



## Tutoriel développement Java - Swing

---

## Introduction

- **Introduction**

Dans ce tutoriel, vous découvrirez comment mettre en œuvre une interface graphique Java à l'aide de AWT et Swing.

Pour cela, nous utiliserons l'outil **WindowBuilder** sous Eclipse.

# 1

## Présentation

- **Historique**

Dès ses premières versions Java comprenait une bibliothèque de création d'interfaces utilisateurs graphiques - **GUI** (Graphic User Interface).

Cette bibliothèque nommée **AWT** (Abstract Window Toolkit) devait permettre d'obtenir des interfaces graphiques identiques sur tous les systèmes.

## 1

### Présentation

- **Historique**

AWT utilisait une approche par **peers**. Les composants utilisés par AWT étaient associés à leurs équivalents dans le système d'exploitation (on parle de ***composants lourds***).

Par exemple, pour construire un bouton, AWT appelait le composant bouton du système d'exploitation (donc un bouton Windows sous Windows, un bouton Mac sur un Mac, . . .).

Pour chaque composant, il y avait donc une couche d'adaptation entre Java et le système d'exploitation.

# 1

## Présentation

- **Historique**

### Problème :

- compatibilité => certains composants se comportent différemment d'un système à l'autre.
- Il manque des éléments importants.

### Solution :

- nouvelle API reprenant les éléments efficaces de **AWT** : refonte de la bibliothèque et nouveau nom **Swing**.

## 1

### Présentation

- **Historique**

Parmi les modifications importantes, les composants sont devenus des éléments de la **JVM** (ce sont des ***composants légers***).

Dans une application **Swing**, seule la fenêtre principale est construite par le système d'exploitation, tous les autres composants sont dessinés par la JVM.

Contrairement à la plupart des interfaces graphiques, Swing est directement liée au langage Java et ne peut être utilisé qu'avec lui.

## 2

### Composants

- **Composants**

Les éléments constituant les interfaces graphiques sont appelés composants : boutons, textes, images, fenêtres,...

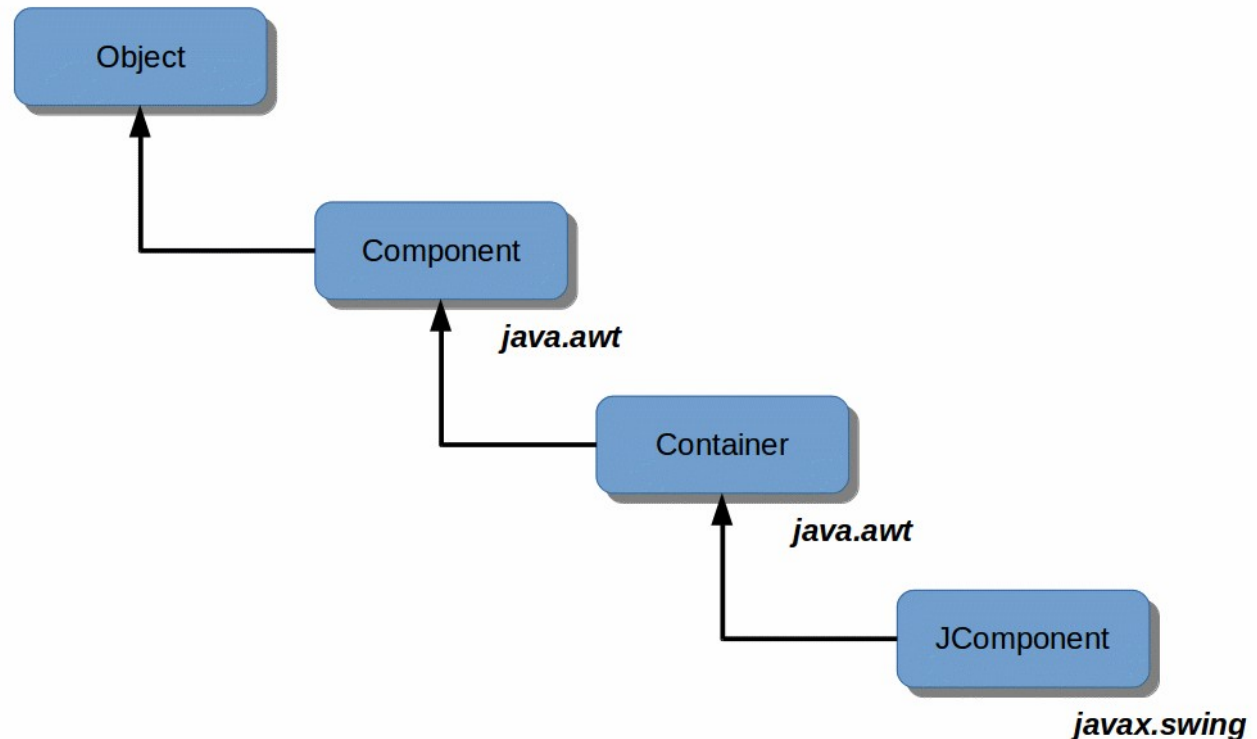
Tous les composants de **Swing** descendent de la classe mère **JComponent**, qui descend elle-même de la classe **Component** de **AWT**.

## 2

### Composants

- **Composants**

Cette hiérarchie permet de proposer de nombreuses méthodes communes à tous les éléments.





## 2

### Composants

- **Conteneurs**

Tous les composants sont hébergés par un conteneur (sauf les conteneurs primaires).

Il est possible de connaître le conteneur d'un composant en utilisant la méthode ***getParent()***.

On ajoute un composant dans un conteneur en utilisant la méthode ***add()***.

Les conteneurs sont des descendants de la classe **Container**.

## 2

### Composants

- **Gestionnaires de placement**

Le placement des composants est souvent confié à un gestionnaire de placement :

- positionnement absolu,
- positionnement relatif,
- ...

Swing propose de nombreux gestionnaires de placement pour construire des applications.

## 2

### Composants

- **Gestionnaires d'événements**

Les actions de l'utilisateur sont représentées par des événements.

Les événements sont des objets qui sont transmis par un composant vers un gestionnaire d'événements.

A noter : la gestion des événements est séparée de la création de l'interface ce qui facilite les modifications éventuelles du programme.

## 2

### Composants

- **Les fenêtres : JWindow, JFrame**

Swing propose deux types de fenêtres : les **JWindow** et les **JFrame**.

Ces deux composants sont proches, ils descendent tous les deux de **Window** (composant de **AWT**) et ont donc de nombreux points communs.

Pour créer un programme exécutable en mode graphique, il suffit de créer une (ou plusieurs) fenêtre(s) dans la classe qui contient la méthode statique main.

## 2

### Composants

- **Les fenêtres : JWindow, JFrame**

La classe **JWindow** permet de créer une fenêtre graphique dans le système de fenêtrage utilisée.

Cette fenêtre n'a aucune bordure et aucun bouton. Elle ne peut être fermée que par le programme qui l'a construite (ou en mettant fin à l'application via le système d'exploitation en tuant le processus associé).

## 2

### Composants

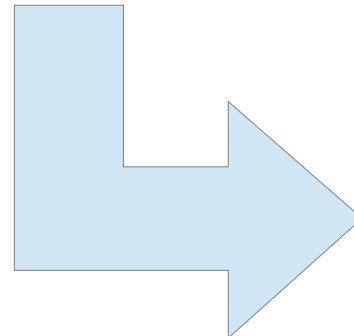
- **Les fenêtres : JWindow, JFrame**

```
import javax.swing.JWindow;
```

```
public class Main {
```

```
    public static void main(String[] args) {  
        // TODO Auto-generated method stub  
        JWindow fenetre = new JWindow () ;  
        fenetre . setSize (300 ,300) ;  
        fenetre . setLocation (500 ,500) ;  
        fenetre . setVisible( true );  
        fenetre.dispose();  
    }
```

```
}
```



## 2

### Composants

- **Les fenêtres : JWindow, JFrame**

**JFrame** construit des fenêtres qui comportent une bordure, un titre, des icônes et éventuellement un menu.

La plupart des applications sont construites à partir d'une (ou plusieurs) JFrame.

## 2

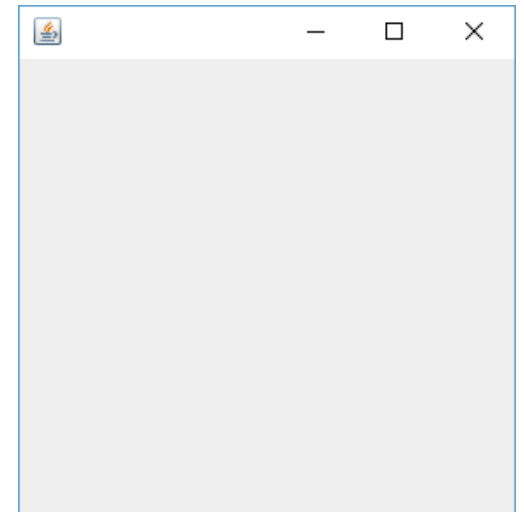
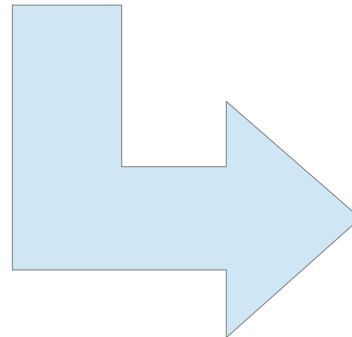
### Composants

- **Les fenêtres : JWindow, JFrame**

```
import javax.swing.JFrame;

public class Main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        JFrame fenetre = new JFrame () ;
        fenetre . setSize (300 ,300) ;
        fenetre . setLocation (500 ,500) ;
        fenetre . setVisible( true );
    }
}
```





## 2

### Composants

- **Fermeture des fenêtres : JWindow, JFrame**

Lorsqu'une fenêtre n'est plus utile, elle peut être caché (avec **setVisible(false)**) ou détruite.

Si cette fenêtre doit être réaffichée rapidement le mieux est de la cacher.

En revanche, si elle n'est plus utile il convient de libérer les ressources associée avec la méthode **dispose()**.

## 2

### Composants

- **Fermeture des fenêtres : JWindow, JFrame**

#### ***dispose()***

Cette méthode libère les ressources allouées par le système d'exploitation, puis le garbage collector détruit la fenêtre.

Quand il n'y a plus aucune fenêtre dans une application elle est quittée.

## 2

### Composants

- **Fermeture des fenêtres : JWindow, JFrame**

#### ***dispose()***

Cette méthode libère les ressources alloués par le système d'exploitation, puis le garbage collector détruit la fenêtre.

Quand il n'y a plus aucune fenêtre dans une application elle est quittée.

## 2

### Composants

- **JPanel**

**Jpanel** est un composant de type conteneur qui permet de mettre d'autres objets de même type ou des objets de type composant (boutons, cases à cocher...).

### Utilisation :

- Instanciez un JPanel (import de la classe `javax.swing.JPanel`)
- Indiquez que le Jpanel sera le « content pan » du JFrame

## 2

### Composants

- **JPanel**

```
import java.awt.Color;

import javax.swing.JFrame;
import javax.swing.JPanel;

public class Main {

    public static void main(String[] args) {
        JFrame fenetre = new JFrame () ;
        fenetre . setSize (300 ,300) ;
        fenetre . setLocation (500 ,500) ;

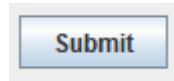
        //Instancier d'un objet JPanel
        JPanel panel = new JPanel();
        //Définition de sa couleur de fond
        panel.setBackground(Color.GRAY);
        //Indiquer au JFrame que le JPanel sera son content pane
        fenetre.setContentPane(panel);
        fenetre.setVisible(true);
    }
}
```

## 2

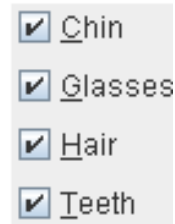
### Composants

- **Jpanel**

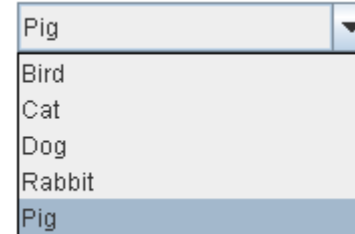
Liste des composants (non exhaustive):



JButton



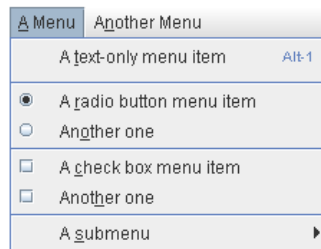
JCheckBox



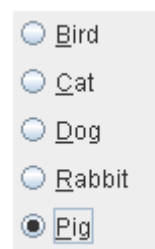
JComboBox



JList



JMenu



JRadioButton



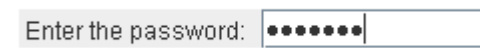
JSlider



JSpinner



JTextField



JPasswordField

## 2

### Composants

- **JPanel**

## Exemple de l'appel d'une fenêtre JPanel

```
import javax.swing.JFrame;

public class Main {

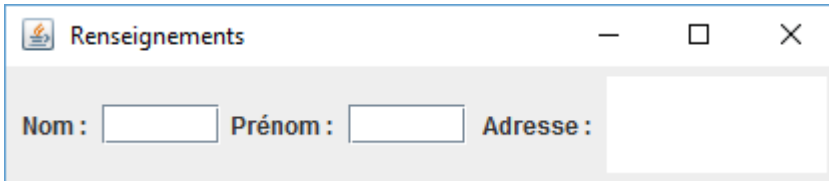
    public static void main(String[] args) {
        JFrame fen = new JFrame () ;
        FicheIdentite laFiche = new FicheIdentite () ;
        fen . setContentPane( laFiche ) ;
        fen . pack () ;
        fen . setVisible( true );
        fen . setTitle ( " Renseignements" ) ;

    }
}
```

## 2

### Composants

#### • JPanel



```
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextArea;
import javax.swing.JTextField;
```

```
public class FicheIdentite extends JPanel {
    private JTextField nom ;
    private JLabel labelNom ;
    private JTextField prenom ;
    private JLabel labelPrenom;
    private JTextArea adresse ;
    private JLabel labelAdresse;

    public FicheIdentite () {
        super () ;
        labelNom = new JLabel (" Nom : " ) ;
        labelPrenom = new JLabel ("Prénom : " ) ;
        labelAdresse = new JLabel (" Adresse : " );
        nom = new JTextField (5) ;
        prenom = new JTextField (5) ;
        adresse = new JTextArea( "" ,3 ,10) ;
        this . add ( labelNom );
        this . add ( nom ) ;
        this . add ( labelPrenom);
        this . add ( prenom ) ;
        this . add ( labelAdresse);
        this . add ( adresse );
    };
}
```



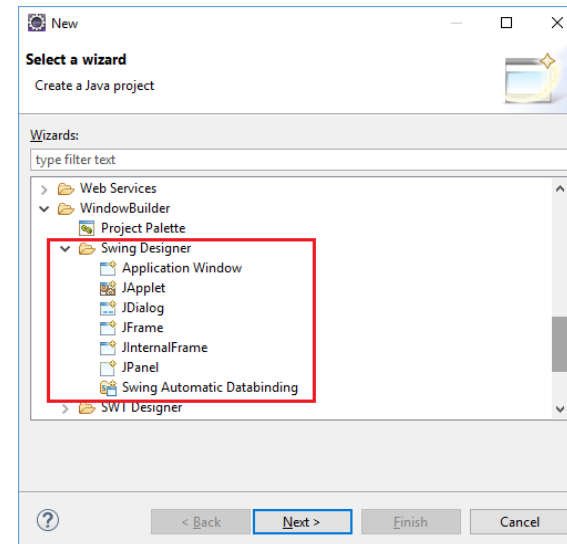
# 3

## WindowBuilder

- **Windows Builder**

Il est possible d'utiliser **WindowBuilder** pour réaliser les interfaces graphiques.

Après avoir créé un projet Java, cliquez avec le bouton droit de la souris, puis sélectionnez New → Other...

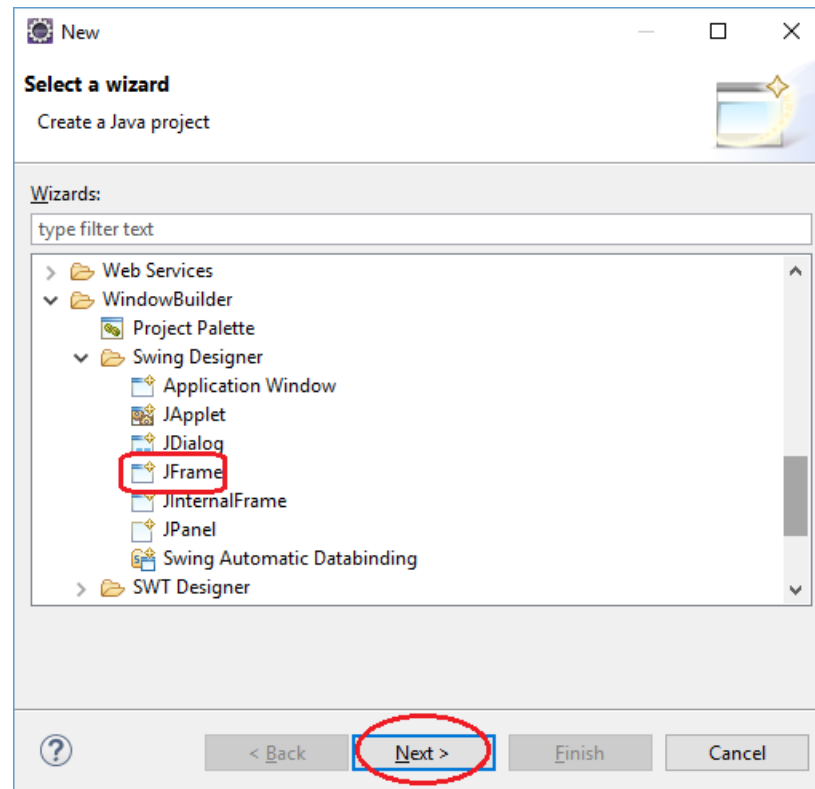


# 3

## WindowBuilder

- **Windows Builder**

Allez dans WindowBuilder, puis Swing Designer et sélectionnez JFrame. Cliquez sur Next.



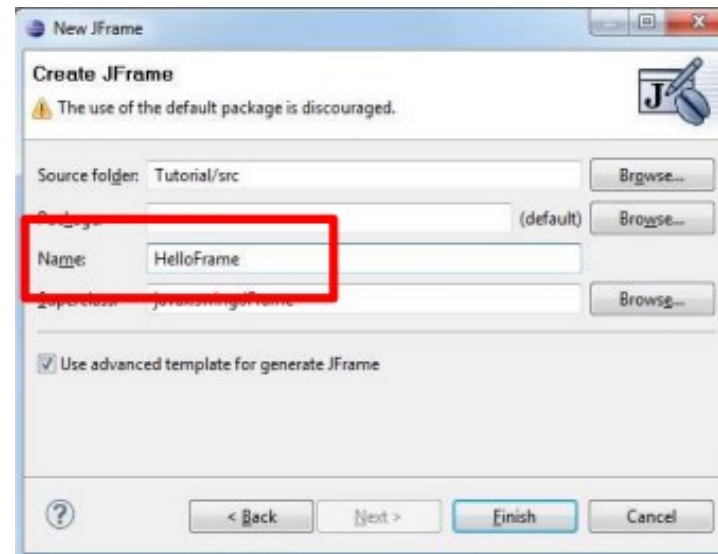
# 3

## WindowBuilder

- **Windows Builder**

Donnez un nom à cette nouvelle classe, qui héritera de JFrame (ici: HelloFrame).

Cliquez sur Finish



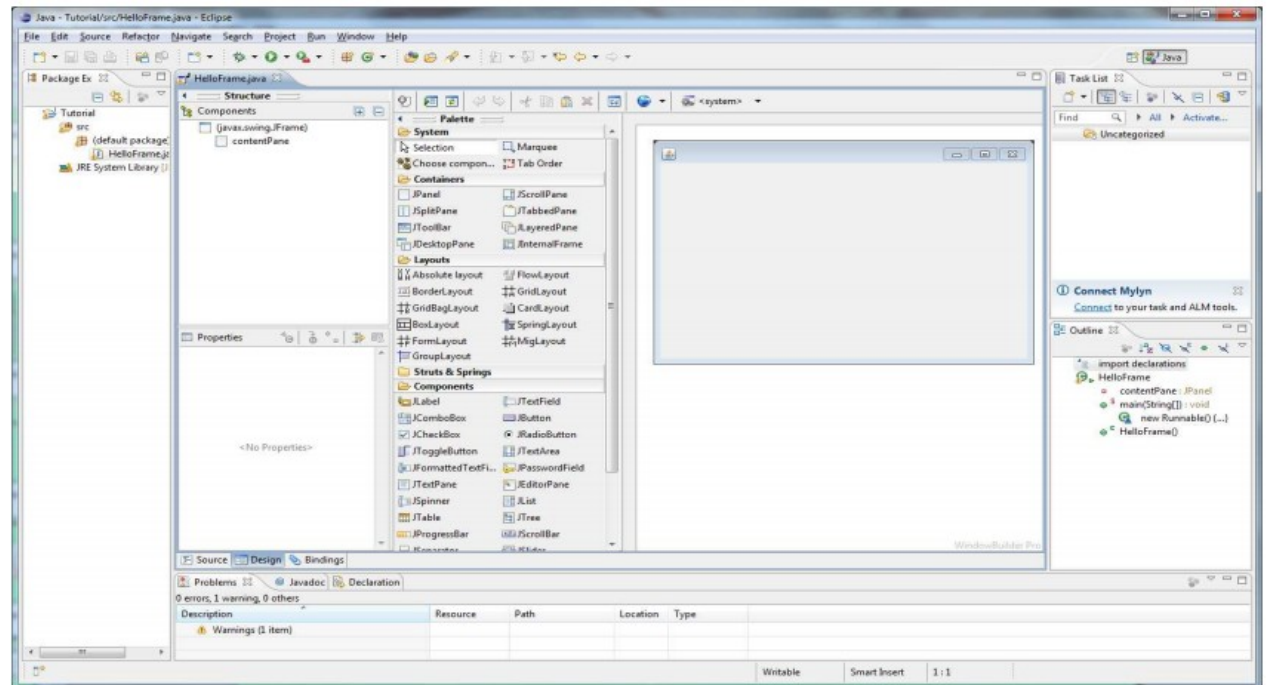
**3**

WindowBuilder

- **Windows Builder**

On obtient le code de la classe HelloFrame.

Choisissez la vue Design pour réaliser l'interface:



## 3

### WindowBuilder

- **Windows Builder**

Voici le code  
généré :

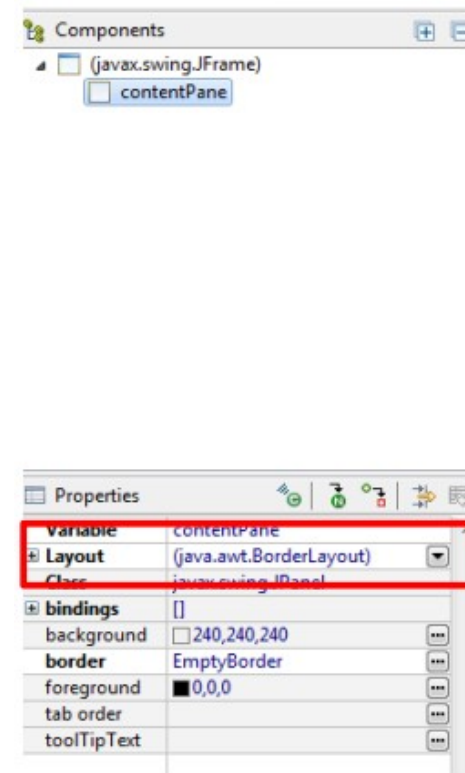
```
public class HelloFrame extends JFrame {  
  
    private JPanel contentPane;  
  
    /**  
     * Launch the application.  
     */  
    public static void main(String[] args) {  
        EventQueue.invokeLater(new Runnable() {  
            public void run() {  
                try {  
                    HelloFrame frame = new HelloFrame();  
                    frame.setVisible(true);  
                } catch (Exception e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
    }  
  
    /**  
     * Create the frame.  
     */  
    public HelloFrame() {  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        setBounds(100, 100, 450, 300);  
        contentPane = new JPanel();  
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));  
        contentPane.setLayout(new BorderLayout(0, 0));  
        setContentPane(contentPane);  
    }  
}
```

# 3

## WindowBuilder

- **Windows Builder**

Cliquez sur le panel `contentPane` et vérifiez dans les propriétés que le layout est de type `BorderLayout`

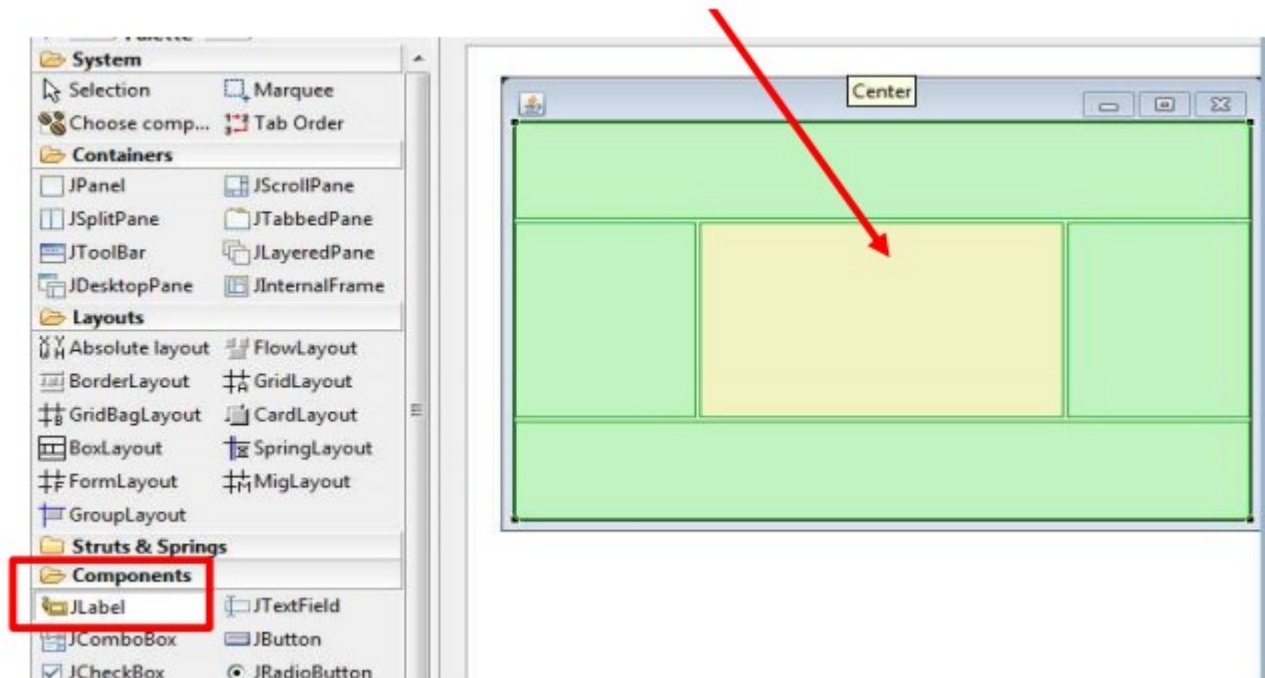


# 3

## WindowBuilder

- **Windows Builder**

Choisissez le composant JLabel et posez le sur la partie CENTER du contentPane

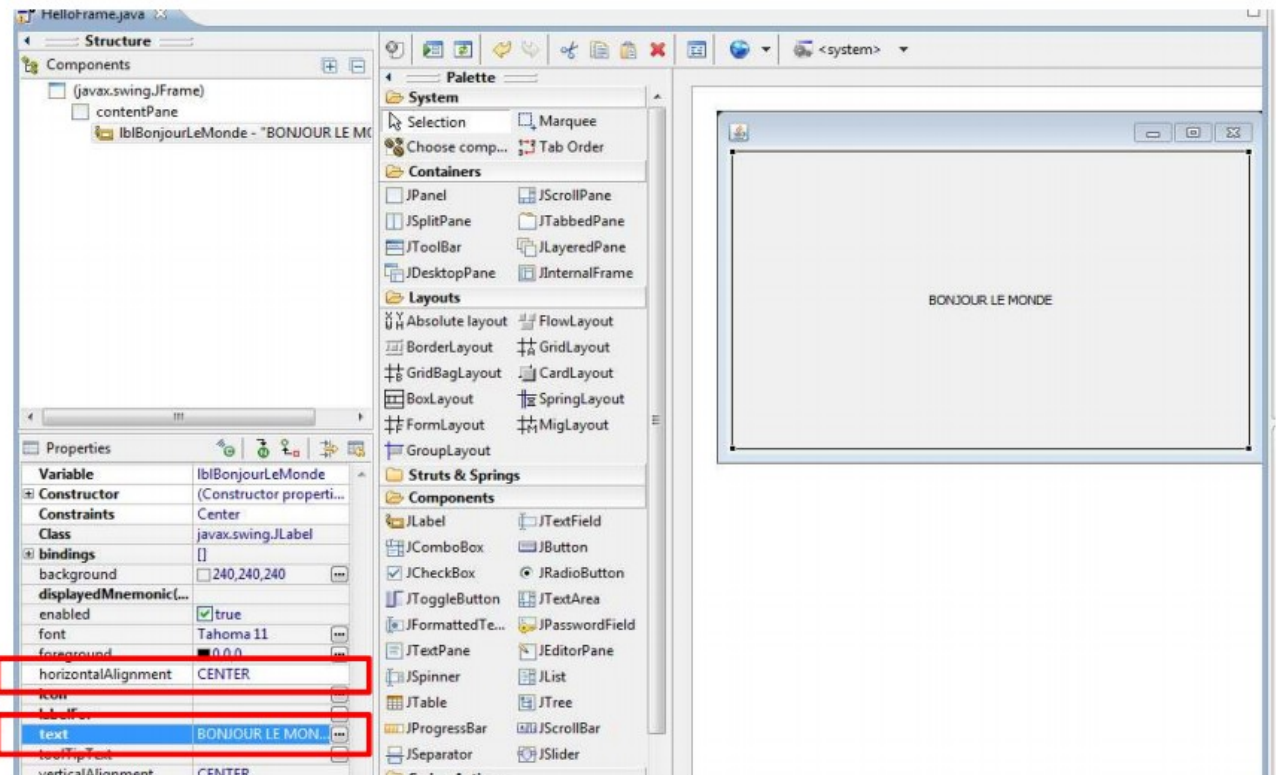


# 3

## WindowBuilder

- **Windows Builder**

Dans la boîte de propriétés du JLabel, modifiez les propriétés *horizontalAlignment* et *text*



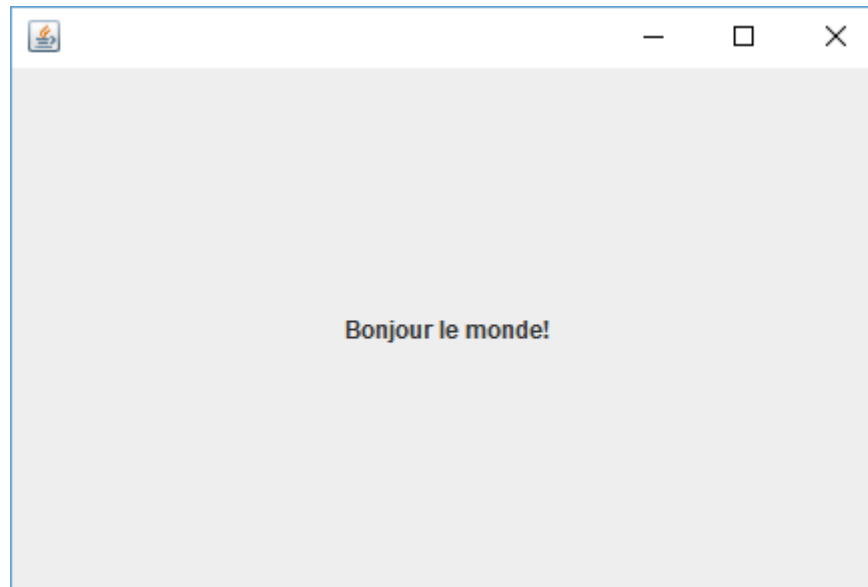
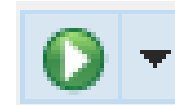


# 3

## WindowBuilder

- **Windows Builder**

Lancez l'application en cliquant sur

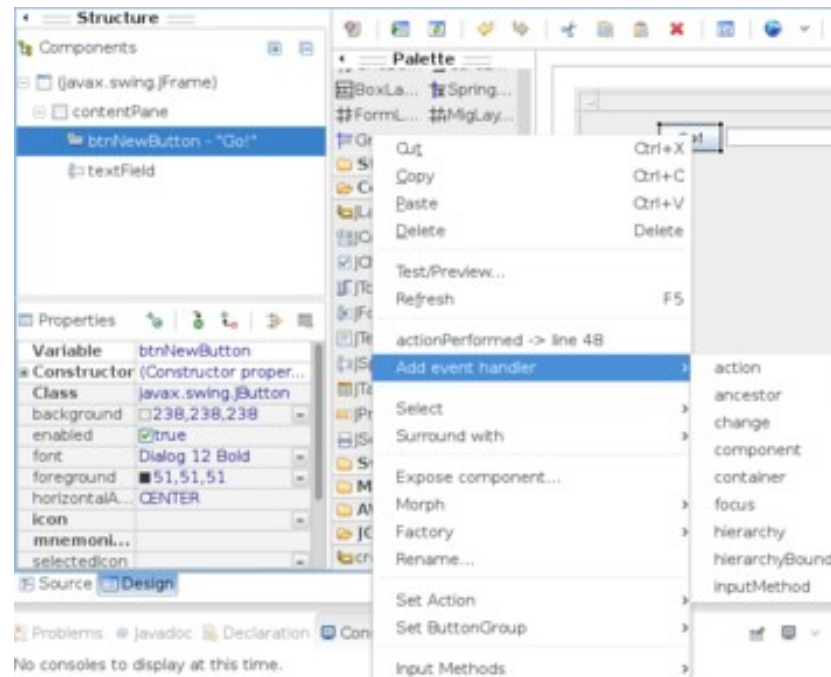


# 4

## Listener

### • Ajout d'un listener

Pour ajouter un ***listener*** sur un composant, il suffit de cliquer-droit sur le composant et d'utiliser le menu « Add event handler »:





Fin du tutoriel Java

---