POO et Programmation Réseaux

Prof. Samira MABTOUL

Ecole Nationale des Sciences Appliquées de Safi

- ENSA-S -

2014-2015

Chapitre 3:
Interface graphique

Interfaces graphiques

- Les interfaces graphiques assurent le dialogue entre les utilisateurs et une application

- En java, il existe deux types de composants graphiques :
  - Composants **AWT** (**Abstract Window ToolKit**) : Composants qui permettent d'écrire des interfaces graphiques dépendantes du système d'exploitation sur lesquelles elles vont fonctionner. Cette librairie utilise le système graphique de la plateforme d'exécution (Windows, MacOS, …) pour afficher les objets graphiques (***package java.awt)***
  - Composants **SWING** : Composants écrit complètement avec java et sont indépendant de la plate-forme d'exécution.

  (package javax.swing)

Le AWT (Abstract Windowing Toolkit)

Le AWT est subdivisé en trois groupes :

- Graphiques : classes traitant des couleurs, polices, images, etc.
- Composants : classes pour les boutons, menus, listes, boîtes de dialogues, etc.
- Layout-Manager : classes pour la configuration (organisation) des composants graphiques dans les conteneurs (comme les fenêtres)

AWT contient aussi un package spécifique pour le traitement des événements (awt.event.*)…

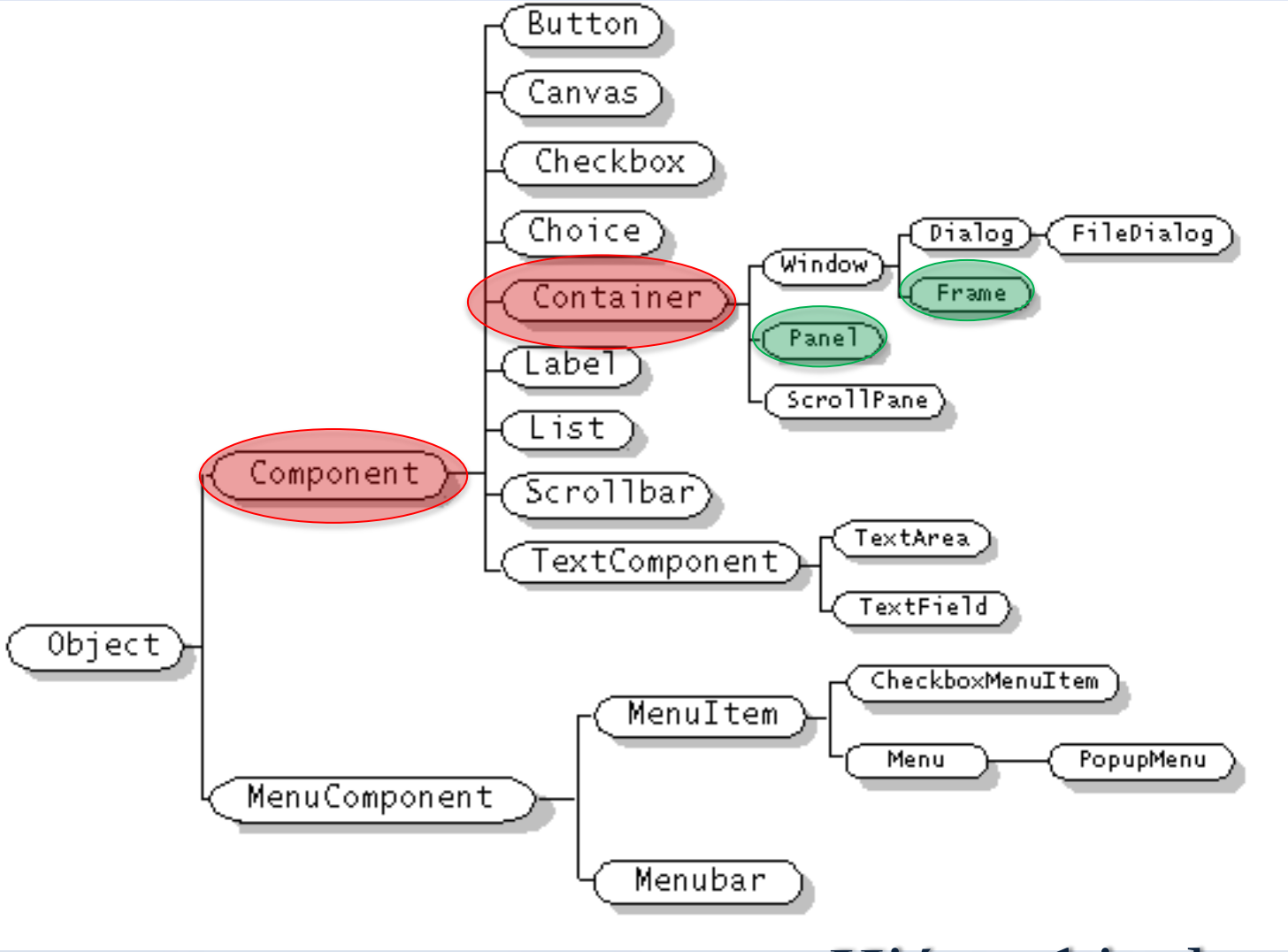
AWT : Conteneurs et Composants

- Une interface graphique en Java est un assemblage de conteneurs (*Container*) et de composants (*Component*) :

  - **Un composant :** est la partie "visible" de l'interface utilisateur Java
    - C'est une sous-classe de la classe abstraite **java.awt.Component**
    - Exemple : les boutons, les zones de textes ou de dessin, etc.

  - **Un conteneur :** est l'espace dans lequel on peut positionner plusieurs composants
    - Sous-classe de la classe **java.awt.Container**
    - La classe Container est elle-même une sous-classe de la classe Component
    - Exemple : les fenêtres, etc.

AWT : Conteneurs et Composants



Hiérarchie des classes

AWT : Conteneurs et Composants

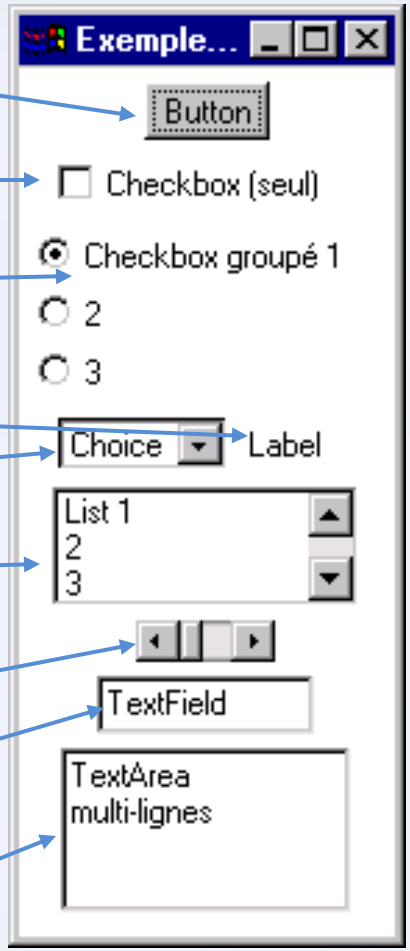
Les deux conteneurs les plus courants sont le **Frame** et le **Panel**

- **Un Frame** représente une fenêtre de haut niveau avec un titre, une bordure et des angles de redimensionnement. La plupart des applications utilisent au moins un Frame comme point de départ de leur interface graphique
- **Un Panel** n'a pas une apparence propre et ne peut pas être utilisé comme fenêtre autonome
  - Les Panels sont créés et ajoutés aux autres conteneurs de la même façon que les composants tels que les boutons
  - Les Panels peuvent ensuite redéfinir une présentation qui leur soit propre pour contenir eux-mêmes d'autres composants

AWT : Conteneurs et Composants

Exemple : Les composants graphiques

- Button
- Canvas (zone de dessin)
- Checkbox(case à cocher)
- CheckboxGroup
- Label
- Choice (Sélecteur)
- List
- Scrollbar (barre de défilement)
- TextField (zone de saisie d'une ligne)
- TextArea (zone de saisie multilignes)



AWT : Conteneurs et Composants

- On ajoute un composant dans un conteneur, avec la méthode **add()** :

```
Panel p = new Panel();
Button b = new Button();
p.add(b);
```

- De manière similaire, un composant est retirer de son conteneur par la méthode **remove()** :

```
p.remove(b);
```

- Un composant a :
  - une taille préférée que l'on obtient avec **getPreferredSize()**
  - une taille minimum que l'on obtient avec **getMinimunSize()**
  - une taille maximum que l'on obtient avec **getMaximunSize()**

AWT : Conteneurs et Composants

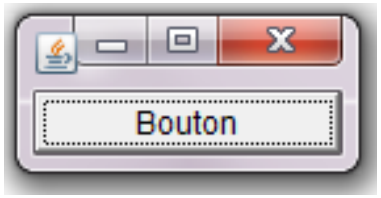
Exemple :

```
import java.awt.*;
public class FirstExample {
public static void main(String[] args){
//Création d'une fenêtre (un objet de la classe Frame) avec un titre
Frame f = new Frame("Ma première fenêtre");

//Création du bouton ayant pour label « Bouton »
Button b = new Button("Bouton");

//Ajout du bouton dans la fenêtre
f.add(b);

// On demande à la fenêtre de choisir la taille minimum avec pack()
//et de se rendre visible avec show()
    f.pack();
    f.show();
  }
}
```



SWING

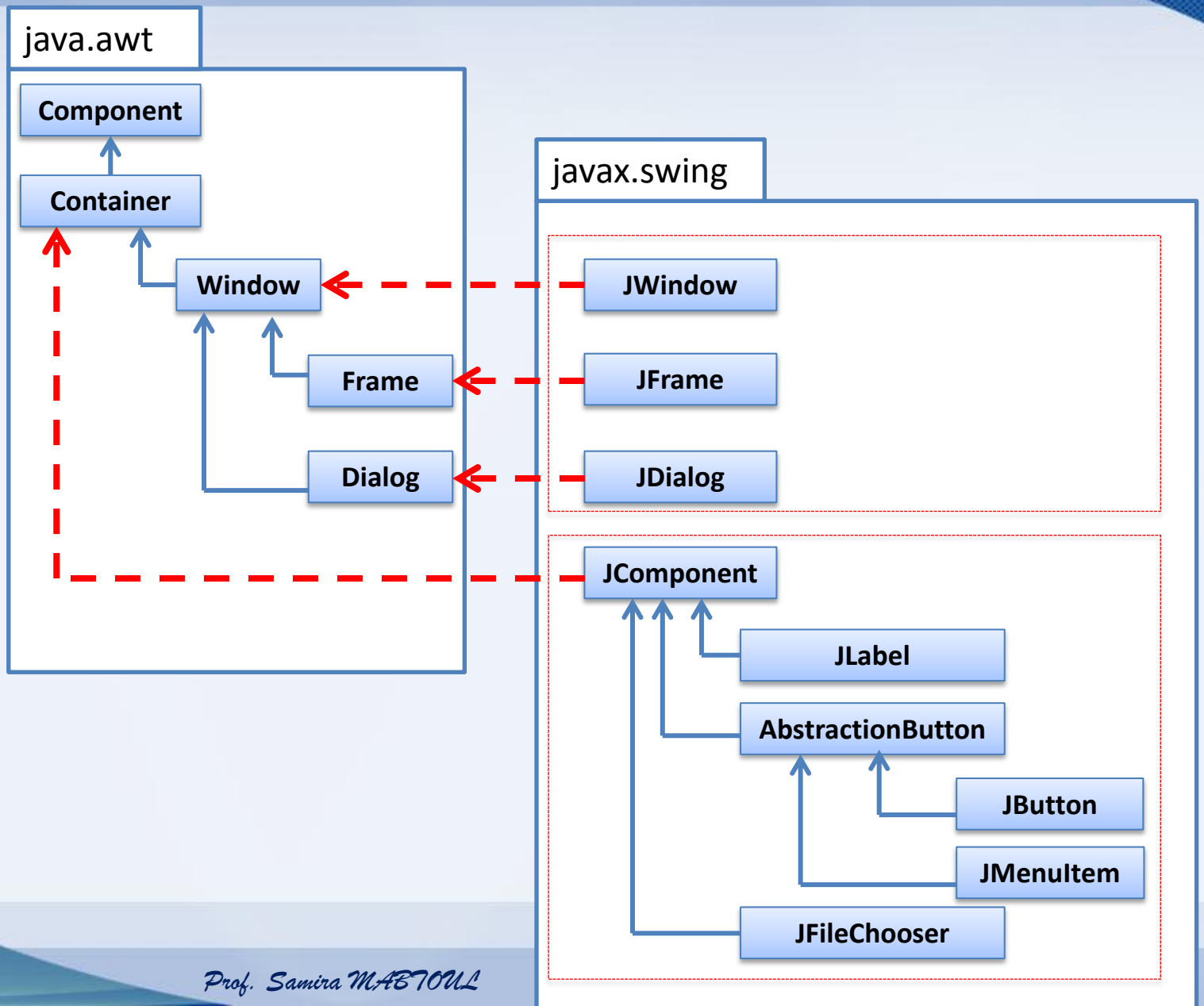
- A cause des problèmes de portabilité, le développement de l'AWT a été interrompu. Il a été étendu par le JFC (Java Foundation Class)
- Les éléments graphiques ont été regroupés dans le package Swing
- La bibliothèque Swing est une nouvelle bibliothèque de composants graphiques pour Java
  - Swing est intégré à Java 1.2
  - Swing peut être téléchargé séparément pour une utilisation avec des versions de Java antérieures (1.1.5+)
- Les anciens navigateurs ne connaissent pas les Swing
- Les AWT sont plus rapides et plus faciles à utiliser
- Un Composant léger de Swing (en anglais, *lightweight GUI component*) est un composant graphique indépendant du gestionnaire de fenêtre local

SWING : Classes

- La plupart des classes du composant **Swing** commencent par la lettre **J** : **JButton**, **JFrame**, etc. ce sont des classes comme **Button** et **Frame** (de composants AWT)

- En réalité **JFrame** hérite de **Frame** alors que les autres composants comme **JButton**, **JLabel**, **JTextField**, **JMenu**, etc. héritent tous de la classe de base **JComponent** qui fait parti de Swing, et qui hérite elle-même indirectement de **Component** qui est un élément de AWT

SWING : Classes



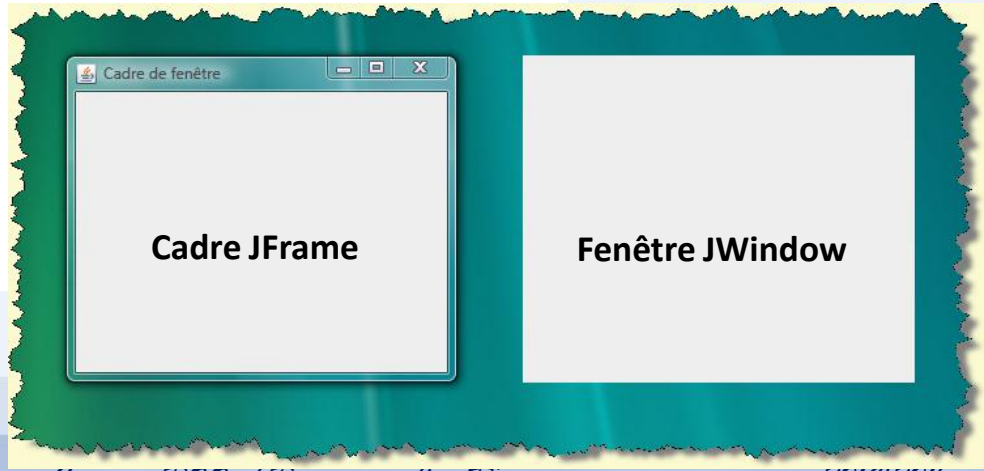
Fenêtre et cadre de fenêtre : JWindow et JFrame

- ***JFrame :*** fenêtre avec barre de titre et bordure
- ***JWindow :*** fenêtre sans décoration, sans titre et bordure

```
public class Fenêtre {
    public static void main(String[] args) {
        JFrame cadre = new JFrame("Cadre de fenêtre");
        cadre.setSize(300, 250);
        cadre.setLocation(150, 100);

        JWindow fenêtre = new JWindow();
        fenêtre.setSize(300, 250);
        fenêtre.setLocation(500, 100);

        cadre.setVisible(true);
        fenêtre.setVisible(true);
    }
}
```



Fenêtre et cadre de fenêtre : JFrame et JWindow

Décrivons le fonctionnement de ce petit programme :

- Le constructeur de JFrame peut prendre un argument String indiquant le titre affiché dans la barre de titre. Il est également possible de créer un JFrame sans titre, puis d'appeler ensuite la méthode **setTitle()** pour en fournir un ultérieurement
- La taille et la position de JFrame et de JWindow sur le bureau sont déterminées par les méthodes **setSize()** et **setLocation()**
- JWindow ne possédant pas de barre de titre, le constructeur de JWindow ne comporte donc pas d'arguments
- Pour afficher une fenêtre on utilise la méthode **setVisible()** (ou **show()**)
- On peut manipuler une fenêtre comme n'importe quelle fenêtre graphique d'un logiciel du commerce, et en particulier : la retailler, la déplacer, la réduire à une icône…

Fenêtre et cadre de fenêtre : JFrame

Actions sur les caractéristiques d'une fenêtre :

```
void setBounds (int x, int y, int largeur, int hauteur)
```

Permet de fixer la position de la fenêtre à l'écran, le coin supérieur gauche de la fenêtre est placé au pixel de coordonnées (x, y) et ses dimensions seront de (largeur*hauteur pixel)

```
void setBackground (Color c)
```

Permet de modifier la couleur du fond d'une fenêtre

```
Dimension getSize()
```

Retourne la taille courante d'une fenêtre

```
void setDefaultCloseOperation(int operation)
```

Pour déterminer le comportement de la fenêtre lors de sa fermeture

Cadre de fenêtre : JFrame

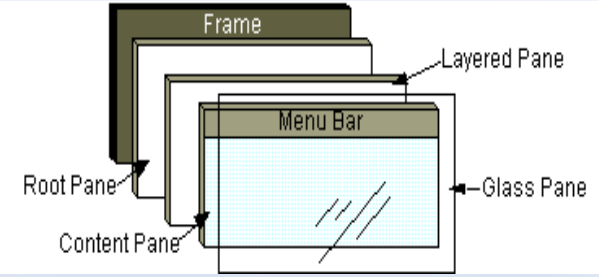
La structure de JFrame possède quatre couches superposées :

JRootPane : possède un gestionnaire de disposition spécifique qui organise automatiquement la disposition des éléments graphiques placés sur la fenêtre

JLayeredPane : ce composant fournit les fonctionnalités d'empilement que requiert Swing pour implémenter les dialogues modaux, les palettes flottantes, les menus contextuels, les bulles d'aide et les effets graphiques de glisser-déplacer

ContentPane : est la partie principale de la fenêtre. C'est sur cette couche que nous pouvons placer tous les différents composants graphiques nécessaire pour application

GlassPane : le panneau de verre (ou la vitre) doit être soit caché soit transparent ; faute de quoi, il obscurcit tous les autres composants du JRootPane. Il peut être utilisé pour intercepter les événements souris destinés aux autres composants du JRootPane, et pour l'affichage de graphismes temporaires au dessus de ces composants



Cadre de fenêtre : JFrame

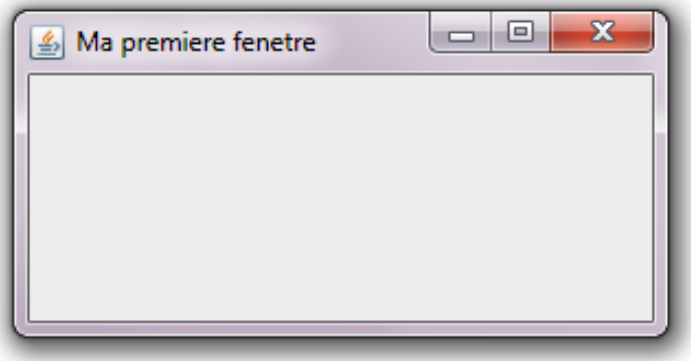
- Une autre manière plus pratique : Hériter de la classe JFrame

```
import javax.swing.* ;

class MaFenetre extends JFrame
{ public MaFenetre () // constructeur
        { setTitle ("Ma premiere fenetre") ;
          setSize (300, 150) ;
          setDefaultCloseOperation(EXIT_ON_CLOSE);
          setVisible (true) ;
        }
}

public class FirstExample {
public static void main (String args[])
{       JFrame fen = new MaFenetre() ;

}
}
```

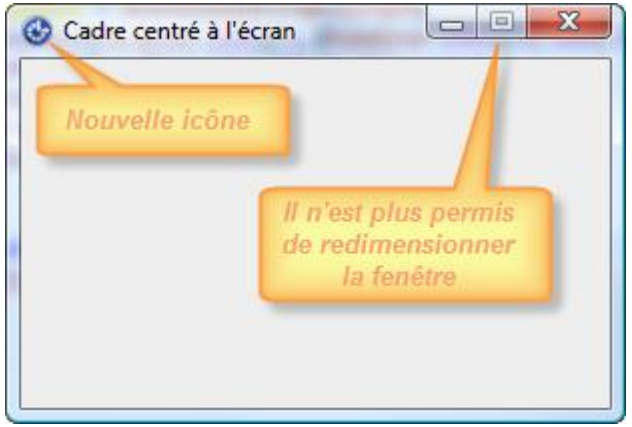


Cadre de fenêtre : JFrame

- Exemple : pour donner une icône à notre cadre

```
public class MaFenetre extends JFrame {
    public MaFenetre() {
     setTitle("Cadre centré à l'écran");
     setSize(300, 200);
     setVisible(true);
//pour placer la fenêtre au centre du bureau
     setLocationRelativeTo(null);
//pour terminer l'application
     setDefaultCloseOperation(EXIT_ON_CLOSE);
//pour donner une icône au cadre
     setIconImage(new ImageIcon("icone.jpg").getImage());
//pour ne pas redimensionner la fenêtre
     setResizable(false);
}

    public static void main(String[] args) {
       new MaFenetre()  ;
    }
}
```

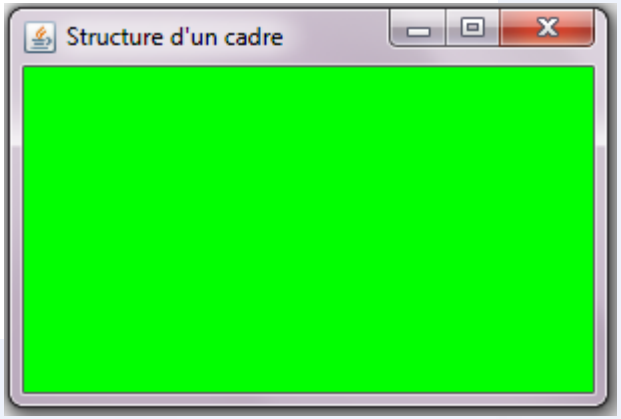


Cadre de fenêtre : JFrame

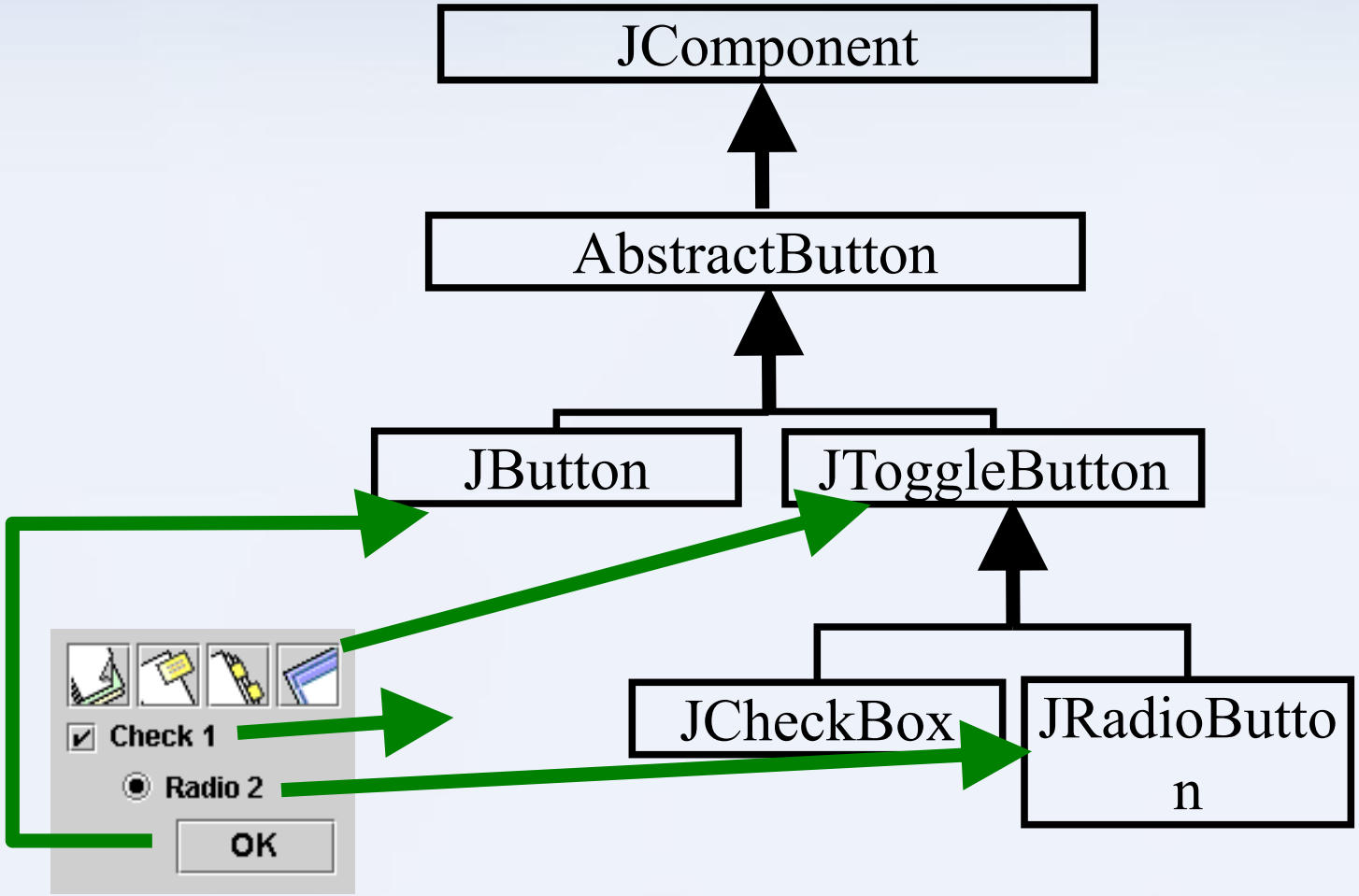
- Exemple : pour donner une couleur du fond à notre zone de contenu

```
public class MaFenetre extends JFrame {
   public MaFenetre() {
      setTitle("Structure d'un cadre");
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setVisible(true);
//pour fixer la taille et la position de la fenêtre
      setBounds(100, 100, 300, 200);
//pour donner une couleur
      getContentPane().setBackground(Color.GREEN);
}

   public static void main(String[] args) {
      new MaFenetre()  ;
   }
}
```



Les composants de sélection ou de choix



Il existe une hiérarchie dans les boutons, de manière à regrouper tout le code commun dans la classe AbstractButton et augmenter l'abstraction.

JButton

Création d'un bouton et ajout dans la fenêtre

- Création d'un bouton pourtant l'étiquette "HELLO"

```
JButton monBouton = new JButton("HELLO");
```

- Pour mettre les composants dans la partie contenu de Jframe. La méthode getContentPane() de la classe **JFrame** fournie la référence à ce contenu de type Container

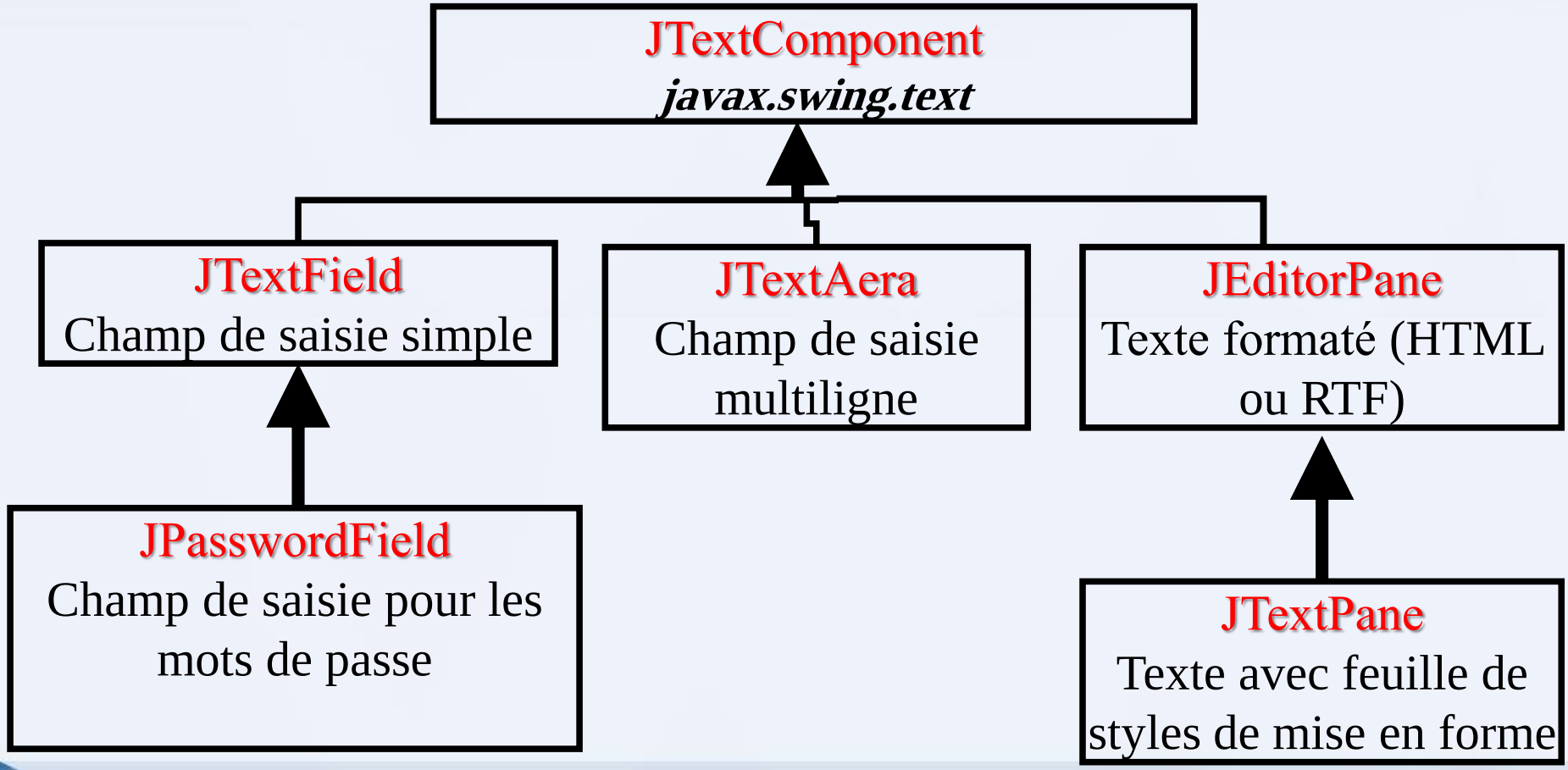
```
Container c = getContentPane();
c.add(monBouton);
```

Ou bien

```
getContentPane().add(monBouton)
```

Les composants textuels

La classe abstraite JTextComponent propose des méthodes générales pour la gestion du cut/copy/paste, polices, ...



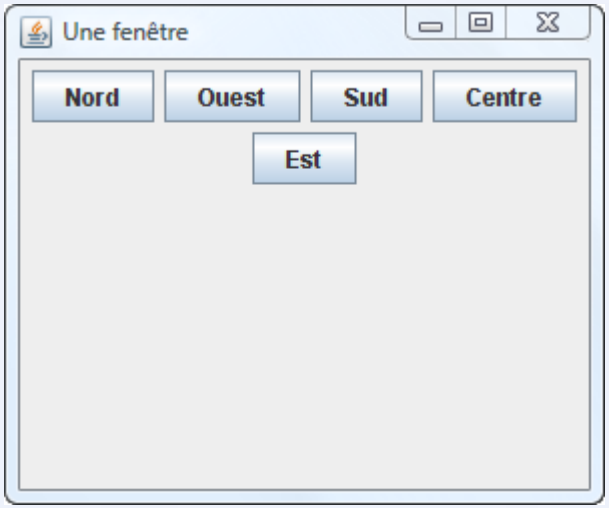
Gestionnaire de disposition

- **Swing** utilise des gestionnaires de présentation pour disposer des composants à l'intérieur de conteneurs en contrôlant leur taille et leur position (**layout manager**)
- Le gestionnaire de présentation gère le positionnement et le (re)dimensionnement des composants d'un conteneur
- Tout conteneur possède un gestionnaire de présentation par défaut :
  - Toute instance de ***Container*** référence une instance de ***LayoutManager***
  - Il est possible d'en changer grâce à la méthode **setLayout()**
- **Swing** possède plusieurs gestionnaires de présentation qui implémentent des modèles de disposition : **FlowLayout, GridLayout, BorderLayout, CardLayout, BoxLayout, SpringLayout …**

Gestionnaire de disposition : FlowLayout

- Dispose les composants d'un container les uns derrière les autres en ligne et à leur taille préférée (*de gauche à droite)*, en retournant à la ligne si le container n'est pas assez large(*de haut en bas*)
- C'est le gestionnaire de placement par défaut d'un **JPanel**

```
public class Fenêtre extends JFrame {
   private JButton nord = new JButton("Nord");
   private JButton ouest = new JButton("Ouest");
   private JButton sud = new JButton("Sud");
   private JButton centre = new JButton("Centre");
   private JButton est = new JButton("Est");
   private JPanel panneau = new JPanel();
   public Fenêtre() {
      setTitle("Une fenêtre");
      setSize(300, 250);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      panneau.add(nord);
      panneau.add(ouest);
      panneau.add(sud);
      panneau.add(centre);
      panneau.add(est);
      add(panneau);
      setVisible(true);
   }
   public static void main(String[] args) {
      new Fenêtre();
   }
}
```

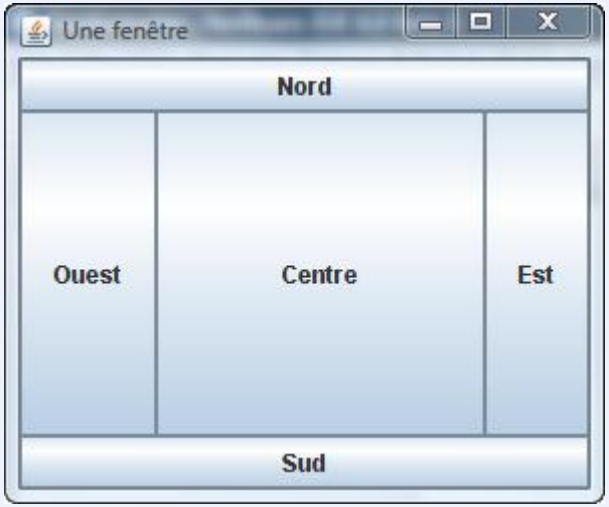


Gestionnaire de disposition : BorderLayout

- Dispose cinq composants maximum dans un container, deux au bords supérieur et inférieur à leur hauteur préférée, deux au bords gauche et droit à leur largeur préférée, et un au centre qui occupe le reste de l'espace
- C'est le gestionnaire de placement par défaut d'un **JFrame**

```
public class Fenêtre extends JFrame {
   private JButton nord = new JButton("Nord");
   private JButton ouest = new JButton("Ouest");
   private JButton sud = new JButton("Sud");
   private JButton centre = new JButton("Centre");
   private JButton est = new JButton("Est");

   public Fenêtre() {
      setTitle("Une fenêtre");
      setSize(300, 250);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      add(nord, BorderLayout.NORTH);
      add(ouest, BorderLayout.WEST);
      add(sud, BorderLayout.SOUTH);
      add(centre, BorderLayout.CENTER);
      add(est, BorderLayout.EAST);
      setVisible(true);
   }
   public static void main(String[] args) {
      new Fenêtre();
   }
}
```



Gestionnaire de disposition : GridLayout

- Dispose les composants d'un container dans une grille dont toutes les cellules ont les mêmes dimensions

```
public class Fenêtre extends JFrame {
   private JButton nord = new JButton("Nord");
   private JButton ouest = new JButton("Ouest");
   private JButton sud = new JButton("Sud");
   private JButton centre = new JButton("Centre");
   private JButton est = new JButton("Est");

   public Fenêtre() {
      setTitle("Une fenêtre");
      setSize(300, 250);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setLayout(new GridLayout(3, 2));
      add(nord);
      add(ouest);
      add(sud);
      add(centre);
      add(est);
      setVisible(true);
   }
   public static void main(String[] args) {
      new Fenêtre();
   }
}
```

Gestionnaire de disposition : autres

Classe	Description
CardLayout	Affiche un composant à la fois parmi l'ensemble des composants d'un container (pratique pour créer des panneaux comme ceux de la boite de dialogue de Preference d'Eclipse)
BoxLayout	Dispose les composants en ligne à leur hauteur préférée ou en colonne à leur largeur préférée
GridBagLayout	Dispose les composants d'un container dans une grille dont les cellules peuvent avoir des dimensions variables. La position et les dimensions de la cellule d'un composant varient en fonction de sa taille préférée et des contraintes de classe GridBagConstraints qui lui sont associées
SpringLayout	Dispose les composants d'un container en fonction de leur taille préférée et de contraintes qui spécifient commet ces composants sont rattachés les uns par rapport aux autres

Les événements graphiques

- L'utilisateur effectue
  - Une action au niveau de l'interface utilisateur (clic souris, sélection d'un item, etc) ➔ Alors un **événement graphique** est émis

- Lorsqu'un événement se produit
  - Il est reçu par le composant avec lequel l'utilisateur interagit (par exemple un bouton, un curseur, un champ de texte, etc.)
  - Ce composant transmet cet événement à un autre objet, un **écouteur** qui possède une méthode pour traiter l'événement
  - Cette méthode reçoit l'objet événement généré de façon à traiter l'interaction de l'utilisateur

Les événements graphiques

- La gestion des événements passe par l'utilisation d'objets "écouteurs d'événements" (les Listener) et d'objets "sources d'événements "
  - ***Un objet écouteur*** est l'instance d'une classe implémentant l'interface EventListener (ou une interface “fille”)
  - ***Une source*** d'événements est un objet pouvant recenser des objets écouteurs et leur envoyer des objets événements
- Lorsqu'un événement se produit,
  - la source d'événements envoie un objet événement correspondant à tous ses écouteurs
  - Les objets écouteurs utilisent alors l'information contenue dans l'objet événement pour déterminer leur réponse

Les événements graphiques

Exemple:

```
import java.awt.event.*;
import javax.swing.*;
public class MonAction implements ActionListener {
//Méthode de l'interface ActionListner
public void actionPerformed (ActionEvent e)  {
System.out.println ("Une action a eu lieu") ;}
}

public class TestBouton {
public TestBouton(){
JFrame f = new JFrame ("Test Bouton");
JButton b = new JButton ("Cliquer ici");
f.add (b) ;
f.pack (); f.setVisible (true) ;
b.addActionListener (new MonAction ());
}

  public static void main(String args[]) {
    TestBouton test = new TestBouton();}
}
```

Les événements graphiques

- Les écouteurs sont des interfaces

- Donc, une même classe peut implémenter plusieurs interfaces écouteur
  - Par exemple une classe héritant de JFrame implémentera les interfaces **MouseMotionListener** (pour les déplacements souris) et **MouseListener** (pour les clics souris)

- Chaque composant graphique est conçu pour être la source d'un ou plusieurs types d'événements particuliers

- À une catégorie donnée **Xxx**, on associera toujours un objet écouteur des événements d'une classe implémente l'interface **XxxListener** par l'intermédière d'une méthode nommée **addXxxListener**

Catégories d'événements graphiques

- Plusieurs types d'événements sont définis dans le package **java.awt.event**

- Pour chaque catégorie d'événements, il existe une interface qui doit être définie par toute classe souhaitant recevoir cette catégorie d'événements

- Cette interface exige aussi qu'une ou plusieurs méthodes soient définies

- Ces méthodes sont appelées lorsque des événements particuliers surviennent

Catégories d'événements graphiques

- ActionListener : Action (clic) sur un bouton, retour chariot dans une zone de texte, « tic d'horloge » (Objet Timer)
- WindowListener : Fermeture, iconisation, etc. des fenêtres
- TextListener : Changement de valeur dans une zone de texte
- ItemListener : Sélection d'un item dans une liste
- MouseListener : Clic, enfoncement/relâchement des boutons de la souris, etc.
- MouseMotionListener : Déplacement de la souris, drag&drop avec la souris, etc.
- AdjustmentListener : Déplacement d'une échelle
- ComponentListener : Savoir si un composant a été caché, affiché …
- ContainerListener : Ajout d'un composant dans un Container
- FocusListener : Pour savoir si un élément a le "focus"
- KeyListener : Pour la gestion des événements clavier

Catégories d'événements graphiques

Catégorie	Nom de l'interface	Méthodes
Action	ActionListener	actionPerformed (ActionEvent)
Item	ItemListener	itemStateChanged (ItemEvent)
Mouse	MouseMotionListener	mouseDragged (MouseEvent) mouseMoved (MouseEvent)
Mouse	MouseListener	mousePressed (MouseEvent) mouseReleased (MouseEvent) mouseEntered (MouseEvent) (MouseEvent) mouseExited mouseClicked
Key	KeyListener	keyPressed (KeyEvent) keyReleased (KeyEvent) keyTyped (KeyEvent)
Focus	FocusListener	focusGained (FocusEvent) focusLost (FocusEvent)
……..	……..	……..

Catégories d'événements graphiques

Exemple : changement du titre de la fenêtre lors du clique sur le bouton dans l ’interface

```
// Implémentation de l'interface ActionListener
public class EssaiActionEvent1 extends JFrame  implements ActionListener {

  public static void main(String args[])
  {   EssaiActionEvent1 f= new EssaiActionEvent1();   }

  public EssaiActionEvent1()
  {
   super("Utilisation d’un ActionEvent");
   Button b = new Button("action");
   // On enregistre l’écouteur d’evt action auprès de l’objet source "b"
   b.addActionListener(this);
   add(BorderLayout.CENTER,b);

   setDefaultCloseOperation(EXIT_ON_CLOSE);
   setVisible(true);
   setBounds(100, 100, 300, 200);
  }

//méthode de l'interface ActionListener
  public void actionPerformed( ActionEvent e )
  {//Lorsque l'on clique sur le bouton dans l ’interface, le titre de la fenêtre change
   setTitle("bouton cliqué !");
 } }
```

Catégories d'événements graphiques

Exemple : changement du titre de la fenêtre lors du clique sur les boutons dans l ’interface

```
public class EssaiActionEvent2 extends JFrame implements ActionListener {
private JButton b1,b2;

public static void main(String args[])
    {EssaiActionEvent2 f= new EssaiActionEvent2();}

public EssaiActionEvent2(){
super("Utilisation d’un ActionEvent");
b1 = new JButton("action1");
b2 = new JButton("action2");
//Les 2 boutons ont le même écouteur (la fenêtre)
b1.addActionListener(this);
b2.addActionListener(this);
add(BorderLayout.CENTER,b1);
add(BorderLayout.SOUTH,b2);
pack();show(); }

public void actionPerformed( ActionEvent e ) {
//e.getSource()" renvoie l'objet source de l’événement.  On effectue un test sur les
boutons (on compare les références)
if (e.getSource() == b1) setTitle("action1 cliqué");
if (e.getSource() == b2) setTitle("action2 cliqué");
}}
```