Université Cadi Ayyad
Ecole Supérieure de Technologie – Safi

COURS 5

COURS DE LA PROGRAMMATION ORIENTÉE OBJET (JAVA)

Pr. EL ABDELLAOUI SAID

Elabdellaoui.said@yahoo.fr

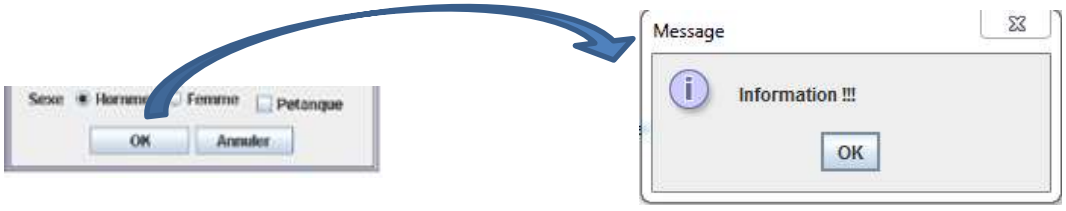
INTERFACES GRAPHIQUES

GÉNÉRALITÉS

GÉNÉRALITÉS

- **Partie I :** Construire fenêtre graphique : Objets graphiques, Affichage….

- **Partie II :** Programmation par événement : Comment faire pour que le programme réagisse?

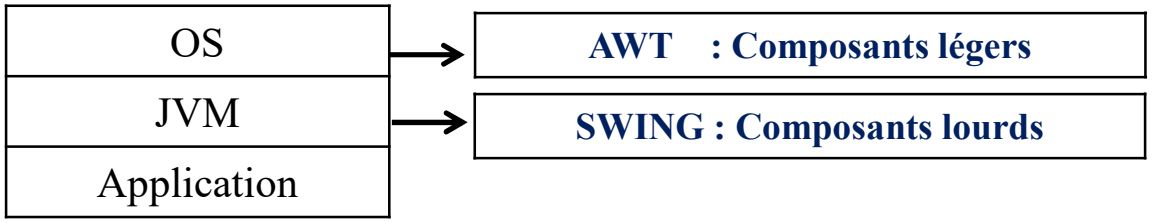


- **Partie III :** Connexion avec la base de donnée

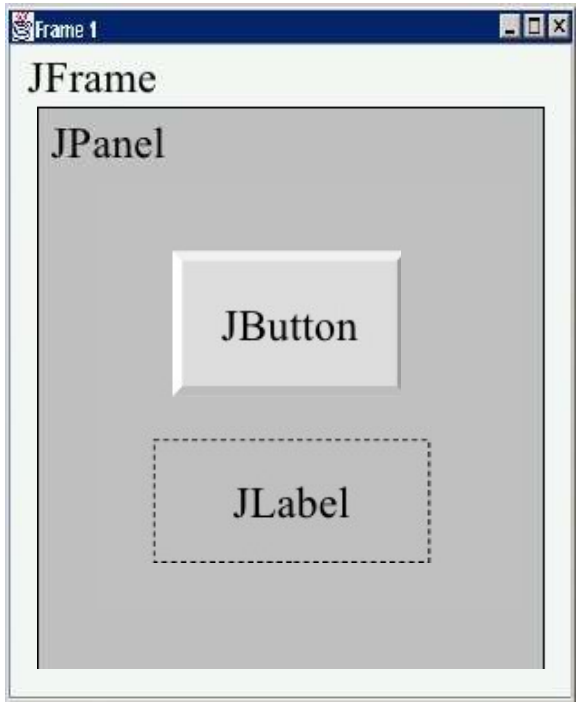
Partie I : Interfaces graphiques

GÉNÉRALITÉS

- Java propose des boîtes à outils graphiques :
  - **AWT** (Abstract Window ToolKit) : qui contient des composants qui font appel aux composants graphiques de l'OS.
  - **SWING** : qui contient des composants écrit complètement avec java et son indépendant de l'OS.
  - **JAVAFx** : est un Framework Java permettant de construire des RIA (Rich Internet Application) et des interfaces graphiques



HIÉRARCHIE GRAPHIQUE



Définir des fenêtres :
JFrame, Jwindow, JApplet, JDialog

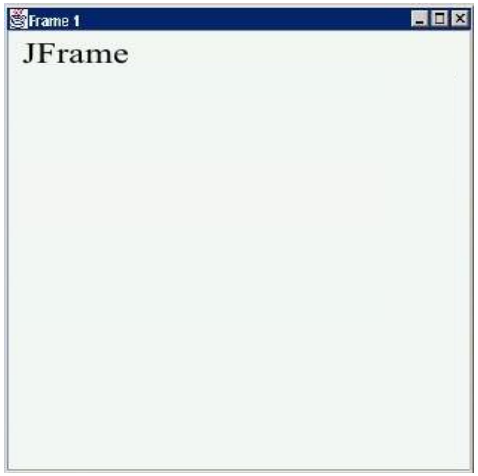
Composer la fenêtre :
JPanel, JScrollPane, JSplitPane, …

Eléments de base :
JButton, JCheckBox, JTextField, JTextArea, …

3 niveaux

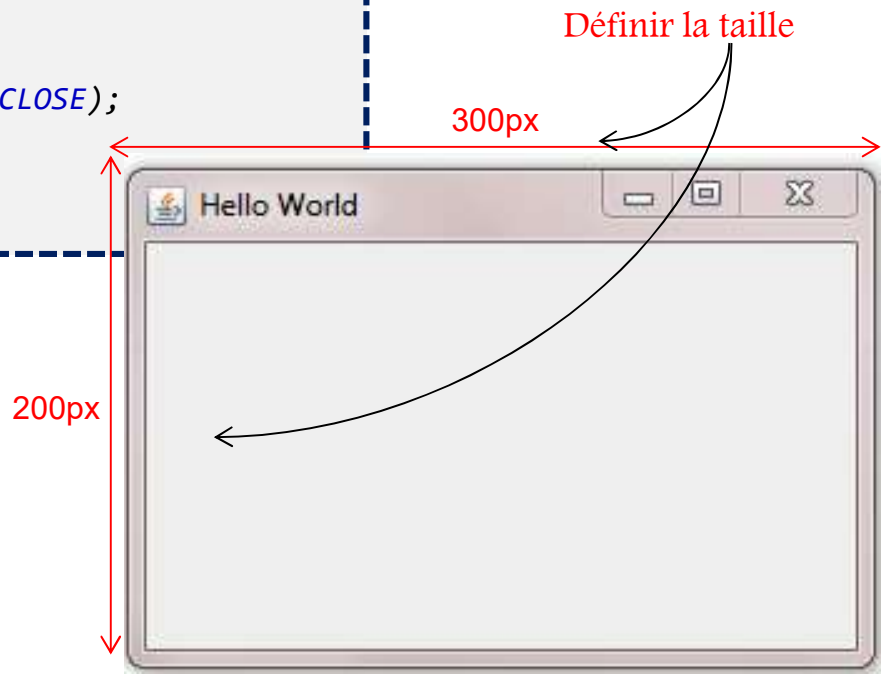
Niveau I :

FENÊTRES ET CADRES



NOTRE 1ERE INTERFACE

```
import javax.swing.*;
public class HelloWorld {
public static void main (String[] args) {
      JFrame f = new JFrame("Hello World");
      // f.setTitle("Hello World");
      //Composants ajoutés au frame courant.
      f.setSize(300,200);
      f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      f.setVisible(true);
}             }
```



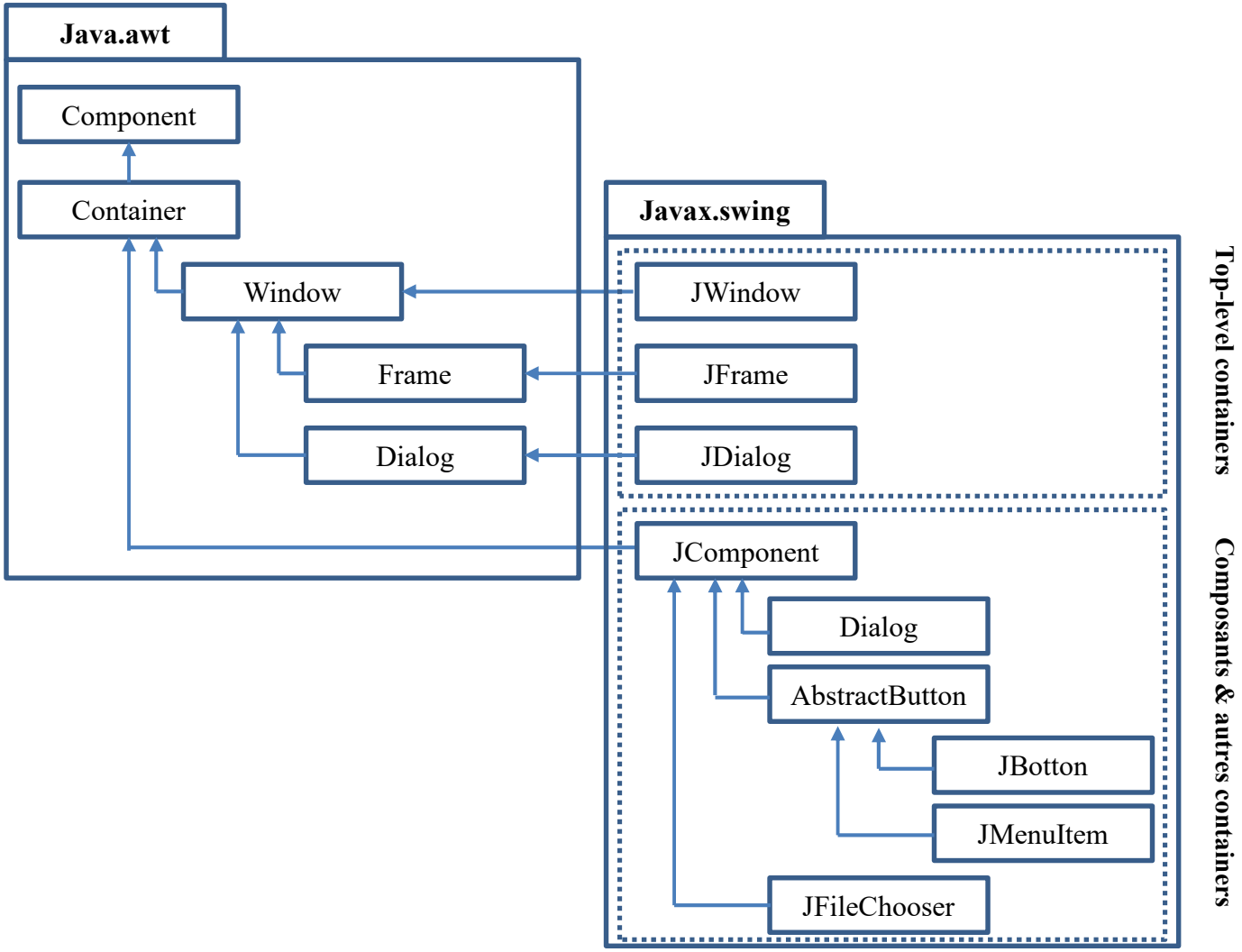
D'AUTRE MANIER PLUS PRATIQUE - HÉRIAGE DU CLASS JFRAME -

```
import javax.swing.*;
public class HelloWorld extends JFrame{
 helloWord(){
      setTitle ("Hello World"); // ou this.setTitle ("Hello World");
      setSize(300,200);
      setDefaultCloseOperation(EXIT_ON_CLOSE);
      setVisible(true);
}

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

MÉTHODES SWING

- **setLocation** (*int,int)* : Modifier la position de la fenêtre.
- **setAlwaysOnTop** (*boolean*) : Toujours au premier plan.
- **setResizable** (*boolean*) : Permettre ou interdire le redimensionnement de la fenêtre.
- **setTitle** (*String*) : Modifier le titre.
- **setVisible** (*boolean*) : La visibilité de la fenêtre.
- **setDefaultCloseOperation** *(f.HIDE_ON_CLOSE) :* cacher la fenêtre
  - *(f.DISPOSE_ON_CLOSE);* détruire l'objet fenêtre
  - *(f.DO_NOTHING_ON_CLOSE);* ne rien faire
  - *(f.EXIT_ON_CLOSE);* terminer l'application
- **setSize**() : Dimensions de la fenêtre.
- **setBackground**(*Color.yellow*) : Modifier la couleur d'arrière-plan.
- **setBounds** (*x, y, larg, long*) la position et la taille de la composante.
- **setIconImage**(*Toolkit.getDefaultToolkit().getImage(icône.png))* icône de l'application.

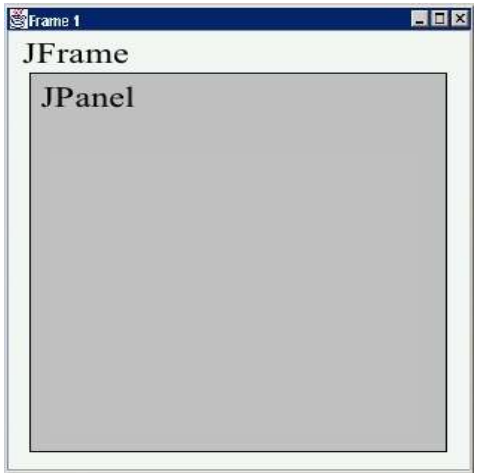
LES COMPOSANTS DU PAQUETAGE JAVA.AWT



- ***JFrame*** hérite de ***Frame***
- ***JButton*** , ***JLabel***, ***JMenu*** ***JTextField***, etc. héritent tous de la classe de base ***JComponent*** qui fait parti de swing.
- ***JComponent*** hérite indirectement de ***Component*** qui est un élément de ***AWT***.

Niveau II :

LES CONTENEURS

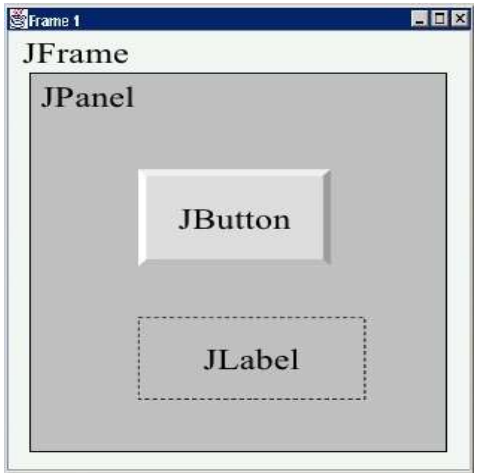


- Un **JPanel** est un composant de type conteneur.
- Il accueille d'autres objets de même type ou des objets de type composant (boutons, cases cocher , etc)


```
import javax.awt.*;
import javax.swing.*;
public class Exemple extends JFrame {
public Exemple(){
      setTitle("Mon cadre");
      setSize(400,100);
      JPanel pan= new JPanel(); //Instanciation d'un objet JPanel
      pan.setBackground(Color.ORANGE);//definition de la couleur de fond.
      add(pan); //equivalent à this.setContentPane(pan);
      setVisible(true);
                                        }
public static void main (String[] args) {  new Exemple();  }
}
```

Niveau III :

LES COMPOSANTES DE BASES



JLABEL

❑ Avec un *JLabel*, on peut créer un texte "étiquette" et/ou une image. On peut également intègre du HTML, spécifier la position du texte par rapport à l'image ou spécifier l'emplacement par rapport à son conteneur.

❑ Quelques méthodes publiques de cette classe :

- ***JLabel(string, Icon, Int)*** // Création d'un Jlabel
- ***Void setText(string)*** // Modifie le texte du Jlabel
- ***String getText()*** // Retourne le texte du Jlabel
- ***Void seticon(Icon)*** //Spécifier la position du texte par rapport à l'icône
  - ✓ *Void setHorizontalTextPosition(int)*
  - ✓ *Void setVerticalTextPosition(int)*
- ***Void setToolTipText(string)*** //Associe un info bulles

❑ Étiquette pouvant contenir du texte et image :

```
JLabel  jl = new JLabel ("Label 1");
    ▪ System.out.println(jl.getText());
    ▪ jl.setIcon(new ImageIcon("java.gif"));
    ▪ jl.setVerticalTextPosition(SwingConstants.BOTTOM)
    ▪ jl.setHorizontalTextPosition(SwingConstants.CENTER);
```

JLABEL

- Exemple :

```
public class LabelPanel extends JFrame{
 JLabel testLabel;
     public LabelPanel ()       {
       testLabel = new JLabel (" Icon Big Label") ;
       testLabel.setToolTipText (" a Label with Icone")  ;
       // Créer une nouvelle fonte
       Font serif32Font = new Font ("Serif" , Font.BOLD,32) ;
       // Donner une fonte au contenu du Label
       testLabel.setFont (serif32Font) ;
       // Créer une icône
       Icon soundIcon  =  new ImageIcon ("images :img.gif ") ;
           }
 }
```

JBUTTON

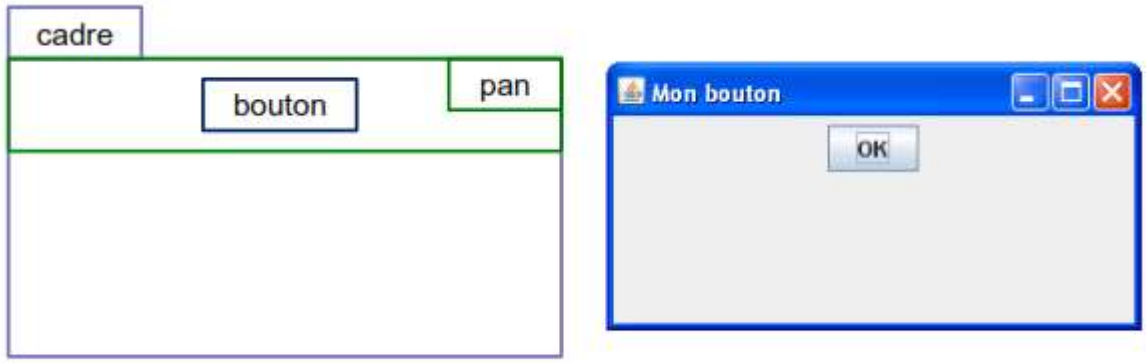
- Issue du package *javax.swing*
- Création :

```
JButton jb= new JButton("OK"); //ou
JButton jb = new JButton("OK", new ImageIcon("icon.gif"));
```

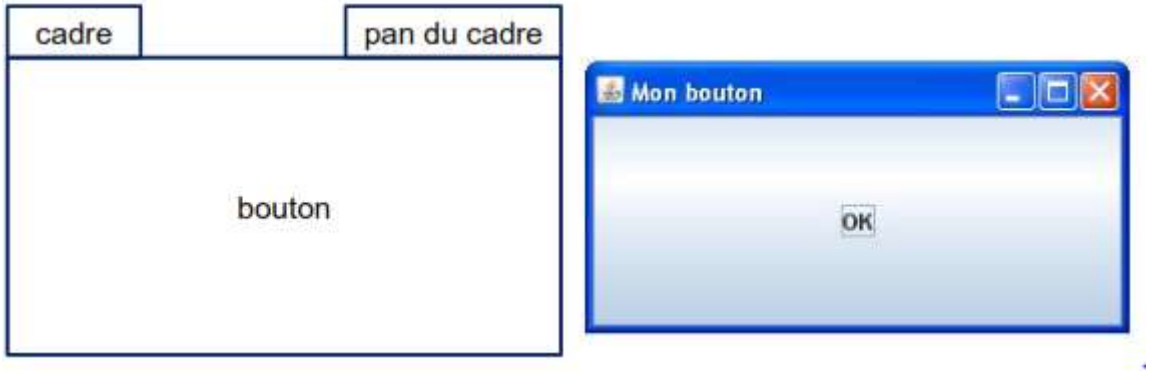
- `jb.setRolloverIcon(new ImageIcon("icon.gif")); // affichage d'image lors du survole`
- `jb.setPressedIcon(new ImageIcon("icon.gif")); // affichage d'image lors du pressage`
- `jb.setDisabledIcon(new ImageIcon("icon.gif"));`
- `jb.setIcon(new ImageIcon ("icon.gif"));`
- `jb.setMnemonic('o'); // ALT + o`
- `jb.setBorderPainted(false);`
- `jb.SetBackground(Color.white);`

JBUTTON

- Le bouton est centré sur le conteneur. Par défaut, *Jpanel* gère la mise en page.



- Dans cet exemple, on utilise la pane de *JFrame*. Le bouton occupe l'espace totale de la fenêtre.



LES LAYOUT MANAGERS (1)

- Java utilise les *layout managers* pour gérer la position des éléments sur la fenêtre.
- Tous ces Layout Managers se trouvent dans le package java.awt.
- Pour utiliser *layout manager* on utilise la fonction suivante:
- Syntaxe :

```
void setLayout(LayoutManager obj)
```

- *Swing* possède plusieurs gestionnaires de placement qui implémente des modèles de disposition courants :

Classe	Description
Java.awt.FlowLayout	Dispose les composants d'un container les uns derrière les autres en ligne et à leur taille préférée, en retournant à la ligne si le container n'est pas assez large.
Java.awt.GridLayout	Dispose les composants d'un container dans une grille dont toutes les cellules ont les mêmes dimensions.
Java.awt.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.

❑ Exemple :


```
import javax.awt.*;
import javax.swing.*;
public class Exemple extends JFrame{
  private JPanel pan= new JPanel();
  private JButton bouton = new JButton ("OK");
      public Exemple(){
            setTitle("Mon bouton");
            setSize(300,150);
            pan.add(bouton);// ajout du bouton au pan
            add(pan); // ajout du pan au cadre
            setDefaultCloseOperation(EXIT_ON_CLOSE);
            setLocationRelativeTo(null);
            setVisible(true);
                        }
public static void main (String[] args) {new Exemple();  }
                                    }
```

- *Swing* possède plusieurs gestionnaires de placement qui implémente des modèles de disposition courants :

Classe	Description
Java.awt.CardLayout	Affiche un composants d'un à la fois parmi l'ensemble des composants d'un container (pratique pour créer des panneaux comme ceux de la boite de dialogue de préférences d'Eclipse).
Javax.swing.BoxLayout	Dispose les composants en ligne à leur hauteur préférée ou en colonne à leur largeur préférée.
Java.awt.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 des composant varient en fonction de sa taille préférée et des contraints de classe Java .awt.GridBagConstraints qui lui sont associées.
Javax.swing.SpringLayout	Dispose les composants d'un container en fonction de leur taille préférée et de contraintes qui spécifient comment ces composants sont rattachés les uns par rapport aux autres.

FlowLayout

- C'est le gestionnaire de placement par défaut d'un JPanel
- Il place les objets sous leur taille préférée de gauche à droit et de haut en bas
- On ne passe à la prochaine ligne que quand l'espace restant sur la ligne n'est plus suffisant pour contenir le composant.

- Syntaxe :

```
P.setLayout(new FlowLayout());
P.add(B1); //JButton B1= new JButton("button 1")
```

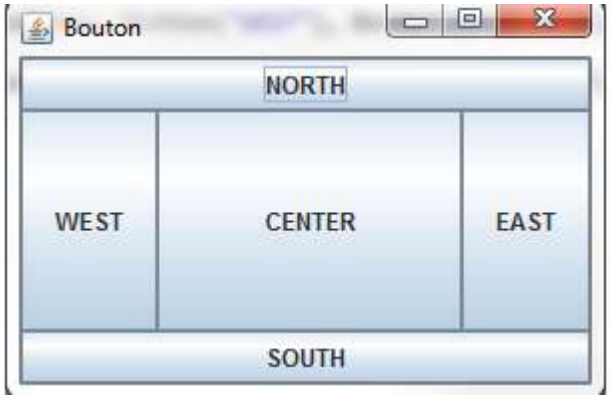
JBUTTON : LES LAYOUT MANAGERS (1')

- Positionnement les composants de base :

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class Fenetre1 extends JFrame{
      JButton b1 = new JButton("bouton N1"); JButton b2 = new JButton("bouton N2");
      JButton b3 = new JButton("bouton N3"); JButton b4 = new JButton("bouton N4");
      JPanal P= new JPane();
public Fenetre1 {
    setTitle("Hello World"); setSize(300,200);
    P.setLayout(new FlowLayout());
    P.add(bouton);
    setDefaultCloseOperation(this.EXIT_ON_CLOSE);
    setLocationRelativeTo(this.getParent());
    setDefaultCloseOperation(3);
                }
                                        }
```

BorderLayout

- Il place les composants dans une région de notre choix (NORTH, WEST, CENTER, EAST, SOUTH).
- Si la région n'est pas spécifiée, la composant est positionné au centre.

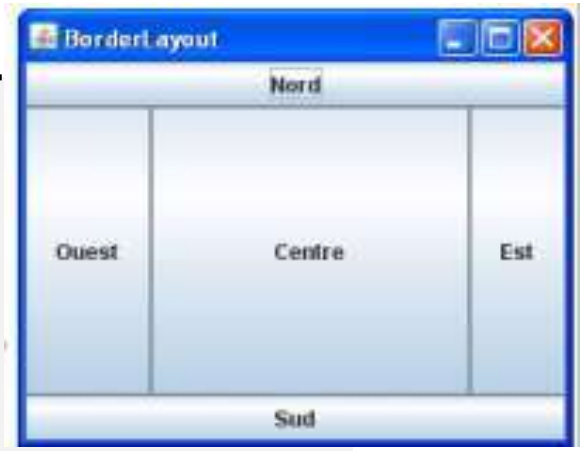


- Syntaxe :

```
P.setLayout(new BorderLayout());
P.add(B1, BorderLayout.CENTER);
```

JBUTTON : LES LAYOUT MANAGERS (1')

- Positionner les composant de base :




```
import javax.swing.*; import java.awt.*; import java.awt.event.*;
public class Exemple extends JFrame{
private JPane P  = new JPane ();
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 Exemple() {
    setTitle("Border Layout");  setSize(300,250);
     setDefaultCloseOperation(EXIT_ON_CLOSE);
     P.setLayout(new BorderLayout());
     P.add(nord, BorderLayout.NORTH); P.add(ouest, BorderLayout.WEST);
     P.add(sud, BorderLayout.SOUTH); P.add(centre, BorderLayout.CENTRE);
     P.add(est, BorderLayout.EAST);
     setVisible(true);           }
public static void main (String[] args) {   new Exemple();  }   }                }
```

JBUTTON : LES LAYOUT MANAGERS (1')

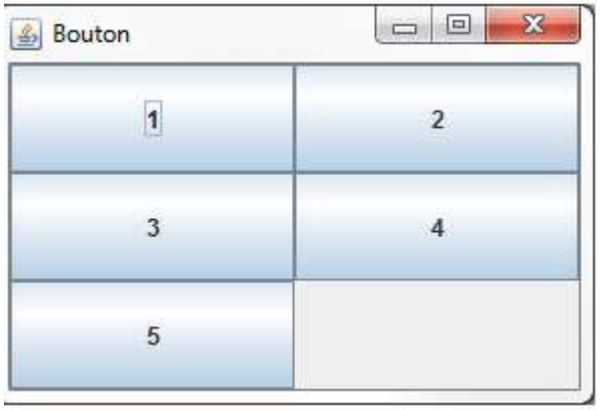
- On utilise dans cet exercice le pane de JFrame :

```
import javax.swing.*;
import java.awt.*;
public class Exemple extends JFrame{
private JButton b1    = new JButton(" Centre ");
public Exemple() {
     setTitle("Border Layout");  setSize(300,250);
      setLocationRelativeto(null); setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      setLayout(new borderLayout()); //la layout  utiliser
      //On ajoute les boutons au content pane de la JFrame
      getContentPane().add(b1, BorderLayout.Centre);
      getContentPane().add(new JButton(" ouest"), BorderLayout.WEST);
      getContentPane().add(new JButton(" sud"), BorderLayout.SOUTH);
      getContentPane().add(new JButton("nord"),BorderLayout.NORTH);
      getContentPane().add(new JButton(" est"), BorderLayout.EAST);
      setVisible(true);          }
public static void main (String[] args) {   new Exemple();   }   }
```

JBUTTON : LES LAYOUT MANAGERS (2)

GridLayout

- Ajoute les composants suivant une grille définie par un nombre de lignes et de colonnes.



- Syntaxe :

```
P.setLayout(new GridLayout(3, 2));
P.add(new Jbutton("button 1"));
```

JBUTTON : LES LAYOUT MANAGERS (1')

- Positionner les composant de base :

```
import javax.swing.*; import java.awt.*;
public class Exemple extends JFrame{
private JButton b1   = new JButton(" Centre ");
private JPane P  = new JPane ();
public Exemple() {
     setTitle("Grid Layout");  setSize(300,250);
     setLocationRelativeto(null);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      P.setLayout(new GridLayout(3,2)); //3lignes et 2 colonnes
      //On ajoute les boutons au content pane de JFrame
      p.add(b1, BorderLayout.Centre);
      p.add(new JButton("1"));//add(new JButton("1"))
      p.add(new JButton("2")); getContentPane().add(new JButton("3"));
      p.add(new JButton("4")); getContentPane().add(new JButton("5"));
      setVisible(true);          }
public static void main (String[] args) {   new Exemple();  }    }
```

JBUTTON : LES LAYOUT MANAGERS (3)

BoxLayout

- Avec, vous pourrez ranger vos composants à la suite soit sur une ligne, soit sur une colonne. Le mieux, c'est encore un exemple de rendu *(voir figure suivante)* avec un code

- Syntaxe :

```
P.setLayout(new BoxLayout(P, BoxLayout.LINE_AXIS));
P.add(new Jbutton("button 1"));
```

JBUTTON : LES LAYOUT MANAGERS (3')

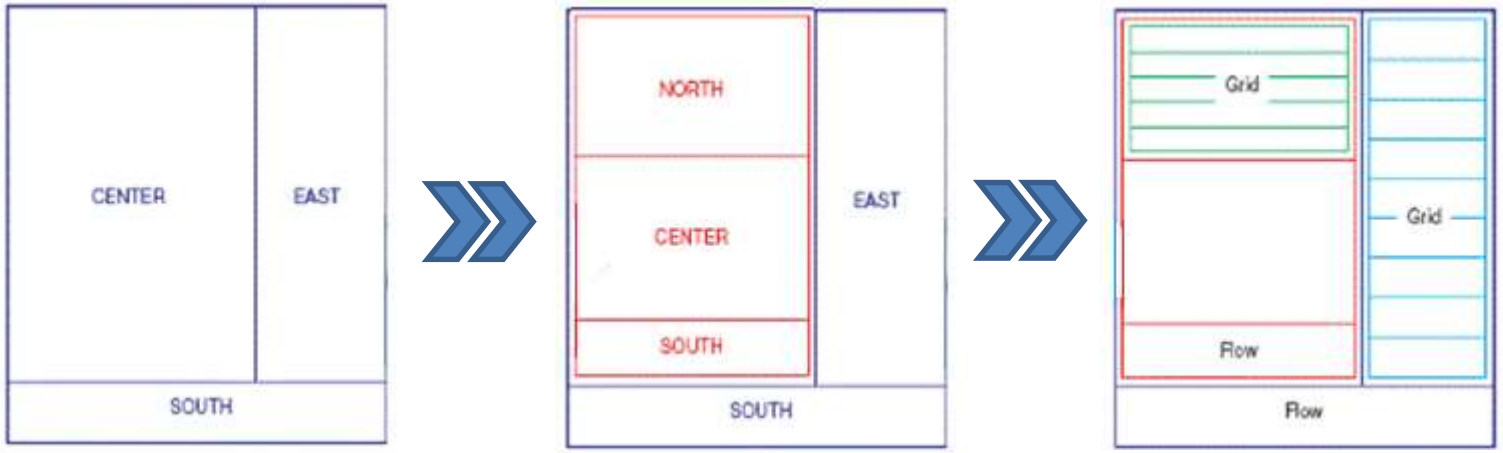

```
Class TEST extends Jframe      {
public TEST()  {
//On définit le layout en lui indiquant qu'il travaillera en ligne
    JPanel P= new JPanel ();
    P.setLayout(new BoxLayout(P, BoxLayout.LINE_AXIS));
    JButton bouton1 = new JButton("bouton 1"); JButton bouton2 = new JButton("bouton 2");
    P.add(bouton1); P.add(bouton2);  //P.add(new JButton("Bouton1"));
//2eme en ligne
    JPanel P1= new JPanel ();
    P1.setLayout(new BoxLayout(P1, BoxLayout.LINE_AXIS));
    JButton bouton3 = new JButton("bouton 3"); JButton bouton4 = new JButton("bouton 4");
    P.add(bouton3);   P.add(bouton4);
//On positionne maintenant ces trois lignes en colonne
    JPanel P3= new JPanel ();
    P3.setLayout(new BoxLayout(P3, BoxLayout.PAGE_AXIS));
    P3.add(P1);P3.add(P)
// Ajout
    add(P3);
```

JBUTTON : LES LAYOUT MANAGERS (3')


```
class TEST extends JFrame{
public TEST()     {
              setTitle("Une fenetre dynamique");
//On définit le layout en lui indiquant qu'il travaillera en ligne
              Box P = Box.createHorizontalBox();
              P.add(new JButton("bouton 1"));
              P.add(new JButton("bouton 2"));
//2eme en ligne
              Box P1 = Box.createHorizontalBox();
              P1.add(new JButton("bouton 3"));
              P1.add(new JButton("bouton 4"));
//On positionne maintenant ces trois lignes en colonne
              Box P3 = Box.createVerticalBox();
               P3.add(P);P3.add(P1);
// Ajout
              add(P3); setSize(500, 300); setLocationRelativeTo(this.getParent());
              setDefaultCloseOperation(3);
                   }
```

EXERCICE

- Pour d'obtenir le schéma suivant on doit combiner l'ensemble de ces gestionnaires



1ERE ÉTAPE

❑ 1ere Etape : Au début, on utilise un pan avec *BorderLayout* avec les bordures Centre EST et Sud comme illustré sur la figure suivante

```
import java.awt.BorderLayout;
import javax.swing.*;
public class sport1 extends JFrame  {
private JButton sud = new JButton ("sud");
private JButton centre = new JButton ("centre");
private JButton est = new JButton ("est");
private JPanel pan1 = new JPanel();
public sport1(){
setTitle("Mélange de Layouts ");
setSize(300,250);
add(pan1);
pan1.setLayout(new BorderLayout());
pan1.add(sud,BorderLayout.SOUTH);
pan1.add(centre,BorderLayout.CENTER);
pan1.add(est,BorderLayout.EAST);
setDefaultCloseOperation(EXIT_ON_CLOSE);
setLocationRelativeTo(null);
 setVisible(true);
       }
public static void main(String[] args) { new
sport1(); } }
```

2EME ÉTAPE

❑ 2éme Etape : On ajoute un autre pan *BorderLayout* avec les bordures Nord, Centre et Sud2

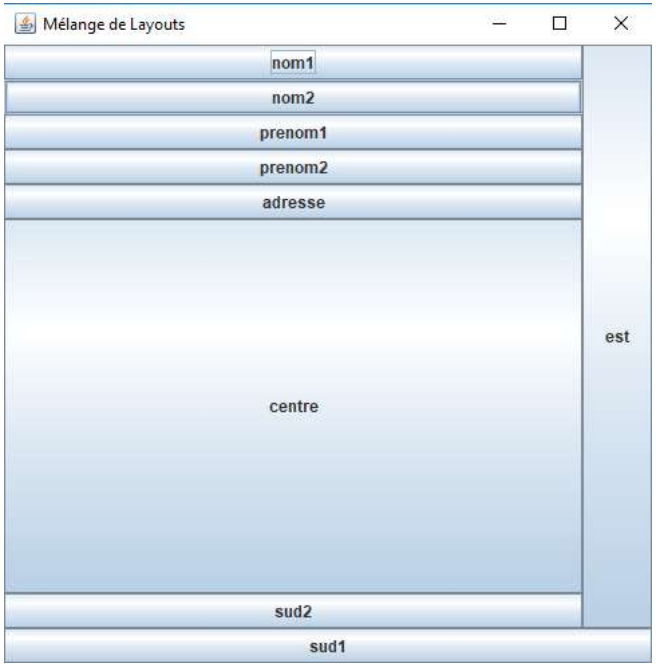
```
import java.awt.BorderLayout;
import javax.swing.*;
public class sport2 extends JFrame {
private JButton sud1 = new JButton ("sud1");
private JButton sud2 = new JButton ("sud2");
//…… bouttons (sud, centre, est) et pan1
private JPanel pan2 = new JPanel();
public sport2(){
// La Premiere division
pan1.setLayout(new BorderLayout());
pan1.add(sud1,BorderLayout.SOUTH);
pan1.add(est,BorderLayout.EAST);
pan1.add(pan2,BorderLayout.CENTER);//ajout de pan
// La Deuxieme division
pan2.setLayout(new BorderLayout());
pan2.add(north,BorderLayout.NORTH);
pan2.add(centre,BorderLayout.CENTER);
pan2.add(sud2,BorderLayout.SOUTH);
add(pan1);
//…… visibility …  }
public static void main(String[] args) {
new sport2(); } }
```

et on le place au centre du premier pan

3EME ÉTAPE

- **3éme Etape :** On ajoute un autre pan GridLayout avec 5 linges et 1 colonne et on le place au nord du 2eme pane

```
//…… Debut
// La Premiere division
//……
// La Deuxieme division
//…
// La Troisieme division

     pan2.add(pan3,BorderLayout.NORTH);
     pan3.setLayout(new GridLayout(5,1));

     pan3.add(new JButton("nom1"));
     pan3.add(new JButton("nom2"));
     pan3.add(new JButton("prenom1"));
     pan3.add(new JButton("prenom2"));
     pan3.add(new JButton("adresse"));

//…Reste
```

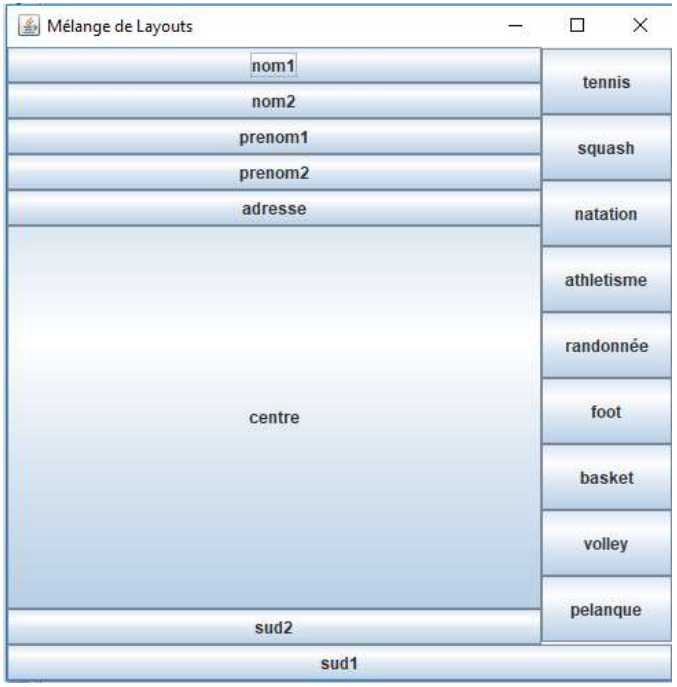


4EME ÉTAPE

❑ 4éme Etape :

Après on ajoute un autre pan4 de type GridLayout avec 9 ligne et 1 colonne et on le place à l'est de pan1.

```
//…… Debut
//…
// La Quatrieme division

     pan1.add(pan4,BorderLayout.EAST);
     pan4.setLayout(new GridLayout(9,1));

     pan4.add(new JButton("tennis"));
     pan4.add(new JButton("squash"));
     pan4.add(new JButton("natation"));
     pan4.add(new JButton("athletisme"));
     pan4.add(new JButton("randonnée"));
     pan4.add(new JButton("foot"));
     pan4.add(new JButton("basket"));
     pan4.add(new JButton("volley"));
     pan4.add(new JButton("pelanque"));
//…Reste
```



5EME ÉTAPE

❑ 5éme et 6éme Etape : Après on ajoute un autre pan4 de type GridLayout avec 9 ligne et 1 colonne et on le place à l'est de pan1.

```
public sport5(){
//….
// La Premiere division
pan1.setLayout(new BorderLayout());
pan1.add(pan6,BorderLayout.SOUTH); // changement
// La Deuxieme division
pan2.add(pan5,BorderLayout.SOUTH);
// La Troisieme et Quatrieme division
// ……
// La Cinquieme division

pan5.setLayout(new FlowLayout());
pan5.add(new JButton("sexe"));
pan5.add(new JButton("homme"));
pan5.add(new JButton("femme"));

// La sixième division
pan6.setLayout(new FlowLayout());
pan6.add(new JButton("ok"));
pan6.add(new JButton("annuler"));
```

6EME ÉTAPE

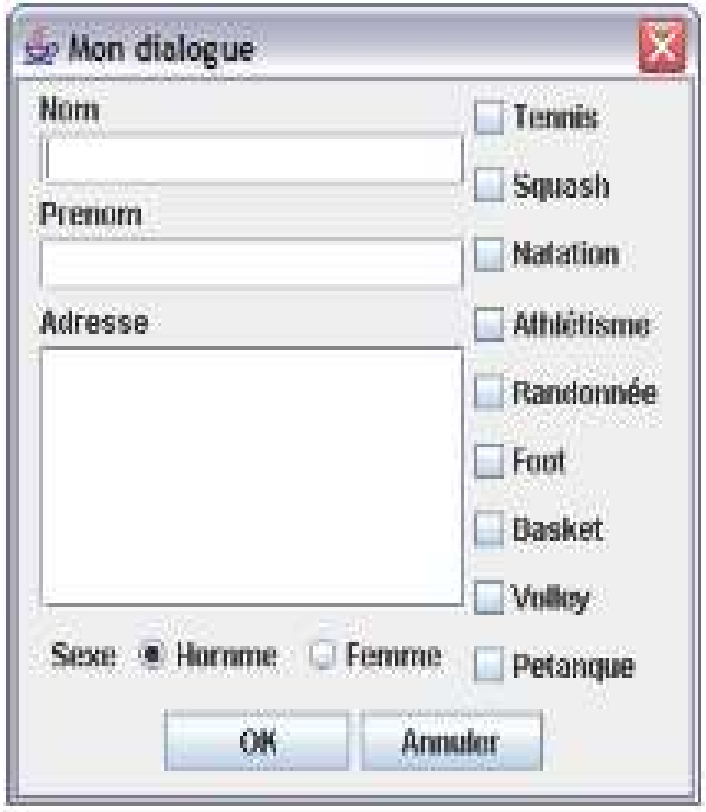
- **Dernière Etape :** Attributions des composantes de bases

```
public sport5(){
//….
// La Premiere division
pan1.setLayout(new BorderLayout());
pan1.add(pan6,BorderLayout.SOUTH); // changement
// La Deuxieme division
pan2.add(pan5,BorderLayout.SOUTH);
// La Troisieme et Quatrieme division
// ……
// La Cinquieme division

pan5.setLayout(new FlowLayout());
pan5.add(new JButton("sexe"));
pan5.add(new JButton("homme"));
pan5.add(new JButton("femme"));

// La sixième division
pan6.setLayout(new FlowLayout());
pan6.add(new JButton("ok"));
pan6.add(new JButton("annuler"));
```

7EME ÉTAPE

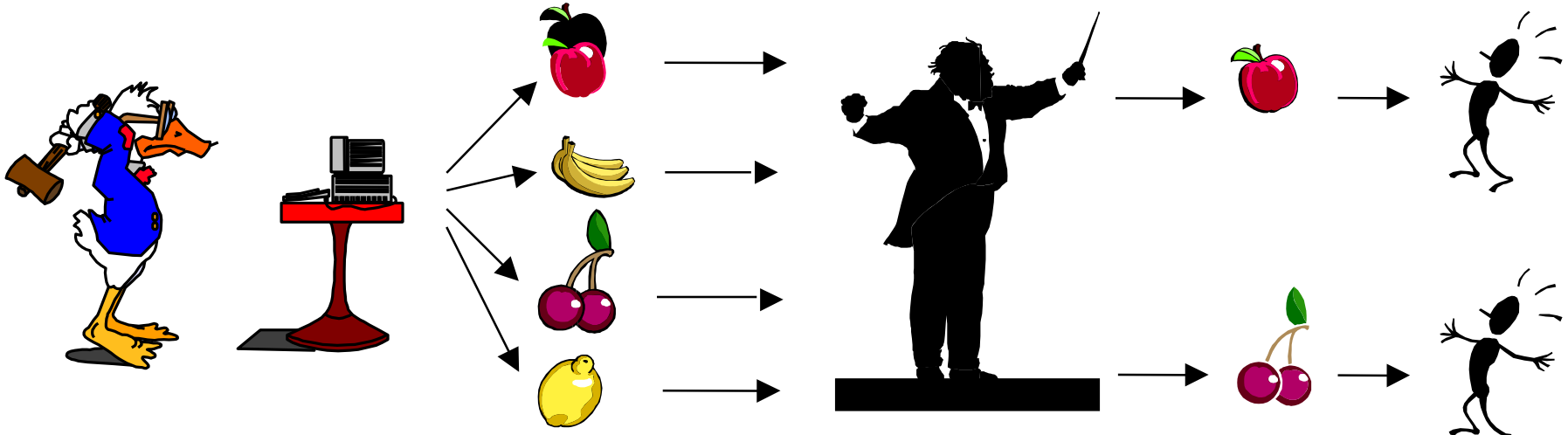
❑ **Dernière Etape :** Attributions des composantes de bases

```
public sport5(){
//….
// La Premiere division
pan1.setLayout(new BorderLayout());
pan1.add(pan6,BorderLayout.SOUTH); // changement
// La Deuxieme division
pan2.add(pan5,BorderLayout.SOUTH);
// La Troisieme et Quatrieme division
// ……
// La Cinquieme division

pan5.setLayout(new FlowLayout());
pan5.add(new JButton("sexe"));
pan5.add(new JButton("homme"));
pan5.add(new JButton("femme"));

// La sixième division
pan6.setLayout(new FlowLayout());
pan6.add(new JButton("ok"));
pan6.add(new JButton("annuler"));
```



Partie II : Programmation évènementielle

L'INTERCEPTION DES ACTIONS DE L'UTILISATEUR



L'utilisateur réalise une action

Des événements sont alors générés

La Machine Virtuelle reçoit tous les événements

Seuls les événements écoutés sont transmis

Source de l'événement

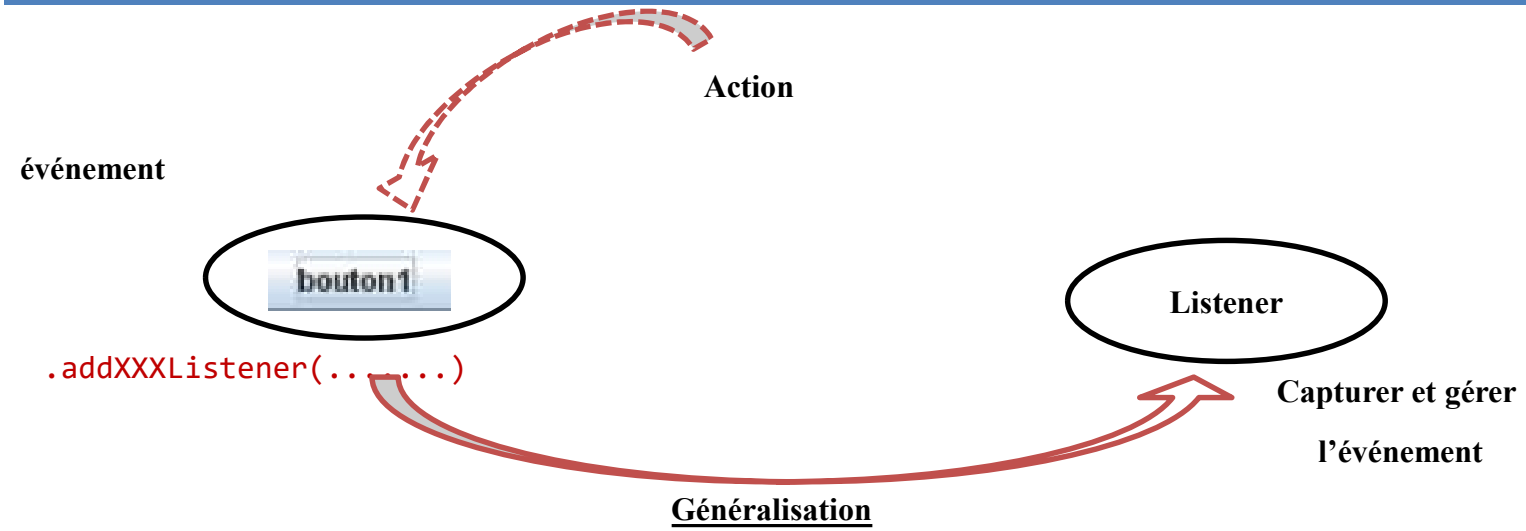
JButton, JTextField…

Événements

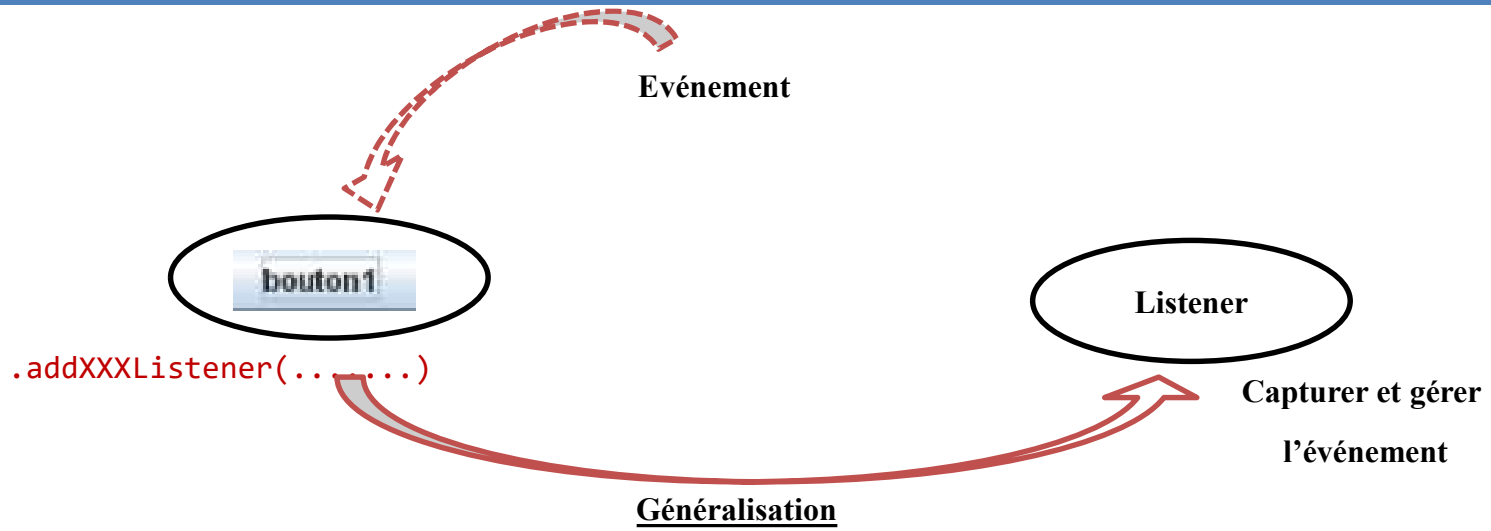
ActionEvent, MouseEvent…

Écouteurs

ActionListener, MouseListener

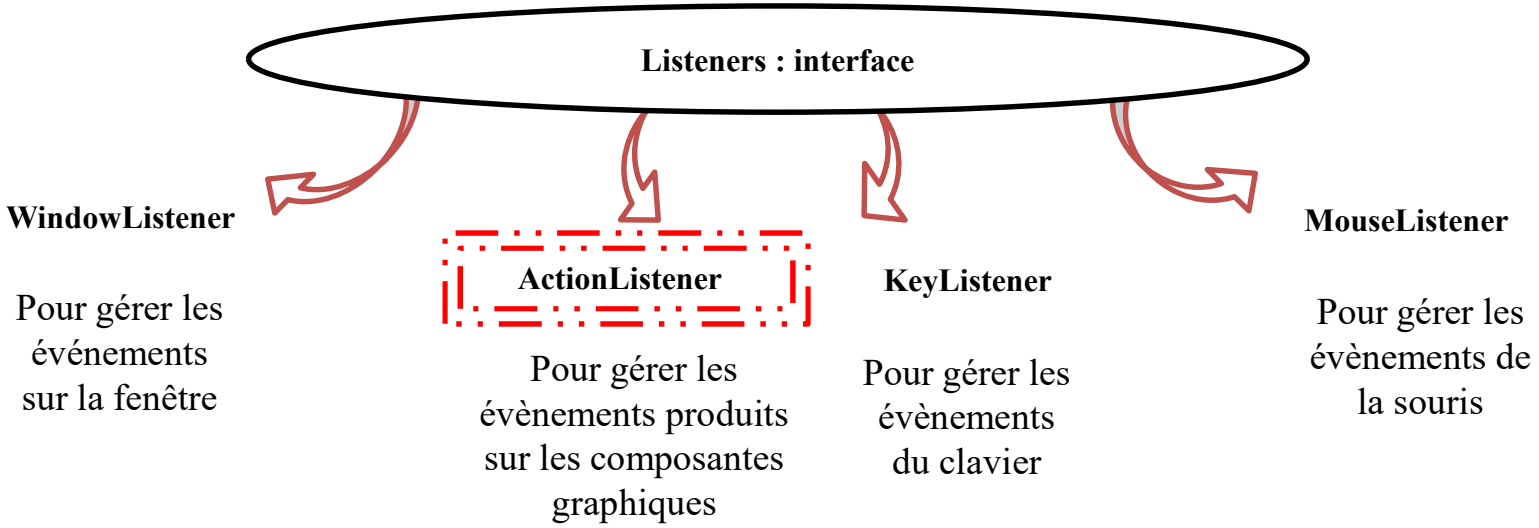


- Les sources d' évènement procèdent des méthodes *addXXXListener*() qui leur permettent d'enregistrer les écouteurs d' vènement sélectionnés de type *XXXListener*.
- Lorsqu'un évènement arrive à la source, celle-ci envoie une notification à tous mes objets écouteurs recensés pour cet évènement.



- Pour gérer cet événement dans une application, il faut :
  1. Créer une classe implémentant l'interface ***XXXListener***
  2. Brancher un bouton sur un écouteur ***addXXXListner***
  3. Redéfinir la réponse aux événements utilisateur, produits dans l'interface, dans sa/ses propres méthode(s)

- Dans JAVA, pour qu'un objet puisse répondre à un événement, il faut lui attacher un écouteur (Listener).
- Ces évènements sont gérés par AWT



GESTIONNAIRE D'ÉVÉNEMENT ACTIONLISTENER

ActionListener

- *ActionListener* est une interface qui définit une seule méthode:

```
public void actionPerformed (ActionEvent e);
```

- L'événement *actionPerformed* est produit quand on valide une action par un clique ou par la touche de validation du clavier.
- Pour gérer cet événement dans une application, il faut :
  1. Créer une classe implémentant l'interface *ActionListener*
  2. Brancher un bouton sur un écouteur *addActinListner*
  3. Redéfinir la réponse aux événements utilisateur, produits dans l'interface, dans la méthode *actionPerformed.*
- Quand on clique, par exemple, sur un bouton qui est branché sur cet écouteur, la méthode *actionPerformed* de l'écouteur s'exécute.
- La classe implémentant cette interface s'appelle un listener (écouteur) ou un gestionnaire d'événements.

L'INTERCEPTION DES ACTIONS DE L'UTILISATEUR


```
import java.awt.*; import javax.swing.*;
import java.awt.event.*;//package pour les événements
public class Exemple extends JFrame  implements ActionListener  {
private JTextField saisie= new JTextField(20);
private JButton BVider = new JButton("Vider");
private JPanel pan= new JPanel();

public Exemple(){
       setTitle(" Vider un TextField ");
      pan.setLayout(new FlowLayout());
       BVider.addActionListener(this);    // la fenêtre est écouteur
       pan.add(saisie); pan.add(BVider); add(pan);
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       setVisible(true);// La sixième division
}

Public void actionPerformed (ActionEvent e){
Saisie.setText("");   }
```

1.

2.

3.

VERSION AMÉLIORÉ (CLASS ADAPTATIVE)


```
import java.awt.*; import javax.swing.*; import java.awt.event.*;
class Exemple extends JFrame  {
private JTextField saisie= new JTextField(20);
private JButton BVider = new JButton("Vider");
private JPanel pan= new JPanel();

public Exemple(){
       setTitle(" Vider un TextField ");
       pan.setLayout(new FlowLayout());
       BVider.addActionListener( new ActionListener() {
             @Override
2. & 3.      public void actionPerformed(ActionEvent e) {
                          saisie.setText(""); }      }
                                                              );
 pan.add(saisie); pan.add(BVider); add(pan);
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       setVisible(true);// La sixième division
}   }
```

EXPLICATION

- Un composant enregistre des auditeurs d'évènements(Listeners).
- Lorsqu'un événement se produit dans un composant, il est envoyés aux Listeners enregistrés.
- Chaque auditeur définit les actions à exécuter dans des méthodes aux noms prédéfinis.
- Exemple :
  - Un Bouton enregistre des *ActionListener*
  - Lors d'un clic sur un bouton, un *ActionEvent* est envoyé aux *ActionListener* enregistrés.
  - Ceci provoque l'exécution de la méthode *actionPerformed* de chaque *ActionListener*.
  - Associer événement au bouton

```
import java.awt.*; import javax.swing.*; import java.awt.event.*;
public class TestEvent extends Frame {
      Label l=new Label("Votre Nom:");
      TextField t=new TextField (12); List liste  =new List(); Panel p=new Panel();
      Button b  =new Button ("Ajouter"); Button b2  =new Button ("Quitter");
public TestEvent(){
add(p); p.setLayout(new FlowLayout ());
p.add(l); p.add(t); p.add(b); p.add(b2); p.add(liste);

b.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
      liste.add (t.getText());//Lire le contenu du t et l'ajouter à la liste
      }});
b2.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
      System.exit(0); }});
setBounds(10,10,250,250);
setVisible(true); }
```

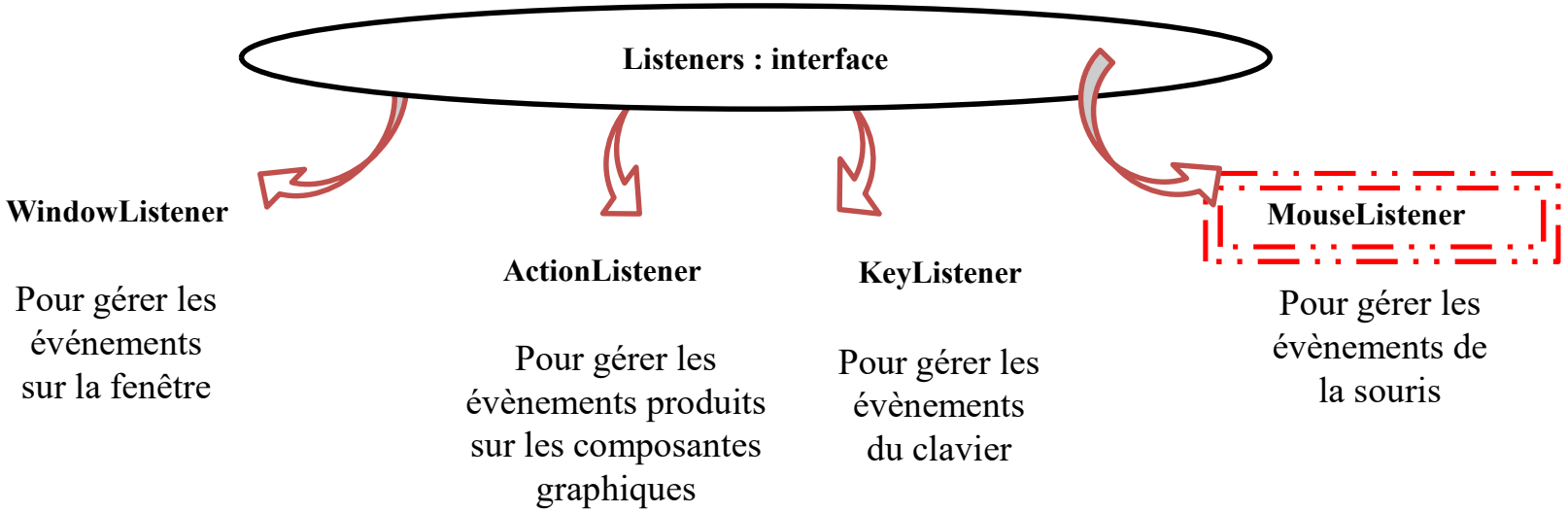
Remarques :

- L'interface *ActionListener* utilisée dans l'exemple n'est pas limitée aux clic sur des boutons, elle peut être utilisée dans bien d'autres situations, par exemple:
  - Lors de la sélection d'un élément de menu.
  - Lors de la sélection d'un élément dans une liste avec un double-clique.
  - Lorsque la touche "*Entrée*" est activée dans un champ de texte. On peut ajouter la ligne:

    saisie.addActionListener(this);
  - Lorsqu'un composant *Timer* déclenche l'écoulement d'un temps donné.
- En résumé, *ActionListener* s'utilise de la même manière dans toutes les situations qu'on vient d'évoquer; unique méthode *actionPerformed()* reçoit en paramètre un objet de type *ActionEvent*. Cet objet fournit des informations sur l'événement qui a été déclenché.

GESTION DES ÉVÉNEMENTS

- Dans JAVA, pour qu'un objet puisse répondre à un événement, il faut lui attacher un écouteur (Listener).
- Ces évènements sont gérés par AWT



MouseListener

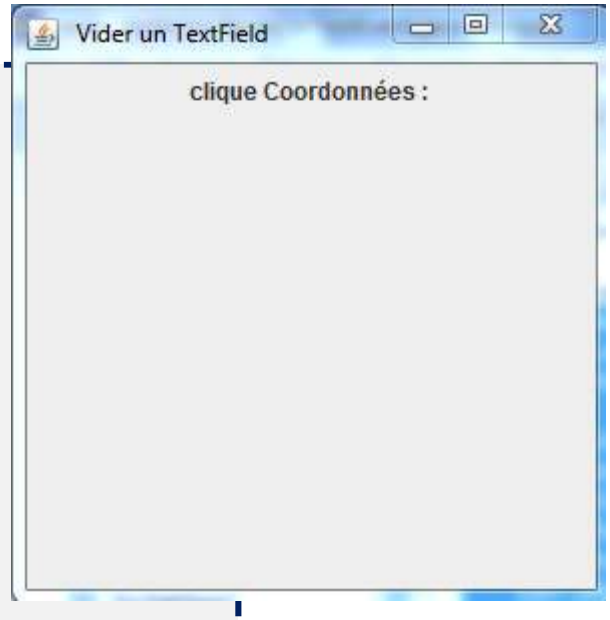
- *MouseListener* est une interface qui définit 5 méthodes:

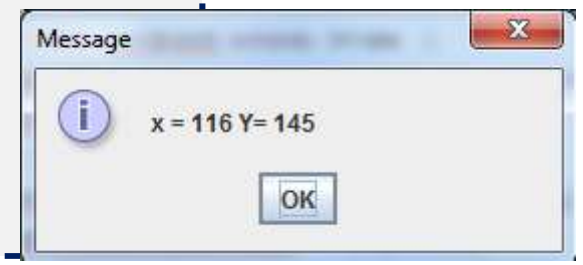
```
public void mouseEntered (MouseEvent e);
public void mouseExited  (MouseEvent e);
public void mousePressed (MouseEvent e);
public void mouseReleased(MouseEvent e);
```

- Pour gérer cet événement dans une application, il faut
  1. Créer une classe implémentant l'interface *MouseListener*
  2. Brancher un bouton sur un écouteur *addMouseListner*
  3. Redéfinir la réponse aux événements utilisateur, produits dans l'interface, dans ses 4 méthodes.

MOUSE LISTENER (1)


```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;//package pour les événements
public class Exemple extends JFrame implements MouseListener {
private JLabel saisie= new JLabel("clique Coordonnées : ");
private JPanel pan= new JPanel();

public Exemple1(){
    setSize(300,300);
    setTitle(" Vider un TextField ");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    pan.setLayout(new FlowLayout());
    pan.add(saisie);
    add(pan);
    pan.addMouseListener(this);// la fenetre est ecouteur
    setVisible(true);
}
```



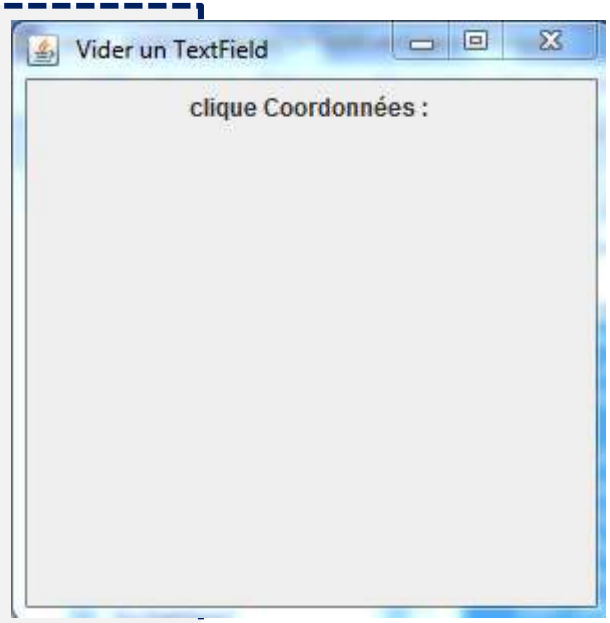


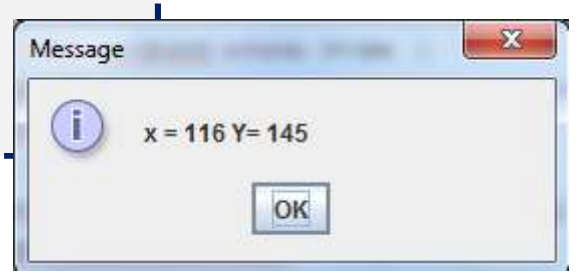
MOUSE LISTENER (2)


```
@Override
public void mouseClicked(MouseEvent e) {
  JOptionPane jop1= new JOptionPane();
  jop1.showMessageDialog(null,"x = "+e.getY()+" Y= "+e.getX()); }

@Override
public void mouseEntered(MouseEvent arg0) {   }
@Override
public void mouseExited(MouseEvent arg0)  {   }
@Override
public void mousePressed(MouseEvent arg0) {    }
@Override
public void mouseReleased(MouseEvent arg0) {   }
}

public class EXO2{   public static void main(String[] args) {new Exemple1();}  }
```

MOUSE LISTENER (1)


```
public class TestEvent extends JFrame  {
      private JLabel saisie= new JLabel("clique Coordonnées : ");
      private JPanel pan= new JPanel();
public TestEvent(){
    setSize(300,300); setTitle(" Vider un TextField ");
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    pan.setLayout(new FlowLayout());
    pan.add(saisie);    add(pan);
    pan.addMouseListener(new MouseAdapter() {
      public void mouseClicked(MouseEvent e) {
      JOptionPane jop1= new JOptionPane();
      jop1.showMessageDialog(null,"x = "+e.getY()+" Y= "+e.getX());
     }});// la fenetre est ecouteur
    setVisible(true);
}
```





AUTREMENT - RÉSUMÉ -

1) Importer le groupe de classes java.awt.event

```
import java.awt.event.*;
```

2) La classe doit déclarer qu'elle utilisera une ou plusieurs interfaces d'écoute : par exemple *ActionListener*

```
class Fenetre1 extends JFrame implements ActionListener {
```

3) Appel à la méthode *addActionListener*() pour que le composant possède un écouteur pour l'événement utilisateur concerné c-a-d préciser la classe qui va recevra et gérera l'événement .

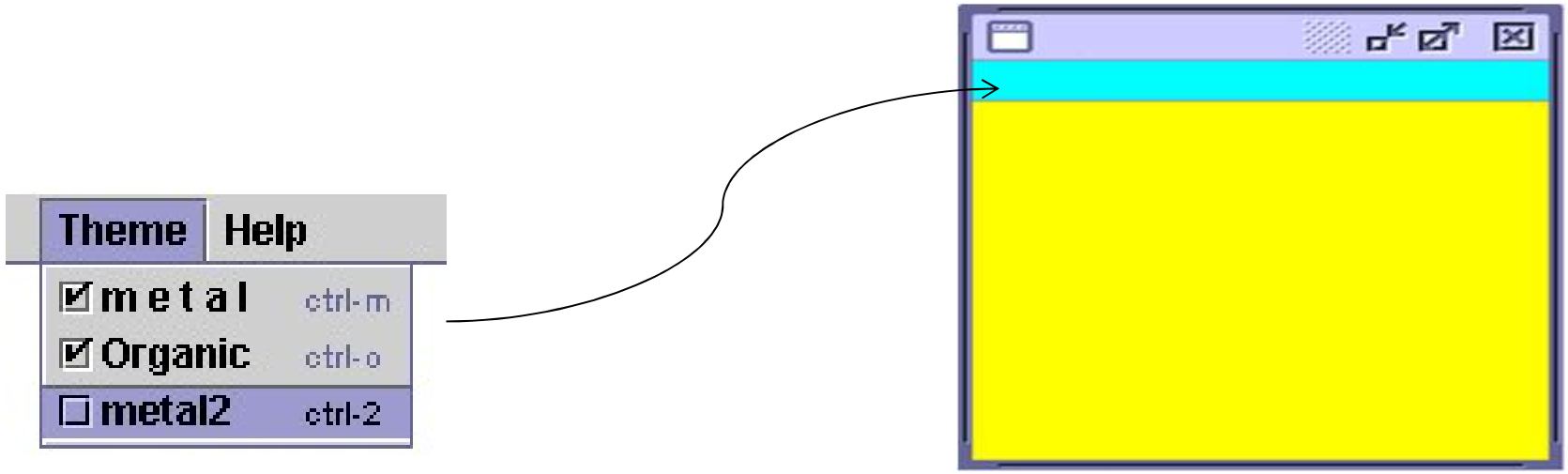
```
JButton bouton = new JButton("bouton simple");
bouton.addActionListener(this);
```

4) Implémenter les méthodes déclarées dans les interfaces ,par exemple :*actionPerformed*()

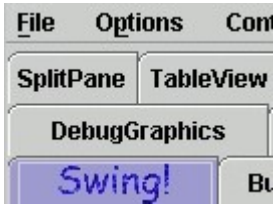
```
public void actionPerformed(ActionEvent arg0) {
    System.out.println("Vous venez de cliquer sur le bouton");

}
```

JMENU

- Une instance de JMenuBar par Jframe
  - setJMenuBar(JMenuBar mb);
- Plusieurs Jmenu par JMenuBar
  - add(JMenu jm);
- Plusieurs JMenuItem/JCheckboxMenu par Jmenu
  - add(JMenuItem mi);
  - addSeparator();



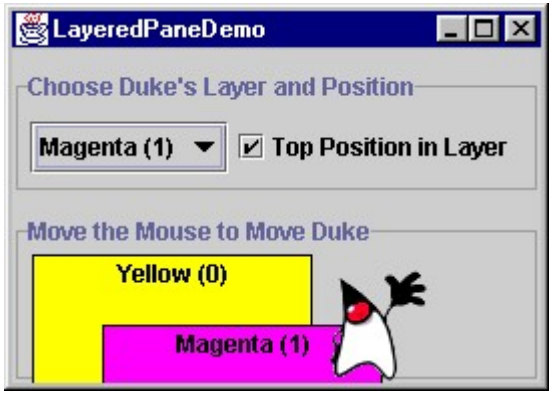
Dialog



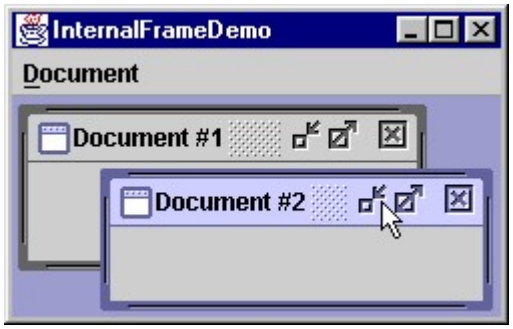
Tabbed Pane

Frame

Split pane



Layered pane



Internal frame

Tool bar

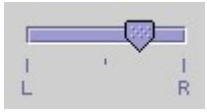
AUTRES COMPOSANTS DE BASE

File chooser

Color chooser

Progress bar

Tool tip



Slider

Tree

Merci pour votre attention