# Introduction à la gestion des évènements en Java

En Java, il existe des boîtes de dialogue prédéfinies qui permettent des affichages simples (basés sur la classe **JOptionPane**), mais elles atteignent vite leurs limites dès que l'on souhaite réaliser des interfaces plus complexes. C'est pourquoi nous allons apprendre à définir nos propres fenêtres avec les composants que l'on désire.

## 1 Première fenêtre (la classe JFrame)

Comme constaté jusqu'à présent, l'exécution d'un programme Java entraîne automatiquement la création d'un fenêtre console.
Il est tout de même possible de réaliser une application en mode graphique, mais le programme devra entièrement créer et gérer cette fenêtre graphique, ce qui n'est pas simple.
Par soucis de simplicité, nous ne créerons qu'une seule fenêtre à la fois.

Pour créer une fenêtre graphique, il existe, dans un paquetage nommé **javax.swing**, une classe **JFrame** possédant un constructeur sans argument.

Exemple :
**JFrame fen = new JFrame();**

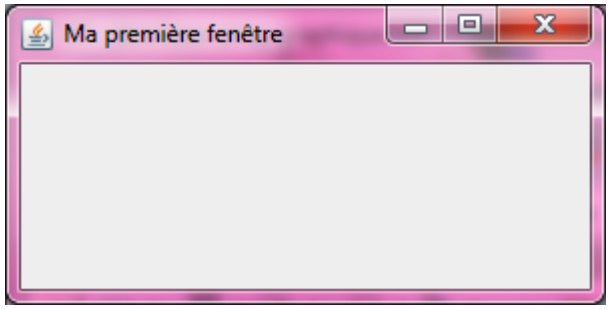
Si on se limite à cela rien n'apparaît à l'écran. Il faut donc ajouter :
**fen.setVisible(true);**

Par défaut, une telle fenêtre est créée avec une taille nulle, il faut donc définir ses dimensions :
**fen.setSize(300,150);**

Il est possible d'ajouter du texte dans la barre de titre :
**fen.setTitle("Ma premiere fenetre");**

Exemple de programme de création de fenêtre graphique :

```
import javax.swing.*;

public class PremFen {
 public static void main (String args[])  {
        JFrame fen=new JFrame();
        fen.setSize (300,150);
        fen.setTitle ("Ma première fenêtre");
        fen.setVisible (true);
 }
}
```




Remarque : Lorsque vous fermez la fenêtre, vous ne "récupérez la main" que lorsque vous faites **Ctrl+c** dans la console.

1. La méthode **setBounds** permet de fixer les dimension ainsi que la position de la fenêtre :
   fen.setBounds(10, 40, 300, 200); //le coin supérieur gauche de la fenêtre est placé au pixel de coordonnées //10, 40 et ses dimensions seront de 300*200 pixels
2. La méthode **setBackground** permet de modifier la couleur de fond de la fenêtre.
3. La méthode **getSize** permet d'obtenir la taille courante de la fenêtre.

## 2 Mise en place d'un premier composant : un bouton

```
import javax.swing.*;
import java.awt.*;

class FenetreBouton extends JFrame {
    JButton monBouton;

    public FenetreBouton()        {
                this.setTitle ("Affichage du bouton");
                this.setSize (300, 150);
            monBouton = new JButton ("mon premier bouton");
            //affiche les composants les après les autres sur une ligne
            this.getContentPane().setLayout(new FlowLayout());
            // ajoute le bouton au conteneur fenêtre
            this.getContentPane().add(monBouton);
    }
}
```

La méthode **getContentPane()** de la classe JFrame permet de définir la fenêtre comme étant le conteneur du bouton à faire apparaître.

On pourra utiliser cette fenêtre avec le programme suivant par exemple :

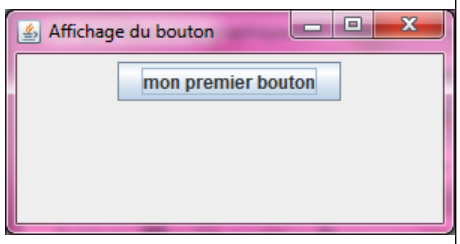
```
public class FenetreBoutonMain {
        public static void main (String args[])     {
                FenetreBouton fen = new FenetreBouton();
                fen.setVisible(true);
        }
}
```

**Attention les clics sur ce bouton ne sont pas encore pris en charge.**

Gérons donc maintenant le bouton avec un écouteur.

Pour cela, il faut créer un écouteur implémentant l'interface **ActionListener** (avec une seule méthode : **actionPerformed**) qui prend en charge l'ensemble des actions qui peuvent survenir.

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

class FenetreBouton2 extends JFrame implements ActionListener {
    JButton monBouton;

    public FenetreBouton2()  {
                setTitle ("Affichage du bouton");
                setSize (300, 150);
          monBouton = new JButton ("mon premier bouton");
           getContentPane().setLayout(new FlowLayout());
          getContentPane().add(monBouton);
              monBouton.addActionListener(this);
    }

    public  void actionPerformed (ActionEvent ev) {
            System.out.println("Je clique sur mon bouton");
    }
}
```




```
FenetreBouton2Main [Java Applicatio
Je clique sur mon bouton
Je clique sur mon bouton
Je clique sur mon bouton
Je clique sur mon bouton
Je clique sur mon bouton
```

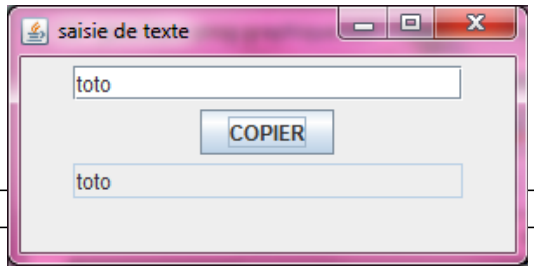
Il faut ensuite réécrire le main permettant l'exécution de cette fenêtre...

# 3 Mise en place de champs de texte

Un champ de texte est un objet de type **JTextField**. Son constructeur attend sa **taille** en nombre de caractères standard.
On peut connaître à tout moment l'information figurant dans un champ de texte en utilisant la méthode **getText()**.

Exemple d'utilisation de champ de texte :

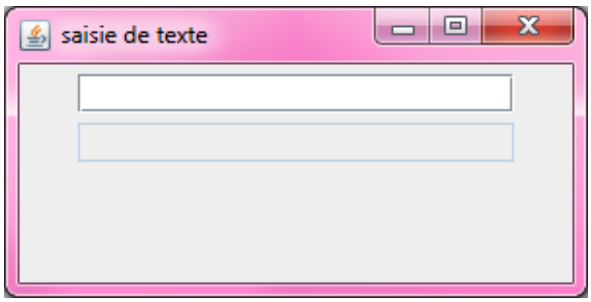
```
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;

class FenetreTexte extends JFrame implements ActionListener {

     private JTextField  saisie, copie;
     private JButton bouton;

     public FenetreTexte()     {
                setTitle ("saisie de texte");
                setSize (300, 150);
           Container contenu = getContentPane();
                contenu.setLayout (new FlowLayout());

                saisie = new JTextField (20);
                contenu.add(saisie);

                bouton = new JButton ("COPIER");
                contenu.add(bouton);
                bouton.addActionListener(this);

                copie = new JTextField (20);
                copie.setEditable(false); // le champ n'est plus modifiable
                contenu.add(copie);
     }

     public  void actionPerformed (ActionEvent e)  {
           // on regarde quelle est la source de l'événement
                if (e.getSource() == bouton)  {
                        String texte = saisie.getText();
                     copie.setText(texte);
           }
     }
}
```




```
public class FenetreTexteMain {
        public static void main (String args[])     {
                FenetreTexte fen = new FenetreTexte();
                fen.setVisible(true);
        }
}
```

Il est également possible de gérer le focus ou la perte de focus sur un champ de saisie :

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

class FenetreTexteFocus extends JFrame implements ActionListener, FocusListener {
     private JTextField  saisie, copie;
     private JButton bouton;

     public FenetreTexteFocus()        {
                setTitle ("saisie de texte");
                setSize (300, 150);
           Container contenu = getContentPane();
                contenu.setLayout (new FlowLayout());

                saisie = new JTextField (20);
                contenu.add(saisie);
                saisie.addActionListener(this);
                saisie.addFocusListener(this);
                copie = new JTextField (20);
                copie.setEditable(false);
                contenu.add(copie);
     }

     public  void actionPerformed (ActionEvent e)    {
                System.out.println("validation saisie");
                String texte = saisie.getText();
                copie.setText(texte);
     }

     public  void focusLost (FocusEvent e)  {
                System.out.println("perte de focus");
                String texte = saisie.getText();
                copie.setText(texte);
     }

     public  void focusGained (FocusEvent e)  {
                System.out.println("focus sur la saisie");
     }
}
```




```
public class FenetreTexteFocusMain {
        public static void main (String args[])     {
                FenetreTexteFocus fen = new FenetreTexteFocus();
                fen.setVisible(true);
        }
}
```

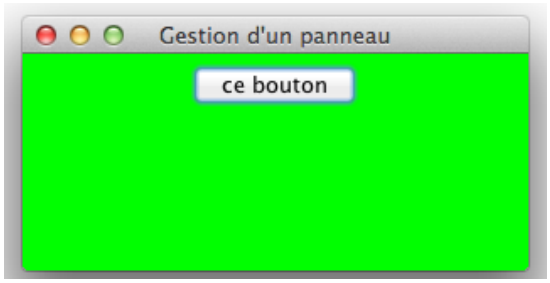
Il existe bien d'autres méthode associées aux champs de texte, par exemple (ajouter **javax.swing.event.***) : s**etBackground**(Color.blue).

# 4 Mise en place d'un panneau

Jusqu'ici, nous avons placé des composants dans une fenêtre JFrame qui sert de conteneur.
Il existe cependant des conteneurs intermédiaires qui pourront eux-mêmes contenir d'autres composants. C'est le cas des panneaux qui sont des sortes de sous-fenêtres sans titre ni bordure. Ils appartiennent à la classe **JPanel**. Ils ont d'autres propriétés que les JFrame.

Mise en place d'un panneau contenant un bouton dans une fenêtre :

```
import javax.swing.*;
import java.awt.*;

class FenetrePanneau2 extends JFrame {
 private JPanel lePanneau;


 public FenetrePanneau2()  {
        setTitle ("Gestion d'un panneau");
        setSize (300, 150);
        lePanneau = new JPanel();
        lePanneau.setBackground(Color.green);
        getContentPane().add(lePanneau);
        JButton bouton = new JButton("ce bouton");
        lePanneau.add(bouton); // rmq : on ajoute le bouton après et ça marche !
 }
}
```

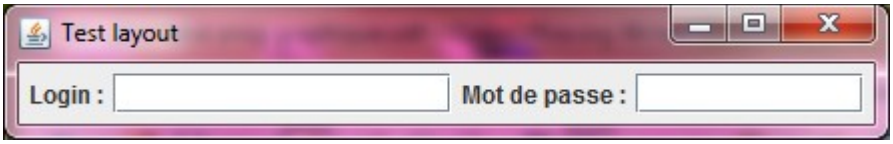



```
public class FenetrePanneau2Main  {
        public static void main (String args[])     {
                FenetrePanneau2 fen = new FenetrePanneau2();
                fen.setVisible(true);
        }
 }
```

# 5 Mise en page des composants avec les layouts

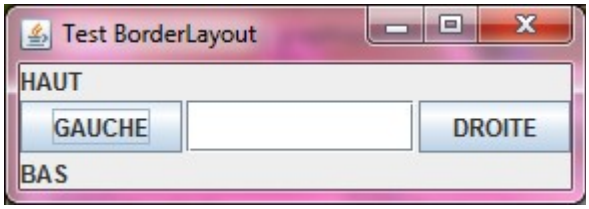
Chaque conteneur détermine la position des composants qu'il contient grâce à son gestionnaire de mise en page associé (layout).
Il est souvent nécessaire de changer la disposition grâce à la méthode setLayout.
Le layout passé en paramètre à cette méthode est généralement une instance d'une des classes **FlowLayout**, **BorderLayout** ou **GridLayout** du paquetage java.awt.

Une fois le layout du panneau intérieur d'une fenêtre déterminé, chacun des composants est ajouté à ce panneau au moyen d'une des méthodes add disponibles.



java.awt.FlowLayout

java.awt.GridLayout



java.awt.BorderLayout

**Remarque :**
Il existe bien d'autres contrôles que ceux abordés ici, comme :

- les cases à cocher,
- les boutons radios,
- les étiquettes,
- les boîtes de listes,
- les boîtes combinées
- les barres d'outils.

***Force est de constater qu'il n'est pas simple de mettre en place les interfaces graphiques en Java si on les construit entièrement manuellement.***
***Fort heureusement, des IDE, NetBeans notamment, sont là pour nous faciliter la vie.***

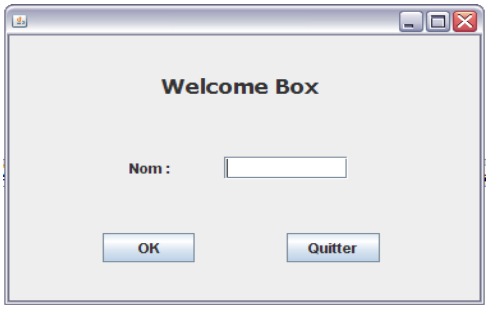
# Découverte des interfaces graphiques Java avec NETBEANS

**A. Prise en main** :

1. Consulter : http://netbeans.org/kb/60/java/gui-functionality.html
2. Dans NetBeans, vous allez créer un projet nommé **votreNomDeFamille-decouverte**.
3. Suivre les instructions de la page ci-dessus pour construire votre première application selon les différentes étapes indiquées :
   - Exercice 1 : créer le projet
   - Exercice 2 : construire l'interface
   - Exercice 3 : ajouter des fonctionnalités
   - Exercice 4 : lancer le programme
   - La gestion des évènements.
4. Déposer sur Chamilo le dossier contenant votre projet (**votreNomDeFamille-decouverte**).

**B. Entraînement** :

5. Consulter : http://imss-www.upmf-grenoble.fr/prevert/Prog/Java/swing/JOptionPane.html
6. Dans NetBeans, vous allez créer un nouveau projet nommé **votreNomDeFamille-welcome**.
7. Construire l'écran suivant :




8. Lorsque l'utilisateur saisit un nom est clique sur **OK,** une nouvelle fenêtre JoptionPane.showMessageDialog s'affiche.


Lorsque l'utilisateur clique sur **Quitter** de la fenêtre « Welcome box », cette dernière se ferme.

Attention, vous utiliserez les conventions de nommage suivantes :
- Bouton : btnXXX
- Zone de texte : txtXXX
- Frame : frmXXX
- etc.

9. Déposer sur Chamilo le dossier contenant votre projet (**votreNomDeFamille-welcome**).