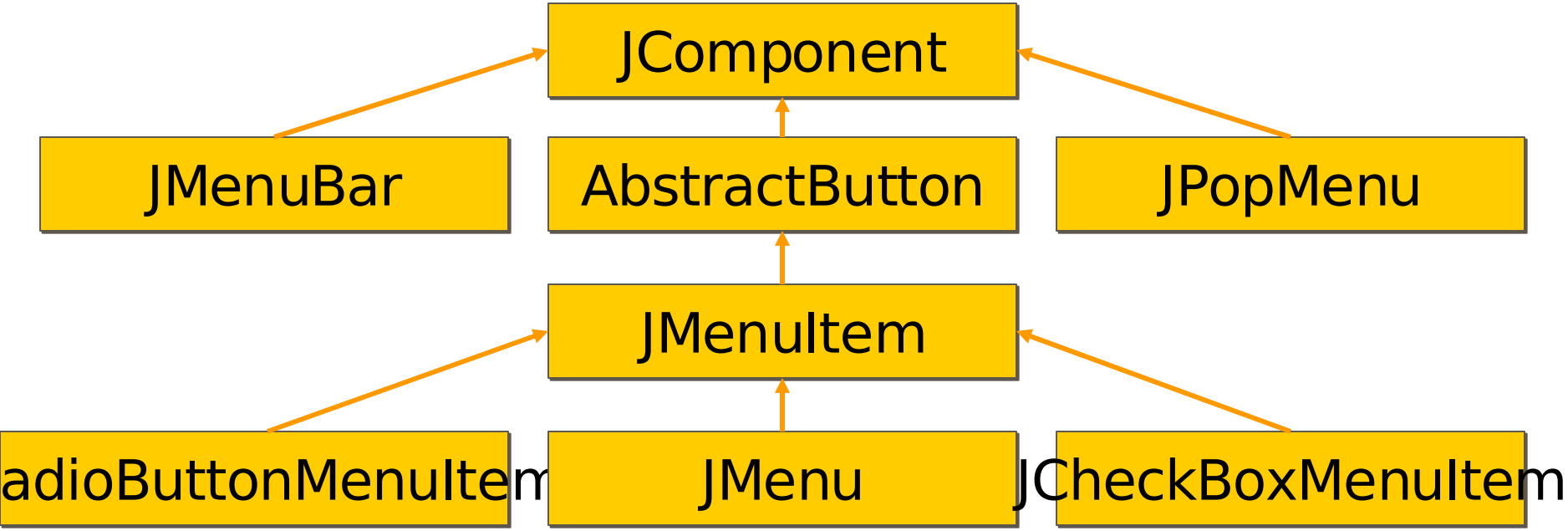
```
public class FrameAvecPanel extends JFrame
        {


public FrameAvecPanel(String title, int
w, int h)  {
  super(title);
  this.initialise();
  this.setSize(w,h);
  this.show();
   }

public void initialise()  {
    Container c=this.getContentPane();
    JPanel pEntree =this.creePanelCentre();
    c.add(pEntree,  "Center");
    JPanel pBoutons=this.creePanelSud();
    c.add(pBoutons,"South");
}
```

```
// ou en une seule instruction
c.add ( this.creePanelCentre(),
"Center" );
```

```
public JPanel creePanelCentre()  {
   JPanel pCentre=new JPanel(new GridLayout(2,2,80,20));

   pCentre.add(new JLabel("Donnez votre prénom"));
   JTextField tPrenom = new JTextField("");
   pCentre.add(tPrenom);

   pCentre.add(new JLabel("Donnez votre nom"));
   JTextField tNom = new JTextField("");
   pCentre.add(tNom) ;

   return pCentre;
   }
public JPanel creePanelSud()  {
   JPanel pSud = new JPanel();
   JButton bCancel = new JButton ("Cancel") ;
   JButton bok = new JButton ("OK") ;
   pSud.add(bCancel);
   pSud.add(bok);
   return pSud;
   }
```

Comment créer des menus ?

Swing supporte les pop-up menus, les menus contextuels et les menubars



Définir des Menus ⇒ utiliser 3 classes JMenuBar, JMenu, JMenuItem

- JMenuBar : barre des menus placée en haut de la fenêtre d'une application
- Une barre de menu est composée de JMenus
- Un objet JMenu possède un label, et quand on clique dessus, il peut montrer un menu déroulant
- Un item d'un objet JMenu peuvent être un objet de type JMenuItem, JCheckBoxMenuItem ouJRadioButtonMenuItem
- Un objet JMenuItem est un simple élément de menu avec un label. Il peut avoir une icône en plus de son label

Démo

Les méthodes d ’ajout : 3 étapes

- On ajoute une barre de menu à une JFrame grâce à la méthode setJMenuBar(JMenuBar) de JFrame
- Une barre de menu est composée de JMenus qu ’on ajoute par la méthode add(JMenu) de JMenuBar
- On ajoute des items de menus à un JMenu par la méthode add(JMenuItem) de JMenu

Un exemple

this.addMenu();

```
public FenetreSimpleMenu(String titre)
    {super(titre);
     JMenuBar jmb = new JMenuBar();
     this.setJMenuBar(jmb);
     JMenu mdef = new JMenu ("Définir");
     jmb.add(mdef);
    JMenuItem defNom= new JMenuItem ("le nom");
    mdef.add(defNom);
    JMenuItem defPrenom= new JMenuItem ("le prenom");
    mdef.add(defPrenom);
    JMenu maff = new JMenu ("Afficher");
    jmb.add(maff);
```

1

2

3