# Chapitre sur les Interfaces Graphiques en Java

Z.Habbas

## 1 Introduction

Une interface graphique est formée d'une ou plusieurs fenêtres qui contiennent différents composants graphiques tels que :

— Les boutons
— Les listes déroulantes
— Les menus
— Les champs texte ...

L'utilisateur peut intéragir à tout moment avec plusieurs objets graphiques : clic sur un bouton, choix dans une liste déroulante, dans un menu, ....

Ces actions peuvent changer le cheminement d'un programme de façon imprévisible.

Les interfaces graphiques imposent donc un style de programmation guidé par les événements (programmation événementielle).

Les intergaces graphiques sont souvent appelés GUI : (Graphical User Interface).

## 2 Les API utilisées

### 2.1 Swing et AWT

Il y a deux bibliothèques dédiées au graphisme : SWING et AWT. Toutes les deux font partie d'une collection plus vaste de logiciels appelée JFC (Javac Foundation Classes).

— AWT (Abstract Window Toolkit) : boîte à outils d'origine de l'interface utilisateur qui est donnée par défaut dans le SE Java.

— est une boîte à outils d'interface utilisateurs construite au-dessus de AWT et qui existe depuis la version 1.1 du jdk.

## 2.2 Swing ou AWT ?

Tous les composants AWT ont leur équivalent Swing en plus joli et avec plus de fonctionnalités.
Swing offre de nombreux composants qui n'existent pas dans AWT.

Il est fortement conseillé d'utiliser Swing.

Voici quelques paquetages utiles

— AWT : java.awt, java.awt.event

— Swing : javax.swing, javax.swing.event, ...

# 3 Modèle MVC : Modèle-Vue-Contrôleur

**Définition 1** *Le modèle MVC est un mode de conception d'applications graphiques permettant de distinguer les données, leur apparence graphique ainsi que les traitements pouvant être effectuées sur les données.*

Une interface graphique se compose de trois parties :

1. **Le modèle** : correspond à une classe chargée de stocker les données, qui permet à la vue de lire son contenu et informe la vue d'éventuelles modifications est bien représenté par une classe.

2. **La vue** : Correspond à son apparence graphique (les composants, les listes déroulantes, ...). Elle ci peut être représentée par plusieurs classes.
3. **Le contrôleur** : Associe des traitements à des évenements pouvant se produire au niveau de composant. Le contrôleur gère les interactions de l'utilisateur et propage les modifications vers la vue et le modèle éventuellement.

   **Exemple 1** *La décomposition MVC de ce bouton (instance de la classe Jbutton) est :*

   — *Le modèle contient la chaîne de caractère quitter.*

   — *La vue est décrite par le schéma suivant :*

   — *Le contrôleur implémentera le traitement à effectuer lors d'un clic.*

**Exemple 2 Une première fenêtre**

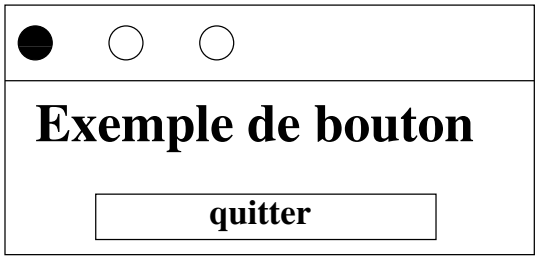


FIGURE 1 – Un exemple de Jbutton.

```
package TG1;
import javax.swing.JFrame;
public class InterfaceG1 {
private JFrame fenetre;
public InterfaceG1()
{
fenetre = new JFrame("premiere_fenetre");
fenetre.setSize(300,300);
fenetre.setLocation (100,100);
fenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  }
     public static void main (String[]  argv) {
     InterfaceG1  monInterface = new InterfaceG1();
     fenetre.setVisible(true);
}
```

**Exemple 3 Création d'un premier bouton**

```
package TG1;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class InterfaceG3  extends JFrame{
private JFrame fenetre;
private JButton bouton;
private JPanel panel;
public InterfaceG3()
{
```

```
fenetre = new JFrame (" Fenetre Bouton ");
fenetre.setSize(500,500);

// fenetre.setLocation(100,100);

setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);
panel = new  JPanel();
    bouton = new JButton ("bouton");
    panel.add(bouton);
    getContentPane().add(bouton);
pack();

}

public static void main (String[]  argv) {
InterfaceG3  monInterface = new InterfaceG3();
setVisible(true);
}

}
```

# 4 Les composants graphiques

Un composant graphique est l'objet fondamental d'une interface utilisateur sous Java.
Un composant graphique comprend

- des fenêtres
- des boutons
- des cases à cocher
- des barres de défilement
- des listes
- des menus
- des champs de saisie ...

Pour être utilisé, un composant doit être généralement placé dans un conteneur. La classe de composant graphique est racine de l'arborescence de sawing et hérite de la classe Container de AWT.

**Définition 2** *Une interface graphique est un objet d'une classe de type java.awt.Container qui regroupe des composants simples et d'autres containers qui regroupent à leur tour*

*des composants simples.*

## 4.1 Les fenêtres et les cadres

Les fenêtres et les cadres sont les conteneurs de plus haut niveau des composants java. Ce sont les seuls composants affichables sans être attachés ou ajoutés à un autre container. Il s'agit des JFrame et des JWindow.

Après avoir créé un Jframe ou un Jwindow, il suffit d'appeler la méthode setVisible() pour afficher la fenêtre.

**Exemple 4 Fenêtres et cadres**

```
package TG1;
import javax.swing.JFrame;
public class Fenetre {
 public static void main (String[]  argv) {
 JFrame  frame  = new JFrame("frame");
 frame.setSize(300,300);
frame.setLocation (100,100);
  JWinfow window = new JWindow ();
   window.setSize(300,300);
window.setLocation (100,100);
frame.setVisible();
window.setVisible();
}
}
```

Le constructeur JFrame peut prendre comme argument un String indiquant le titre du cadre qui sera affiché. Il est possible de créer un JFrame sans titre puis d'appeler setTitle() pour fournir le titre.

```
 JFrame  frame  = new JFrame("frame");
```

ou

```
JFrame  frame  = new JFrame();
frame.setTiltle("frame");
```

Le constructeur JWindow ne comporte pas d'arguments.
setVisible() est appelée pour l'affichage à l'écran.

Les fenêtres ne se ferment pas naturellement. Pour les fermer il faut un ctrl-C.
**Autres méthodes**

— La méthode setLocation() permet de définir la position à l'écran.

— Les méthodes toFront() et toBack() permettent de placer une fenêtre devant ou derrière d'autres fenêtres.

— On peut interdire le redimensionnement d'une fenêtre avec setResizable(false).

## 4.2 Utilisation des panneaux de contenu

Les fenêtres et les cadres ne se comportent pas comme des conteneurs classiques. Pour pouvoir ajouter des composants à une fenêtre ou un cadre, il faut récupérer le panneau de contenu par la méthode getContentPane().

**Exemple 5 Fenêtres et cadres**

```
package TG1;
import javax.swing.JFrame;
public class Contenu  {
 public static void main (String[]  argv) {
 JFrame  frame  = new JFrame("frame");
 frame.setSize(300,300);
frame.setLocation (100,100);
Container content = frame.getContentPane();
content.setLayout(new FlowLayout());
Content.add(new Jlabel("Hello"));
Content.add(new Jlabel("Coucou"));
frame.pack();
frame.setVisible();
   }
}
```

Ou bien

**Exemple 6 Fenêtres et cadres**

```
package TG1;
import javax.swing.JFrame;
public class Contenu  {
 public static void main (String[]  argv) {
```

```
 JFrame  frame  = new JFrame("frame");
frame.setLocation (100,100);
Container content = new JPanel();                    ;
content.setLayout(new FlowLayout());
Content.add(new Jlabel("Hello"));
Content.add(new Jlabel("Coucou"));
frame.setContentPane(Content);
frame.pack();
frame.setVisible();
   }
}
```

**Remarque 1** *Un panel est un composant initialement vide auquel seront ajoutés des composants.*

*La disposition peut être spécifiée lors de l'appel au constructeur du paneL.*

***Exemple 7*** *JPanel panel = new JPanel (new BoarderLayout()) ;*
*ou par un appel explicite à setLayout ()*

*panel.setLayout(new FridLayout(4,5)) ;*

**Remarque 2 Attention** *: avec AWT le dépôt du composant s'effectue directement sur le conteneur.*

— **Avec AWT**

Frame fenetre = new Frame( ) ; Button bouton = new Button( ) ; fenetre.add(bouton) ;

— **Avec Swing**

JFrame fenetre = new JFrame( ) ; JButton bouton = new JButton( ) ;
Container Contenu = fenetre.getContentPane( ) ; Contenu.add(bouton) ;

## 4.3 JBoutons et Jlabels

Les composants JLabel sont des simples textes ou images à placer dans un composant. Ils ne sont pas associés à un événement.

Voir exemple des Jbutton plus haut.

## 4.4 Autres composants de swing

1. Les cases à cocher et les boutons radio
2. Les listes et les boîtes combobox
3. les boutons fléchés

## 4.5 Les gestionnaires de placement

**1. Le flowlayout** Le flowlayout essaie de disposer les composants avec leur taille souhaitée de gauche à droite et de haut en bas du conteneur. Un flowlayout peut spécifier une justification LEFT, CENTER ou RIGHT et un espacement entre deux compsants verticaux ou horizontaux. Par défaut un flowlayout utilise la justification CENTER. Flowlayout est le gestionnaire par défaut des composants JPanel.

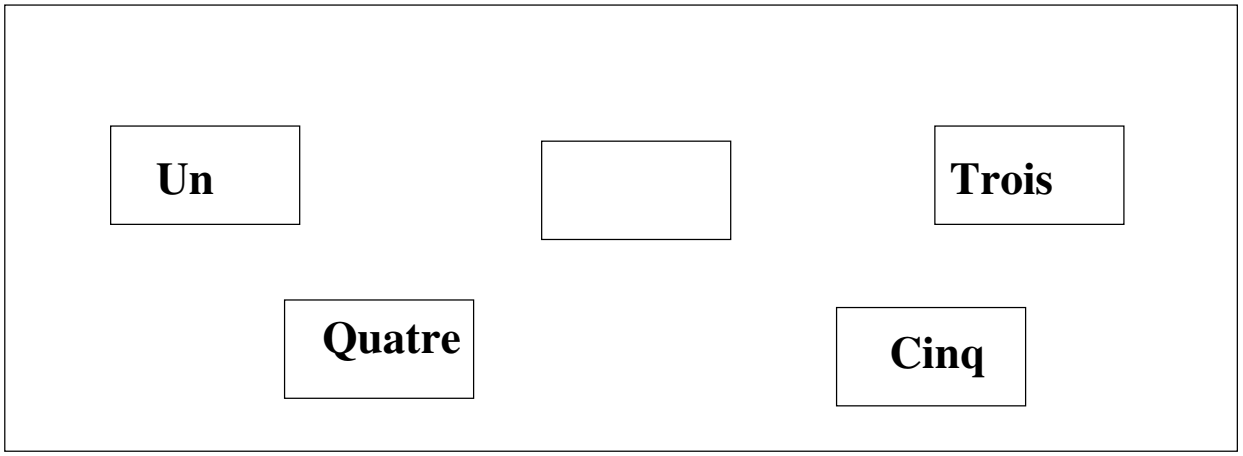


FIGURE 2 – Flowlayout.

Voici l'implémentation java !!!

**Exemple 8**
```
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
public class exemplefl  extends JFrame {
public exemple1()
{
```

```
// Définir  le flowLayout

add(new JButton ("Un"));
add(new JButton ("Deux"));
add(new JButton ("Trois"));
add(new JButton ("Quatre"));
add(new JButton ("Cinq"));
pack();
}

 public static void main (String[]  argv) {
JFrame fenetre = new JFrame (" Exemple de FlowLayout ");

Examplefl  fl = new Examplefl();
 fl.setVisible(true);


}
}
```

**2. Le borderlayout** Le gestionnaire BorderLayout essaie de disposer selon les conq composants géographiques représentées par des constantes de la classe BORDERLatout : NORTH, SOUTH, EAST,WEST,CENTER avec des espacements supplémentaires éventuels. C'est le gestionnaire par défaut de JFrame.

**Exemple 9 BorderLauout**

```
package TG1;
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Examplebl  extends JFrame {
public Examplebl()
```

```
{
JFrame fenetre = new JFrame (" Exemple de Border Layout ");

// Définir  le borderlayout

setLayout (new BorderLayout());
add("North", new JButton ("Bouton1"));
add("South", new JButton ("Bouton2"));
add( "West", new JButton ("Bouton3"));
add("East", new JButton ("Bouton4"));
add("Center", new JButton ("Bouton5"));
pack();
}

public static void main (String[]  argv) {
Exemplebl Interface = new Exemplebl();
Interface.setVisible(true);

}
}
```

**2. Le Gridlayout** Le GridLayout dispose les composants selon les lignes et les colonnes régulièrement espacées. Les composants sont arbitrairement redimensionnés pour tenir dans la grille.
GridLayout prend en entrée le nombre de lignes et de colonnes. Si le nombre d'objets est trop grand il ajoute des colonnes supplémentaires pour faire entrer les objets. Si on lui précise le nombre de lignes et que le nombre de colonnes est à 0 il organisera le nombre de colonnes en fonction des objets à caser.

Voici l'implémentation en java!!!

**Exemple 10 GridLayout**

```
package TG1;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
```

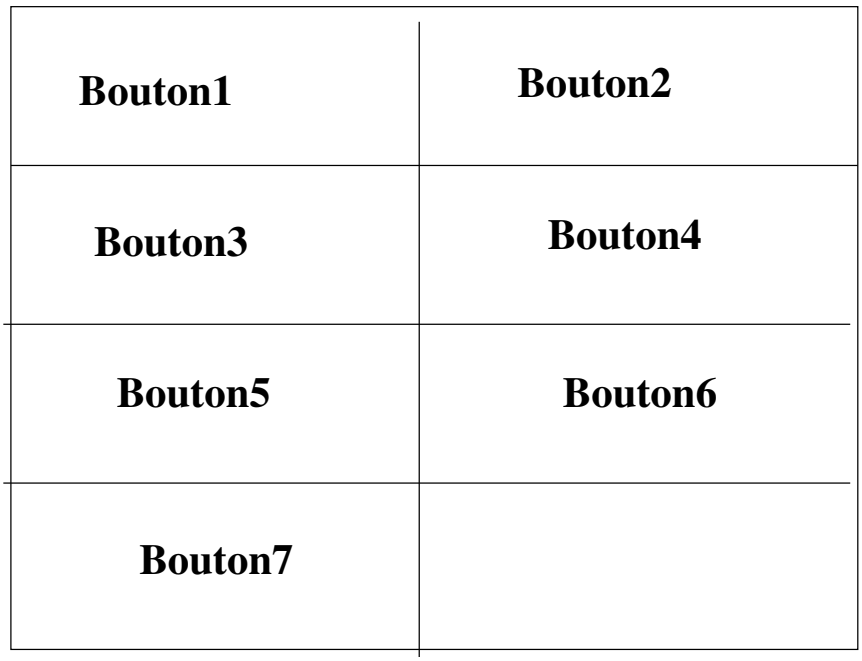


FIGURE 4 – GridLayout.

```
public class Examplegl  extends JFrame {
public Examplegl()
{

JFrame fenetre = new JFrame (" Exemple de Grid Layout ");

// Définir  le GridLayout
setLayout (new GridLayout (4,2));
add(new JButton ("Bouton1"));
add(new JButton ("Bouton2"));
add(new JButton ("Bouton3"));
add(new JButton ("Bouton4"));
add(new JButton ("Bouton5"));
add(new JButton ("Bouton6"));
 add(new JButton ("Bouton7"));

pack();
}

public static void main (String[]  argv) {
Examplegl Interface = new Examplegl();
Interface.setVisible(true);


}
}
```

**3. La BoxLayout** BoxLayout sert à la création de barres d'outils ou de barres de boutons. Il dispose les composants dans une ligne ou une colonne unique.
Cette classe contient deux méthodes statiques permettant de créer une boîte horizontale ou verticale.

```
Container horizontalBox = Box.createHorizontalBox();
Container verticalBox = Box.createverticalBox();
```

Voici l'implémentation en java!!

**Exemple 11 BoxLayout**

```
package TG1;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;

public class Exemplebl  extends JFrame {
public Exemplebl()
{
JFrame fenetre = new JFrame (" Exemple de BoxLayout ");

Container box = Box.createHorizontalBox();
// Définir  le boxLayout
box.add(new JButton ("Bout1"));
box.add(new JButton ("Bout2"));
box.add(new JButton ("Bout3"));
box.add(new JButton ("Bout"));
box.add(new JButton ("Bout5"));
frame.getContePane().add(box, BorderLayout.CENTER);
frame.pack();
}
public static void main (String[]  argv) {
Exemplegl Interface = new Exemplegl();
Interface.setVisible(true);


}
}
```

**4. GridBagLayout** A voir plus tard!!!

# 5 Contrôleur : Les écouteurs et contrôleurs d'événements

Le rôle du contrôleur est d'intercepter certains événements et d'effectuer un traitement associé au type de l'événement.
Un événement peut être produit par un clic sur une fenêtre, sur un bouton, par la sélection d'un élément dans une liste, la pression d'une touche clavier ...

## 5.1 Evénement

Un événement graphique est représenté dans java comme un ojet dont la classe hérite de java.awt.AWTEvent. Les plus couramment utilisés sont :

1. **ActionEvent** : Se produit lorsque l'action est effectuée sur un composant (clic sur un bouton).

2. **ItemEvent** : Se produit lorsqu'on coche une case.

3. **KeyEvent** : Se produit par une touche clavier.

4. **MouseEvent** : Se produit par un mouvement de la souris.

5. **WindowsEvent** : Se produit lorsque l'action est effectuée sur la fenêtre. Exple : clic sur l'icone de fermeture de la fenêtre.

Des méthodes sont rattachées à chacune de ces classes pour avoir accès à plus de détail sur l'événement. On peut récupérer le composant source de l'événement, la position de la souris lors du clic, ....

## 5.2 Interface Listener

Le contrôleur qui intercepte un certain type d'événement doit implémenter une des interfaces héritant de java.util.EventListener.

L'interface à implémenter dépend du type de l'événement.
Après la création d'un objet contrôleur il faut le rattacher à un ou plusieurs composants. Ler contrôleur n'intercepte que les événement des composants auxquels il est rattaché.
Cette opération se fait par l'appel à une méthode du composant de la forme
add .... Lsitener (...)

| Contrôleur | Evénement | Méthode à implémenter |
|---|---|---|
| ActionListener | ActionEvent | ActionPerformed(Action Event) |
| ItemListener | ItemEvent | ItemStateChanged (Item Event ) |
| KeyListener | KeyEvent | KeyPressed (Key Event)<br>KeyReleazed (Key Event)<br>KeyTyped (Key Event) |
| MouseListener | Mouse Event | mouseClicked (Mouse Event)<br>mouseEntered ( (Mouse Event)<br>mouseExited ( (Mouse Event)<br>mousePressed ( (Mouse Event)<br>mouseReleased ( (Mouse Event) |
| WindowsListener | Window Event | windowActivated (Window Event )<br>windowClosed (Window Event )<br>windowSeactivatedg (Window Event )<br>windowDeconified (Window Event )<br>windowActivated (Window Event )<br>windowIconified (Window Event )<br>windowOpened (Window Event )<br>windowActivated (Window Event ) |

**Exemple 12** *Rattacher le contrôleur Mon actionListener à un un bouton s'écrit :*

*Monbouton.addActionListener(MonActionListener) ;*

## 5.3 Exemples de contrôleurs

**1. Ecouteurs de clics sur une fenêtre : différentes approches**

1. **Approche 1**

```
// La fenetre
package EXO1;
import javax.swing.*;
import java.awt.event.*;
public class Fenetre1  extends JFrame implements MouseListener {

public Fenetre1()
{
 setTitle("Gestion des clics sur fenetre");
```

```
 setBounds (10, 20, 300, 200);
 addMouseListener (this);
}

 public void mousePressed (MouseEvent ev )
 {  System.out.println( " Clic appuyé en " + ev.getX() + " " + ev.getY());
 }

  public void mouseReleased (MouseEvent ev)
  {  System.out.println( " Clic relache  en " + ev.getX() + " " + ev.getY());
  }
 public void mouseClicked (MouseEvent ev )  { }
 public void mouseEntered (MouseEvent ev )  { }
 public void mouseExited (MouseEvent ev )  { }
}

// Le main
//package EXO1;
import javax.swing.*;
public class Clic1 {
public static void main(String[] args) {
Fenetre1 fenetre = new Fenetre1();
fenetre.setVisible(true);
}
}
```

2. **Approche 2**

```
// La fenetre
package EXO2;
import javax.swing.*;
public class Fenetre2  extends JFrame{
    public Fenetre2()
    {
    setTitle("Gestion des clics sur fenetre");
    setBounds (10, 20, 300, 200);
    addMouseListener (new Ecoute());}

}
```

```
// L'écouteur ou controleur

package EXO2;
import javax.swing.*;
public class Ecoute  implements MouseListener
{
    public void mousePressed (MouseEvent ev )
  {  System.out.println( " Clic appuyé en " + ev.getX() + " " + ev.getY());
    }

  public void mouseReleased (MouseEvent ev )
  {  System.out.println( " Clic relache  en " + ev.getX() + " " + ev.getY());
    }
  public void mouseClicked (MouseEvent ev )  { }
  public void mouseEntered (MouseEvent ev )  { }
  public void mouseExited (MouseEvent ev )  { }
}

//  Le main
package EXO2;
import javax.swing.*;

public class Clic2 {
 public static void main(String[] args) {
  Fenetre2 fenetre = new Fenetre2();
fenetre.setVisible(true);
    }
}
```

3. **Approche 3**

```
// La fenetre

package EXO3;
import javax.swing.*;
import java.awt.event.*;
public class Fenetre3 extends JFrame  {
```

```
 public Fenetre3()

    {
setTitle("Gestion des clics sur fenetre");
             setBounds (10, 20, 300, 200);
             addMouseListener (new Ecoute());}

}


// L'ecoute

package EXO3;
import javax.swing.*;
import java.awt.event.*;

public class Ecoute extends  MouseAdapter  {
 public void mousePressed (MouseEvent ev )
  {  System.out.println( " Clic appuyé en " + ev.getX() + " " + ev.getY());
 }

  public void mouseReleased (MouseEvent ev )
 {  System.out.println( " Clic relache  en " + ev.getX() + " " + ev.getY());
   }

}


// Le main
package EXO3;
import javax.swing.*;
import java.awt.event.*;
public class Clic3 {
public static void main(String[] args)

 { Fenetre3 fen = new Fenetre3();
fen.setVisible(true);
    }
}
```

## 2. Ecouteurs de clics sur plusieurs fenêtres

1. Approche1

```
// La fenetre

package EXO5;
import javax.swing.*;
import java.awt.event.*;
public class Fenetre5

extends JFrame implements MouseListener {
    private static int nbFen =0;
    private int num;
     public Fenetre5()


    {           nbFen++;
            num = nbFen;

setTitle("Fenetre de numero" + num);
                setBounds (10, 20, 300, 200);
                addMouseListener (this);
    }


    public void mousePressed (MouseEvent ev )
    {  System.out.println( " Clic appuyé dans fenetre " + num +"en"  + ev.getX
    }

  public void mouseReleased (MouseEvent ev )
    {  System.out.println( " Clic relache dans fenetre" + num + " en " + ev.ge
    }

   public void mouseClicked (MouseEvent ev ) { }
   public void mouseEntered (MouseEvent ev ) { }
    public void mouseExited (MouseEvent ev ) { }
```

```
    }


// Le main

package EXO5;
import javax.swing.*;

public class Clic5 {

public static void main(String[] args) {
final int nbFen =3;
        for (int i=0; i < nbFen; i++)
    {
 Fenetre5 fen = new Fenetre5();
fen.setVisible(true);
    }
    }
}
```

2. **Idem avec un écouteur séparé**

```
// La fenetre
package EXO6;
import javax.swing.*;
public class Fenetre6 extends JFrame{

 private static int nbFen =0;
    private int num;

     public Fenetre6()


    {   nbFen++;
         num = nbFen;
```

```
setTitle("Fenetre de numero" + num);
         setBounds (10, 20, 300, 200);
                addMouseListener (new Ecoute(num));
    }
}




// L'ecoute

package EXO6;
import javax.swing.*;
public class Ecoute  extends  MouseAdapter
{
     private int num;
    public Ecoute (int num)
    {this.num = num;}

 public void mousePressed (MouseEvent ev )
    {  System.out.println( " Clic appuyé dans fenetre " + num +"en"  + ev.getX
    }

  public void mouseReleased (MouseEvent ev )
    {  System.out.println( " Clic relache dans fenetre" + num + " en " + ev.ge
    }



}


// Le main
package EXO6;
import javax.swing.*;

public class Clic6 {
```

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

final int nbFen =3;
        for (int i=0; i < nbFen; i++)
    {    Fenetre6 fen = new Fenetre6();
        fen.setVisible(true);
    }
    }


}
```

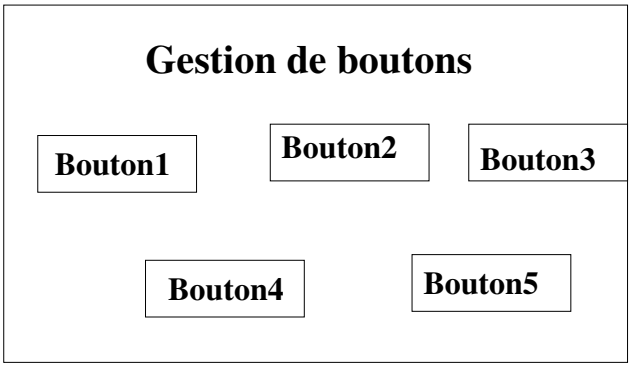


FIGURE 6 – Clic avec GridLayout.

### 3. Création de bouton et choix d'un gestionnaire FloyLaout

**Exemple 13** `// La fenetre`

```
package EX07;
import javax.swing.JButton;
public class Fenetre  extends JFrame implements ActionListener
private static int nBoutons;
// Tableau de JButtons
private JButton[] boutons;
public Fenetre(int nBoutons)
```

```
{ this.nBoutons = nBoutons;
setTitle(" Gestion de boutons");
setSize (300,400);
Container contenu = getContentPane();
contenu.setLayout(new FlowLayout());
boutons = new JButton[nBoutons];
for (int i= 0; i < nBoutons; i++)
{ boutons[i] = new JButton ("Bouton" + (i+1));
contenu.add(boutons[i]);
boutons[i].addActionListener (this);
    }
} public void actionPerformed (ActionEvent e)
 { Object source = e.getSource();
for (int i =0; i < nBoutons; i++)
  if (source == boutons[i])
   System.out.println("Action sur bouton" + (i+1));
    }

}


// Le main

package EXO7;
import javax.swing.*;
public class Clic {

public static void main(String[] args) {
System.out.println("Combien de boutons");
Scanner sc = new Scanner (System.in);
int nBoutons = sc.nextInt();
Fenetre fenetre = new Fenetre (nBoutons);
fenetre.setVisible(true);
    }
    }
```

**Gestion de boutons autre exemple**

**Exemple 14**

```
// La fenetre
package EXO8;
import java.awt.BorderLayout;

public class Fenetre8  extends JFrame  implements ActionListener{


private JButton Bouton1;
private JButton Bouton2;
private JButton Bouton3;
private JPanel panel;
private JLabel titre;
private JLabel message;
public Fenetre8()
{   super(" Exemple de boutons interactifs ");

         setLocation(500, 500);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

         // Définir un border Layout

getContentPane().setLayout(new BorderLayout(5,5));

Bouton1 = new JButton(" Bouton 1");
Bouton2 = new JButton(" Bouton 2");
Bouton3 = new JButton(" Bouton 3");

panel = new JPanel (new GridLayout (3,1));

panel.add(Bouton1);
panel.add(Bouton2);
panel.add(Bouton3);

Bouton1.addActionListener(this);
Bouton2.addActionListener(this);
Bouton3.addActionListener(this);
         message = new JLabel ("Cliquez sur un des boutons");

// getContentPane().add (BorderLayout.NORTH, panel);
```

```
   getContentPane().add (message);
          getContentPane().add (BorderLayout.WEST, panel);
       pack();


}

     public void actionPerformed(ActionEvent e)
     {
      System.out.println(e.getActionCommand());
      message.setText(e.getActionCommand()); }
}

// Le main

public class Clic {


public static void main(String[] args) {

      Fenetre fenetre = new Fenetre();
     fenetre.setVisible(true);
}

}
```