

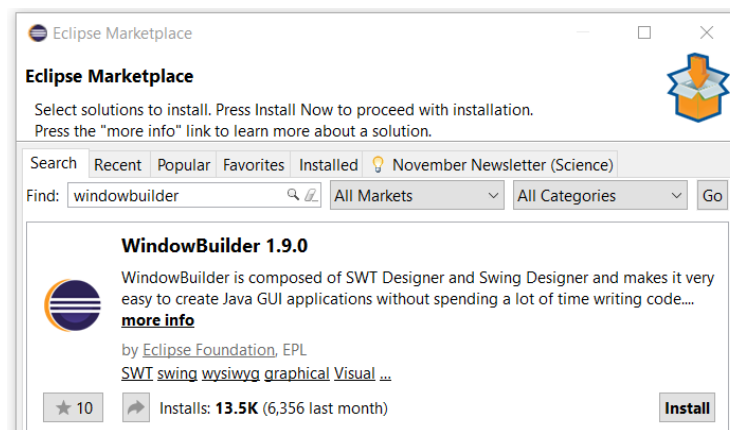
# Programmation Orientée Objet : JAVA

## TP7

WindowBuilder est un outil de conception d'applications Java GUI regroupant à la fois les composants AWT et SWING. La création de la fenêtre se fait simplement en WYSIWYG "what you see is what you get", le code java est généré automatiquement.

### 1. Installation de WindowBuilder

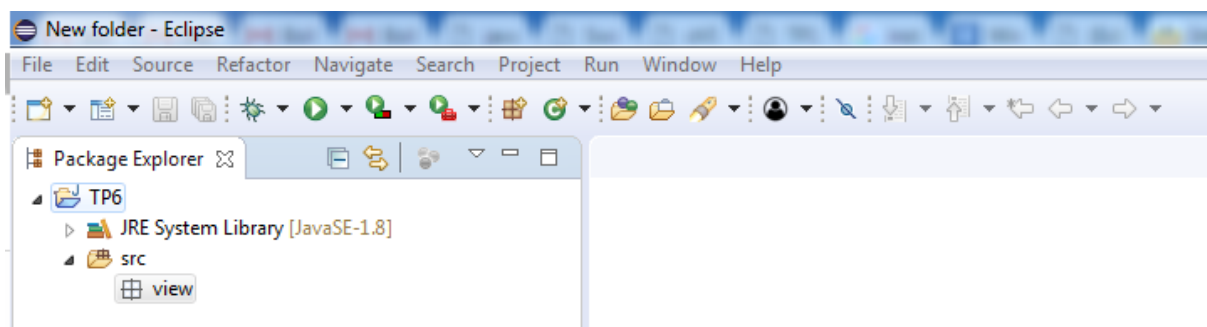
Pour installer le plugin **WindowBuilder** il suffit d'aller à [Help > Eclipse Marketplace...](#), tapez "windowbuilder" dans le champ [Find](#), puis cliquez sur [Go](#).



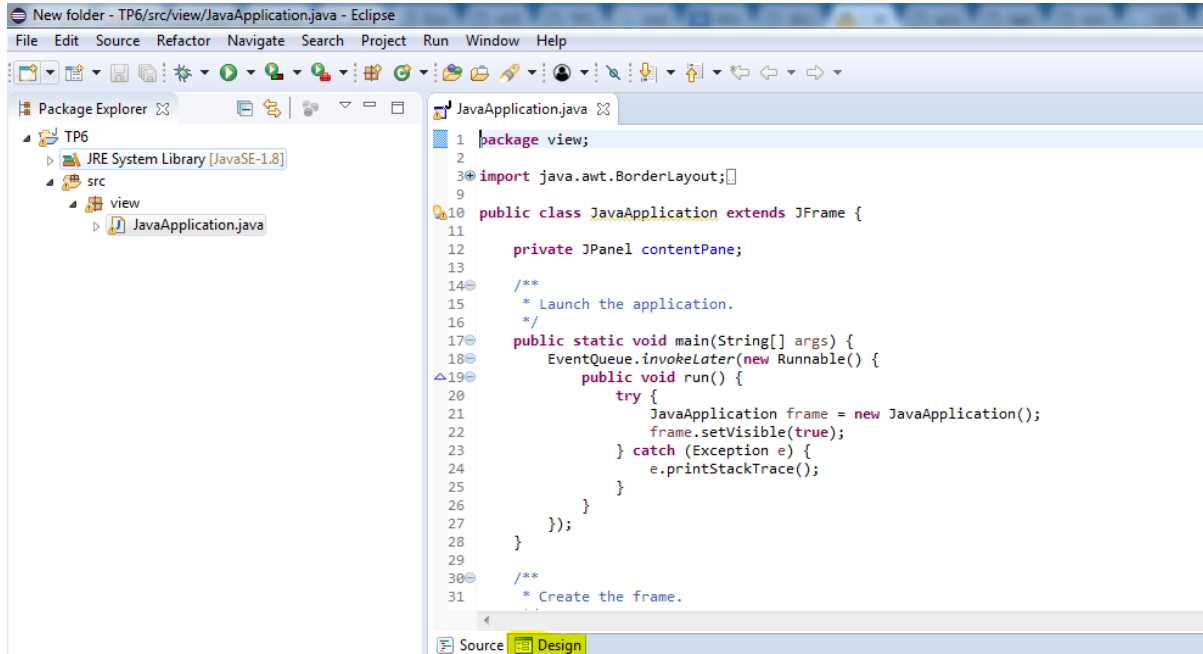
Ensuite cliquez sur [Install](#) et suivez les instructions jusqu'à ce qu'on vous demande de redémarrer Eclipse. Après un bon petit restart de votre Eclipse vous allez enfin pouvoir utiliser le plugin dans vos projets.

### 2. Création d'un projet

On suppose que l'on part d'un projet Java créé classiquement dans Eclipse avec un paquetage `view` disponible:



Cliquez-droit ensuite sur le paquetage, choisir **New**, puis **Other...**, **WindowBuilder**, **SwingDesigner** et enfin **JFrame**. Donnez un nom à votre classe et cliquez sur **Finish**. Vous obtenez alors une classe déjà préparée :

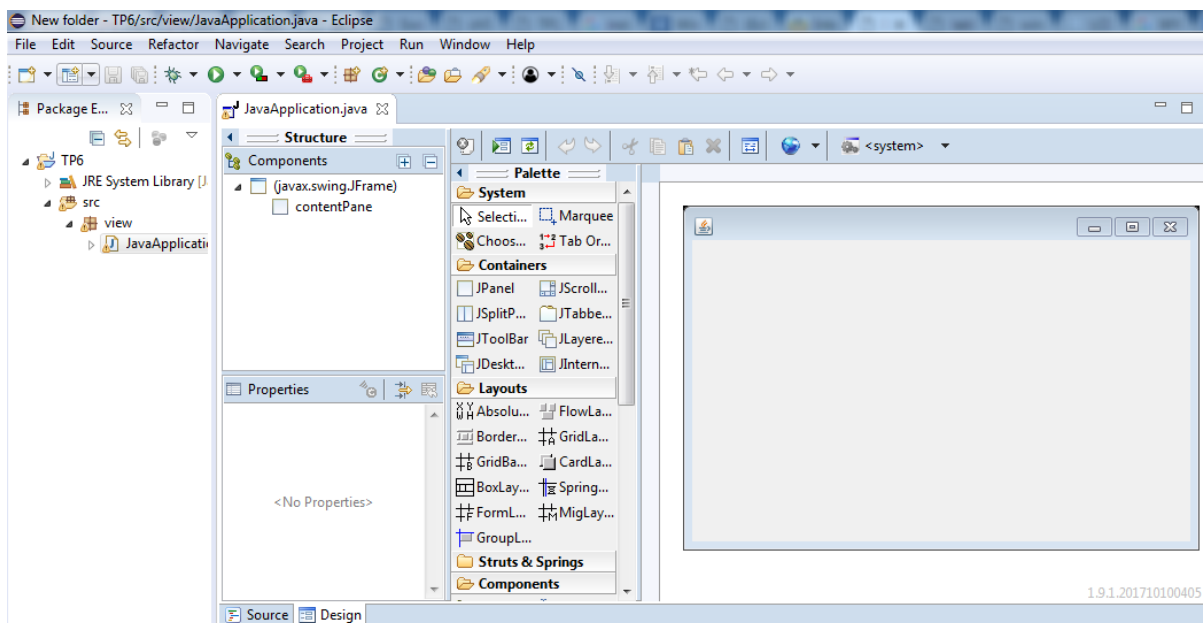


```

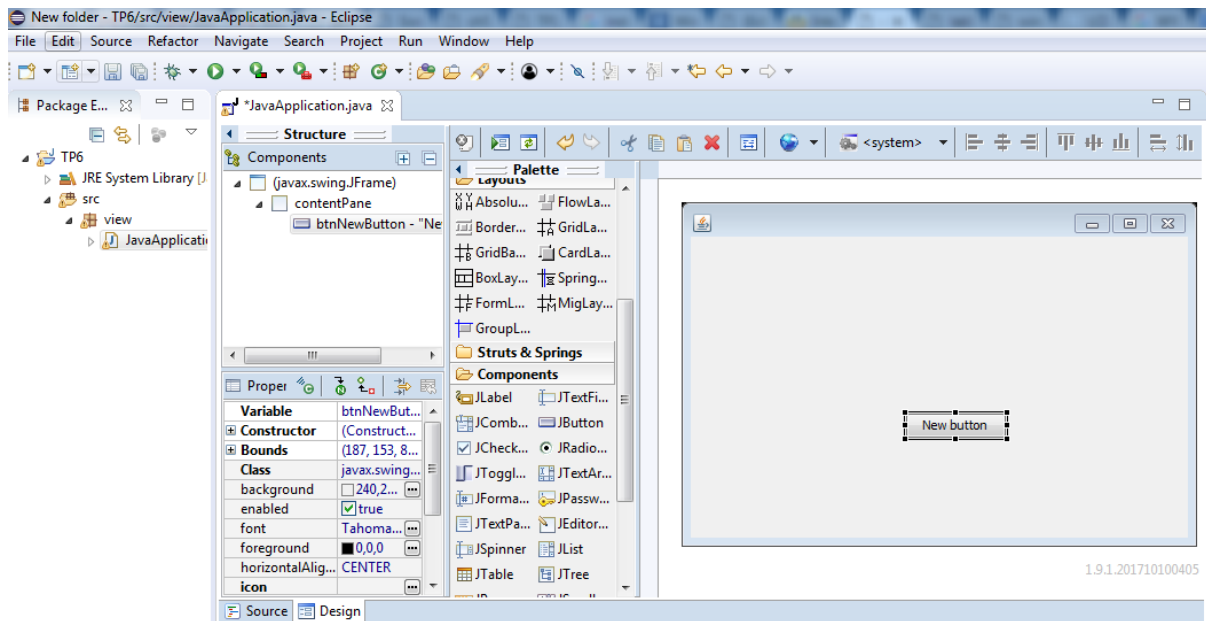
1 package view;
2
3 import java.awt.BorderLayout;
4
5
6
7
8
9
10 public class JavaApplication extends JFrame {
11
12     private JPanel contentPane;
13
14     /**
15      * Launch the application.
16      */
17     public static void main(String[] args) {
18         EventQueue.invokeLater(new Runnable() {
19             public void run() {
20                 try {
21                     JavaApplication frame = new JavaApplication();
22                     frame.setVisible(true);
23                 } catch (Exception e) {
24                     e.printStackTrace();
25                 }
26             }
27         });
28     }
29
30     /**
31      * Create the frame.

```

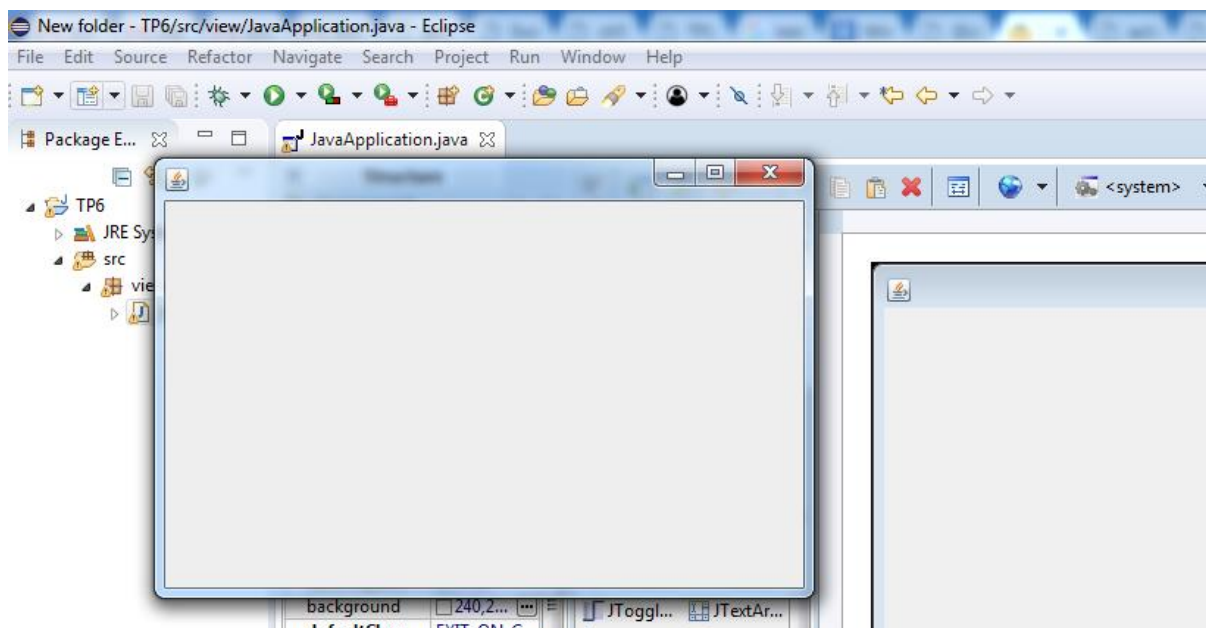
Vous remarquerez un onglet **Design** à côté d'un onglet **Source** en dessous de votre code. En cliquant sur cet onglet, vous passez sur le modèleur graphique et vous allez pouvoir créer votre vue à partir des palettes de composants disponibles :



Vous pouvez alors glisser-déposer des composants dans l'interface et vous pourrez remarquer que le code source est modifié en conséquence. Lorsque vous ajoutez ou cliquez sur un composant, un onglet contenant les propriétés du composant s'ouvre et vous pouvez les modifier :



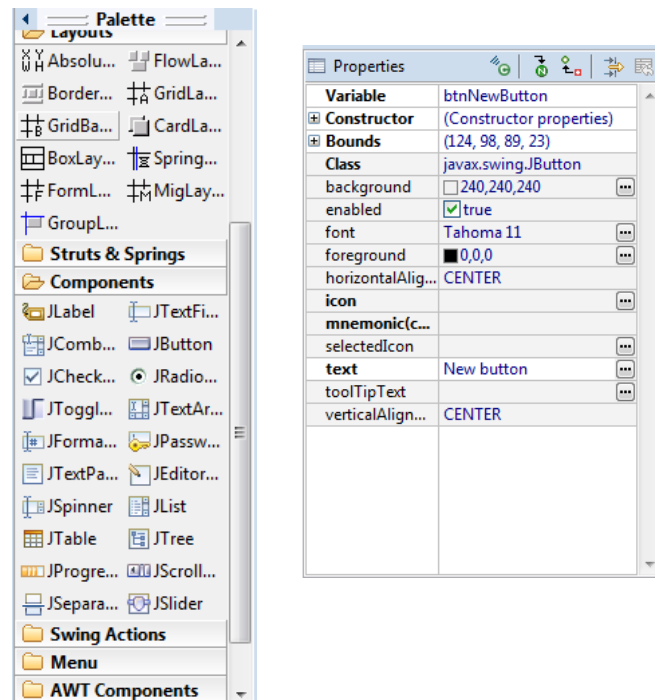
En cliquant sur **Run**, Eclipse vous demandera de sélectionner la classe principale. En validant la classe principale on obtient comme résultat une fenêtre vide (sans nom, sans bouton....).



### 3. Les éléments de base d'une fenêtre

Tous les objets graphiques nécessaires à une interface sont regroupés dans l'onglet **Palette**.

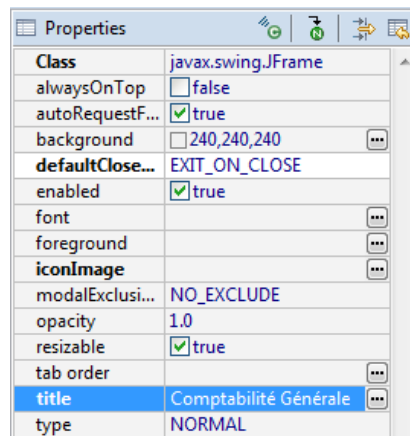
Lorsqu'un objet graphique est sélectionné à la souris, l'onglet **Properties** permet de manipuler ses propriétés (couleur, position, forme,...).



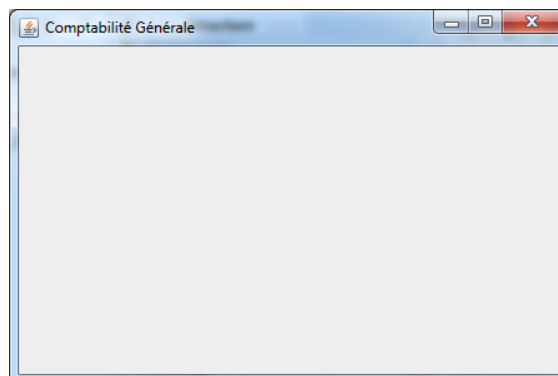
### 3.1. Renseigner les éléments de base de la fenêtre

Afin d'illustrer l'utilisation de l'onglet `Properties`, nous allons donner un nom à cette fenêtre en l'appelant : Comptabilité Générale.

Pour cela, dans l'onglet `Properties`, renseignez le champ : `Title`



Ceci devrait vous donner après compilation :



### 3.2. Le positionnement des composants

Avant d'apprendre à ajouter des composants dans notre `JPanel`, on va voir comment Java permet de positionner les composants à l'intérieur d'un container.

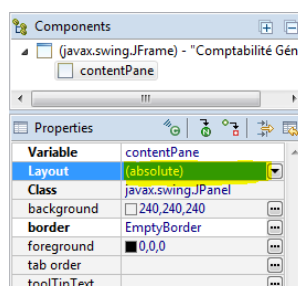
Il y a deux manières de faire ceci :

- Le positionnement absolu (`absolute layout`) : On va placer nos composants nous-même en utilisant un système de coordonnées en x et en y. Ce placement qui semble à première vue très simple, n'est en fait pas pratique du tout. Tout d'abord, si on veut ajouter des composants, il faut souvent changer une grande partie des coordonnées des autres composants et ensuite, il est quasi-impossible de faire du contenu redimensionnable avec cette méthode à moins de recalculer les coordonnées de tous les composants à chaque redimensionnement.
- L'utilisation d'un gestionnaire de placement : On peut aussi utiliser les gestionnaires de placements (`layout`) qui vont s'occuper de placer correctement nos composants dans la fenêtre en fonction des paramètres qu'on leur a donnés pour certains et bien évidemment en fonction des composants eux-mêmes. Elle se révèle bien plus souple et bien plus pratique que le positionnement absolu.

Il existe de nombreux gestionnaires de `layout` dans Swing, en voici quelques-uns :

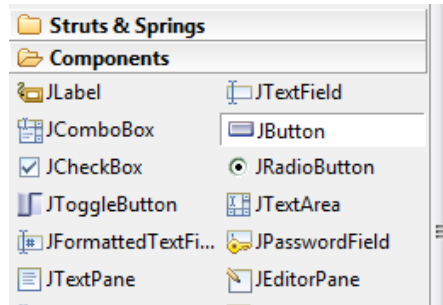
- `FlowLayout` : C'est le layout par défaut des panels dans Swing. Il place les composants sur une ligne et recommence une nouvelle ligne à chaque fois qu'il atteint la fin de la ligne.
- `BorderLayout` : Ce layout place les composants dans 5 zones du container : La zone du haut, la zone du bas, la zone de gauche, celle de droite et la zone du centre. Il est très utile pour gérer le positionnement du contentpane.
- `CardLayout` : Ce layout place les composants sur des couches disposées les unes sur les autres. On ne peut voir qu'une couche à la fois. On utilise surtout ce layout quand on a une série de composants qui s'affichent en fonction de quelques choses (liste déroulante, boutons, ...).
- `GridLayout` : Ce composant place les composants dans une grille. Il va redimensionner les composants pour les rendre tous de la même taille. C'est surtout utile quand on a plusieurs boutons ou champs texte en colonne et ligne qui doivent avoir la même taille, par exemple, les boutons d'une calculatrice.
- `GridBagLayout` : C'est le layout le plus avancé et le plus puissant. Néanmoins, c'est également le plus complexe à utiliser. Il permet de positionner les composants dans une grille et il permet d'indiquer beaucoup de données pour indiquer comment placer tel composant dans la grille. Ainsi on pourra dire à un composant de se redimensionner en hauteur mais pas en largeur et lui mettre des marges à gauche et à droite.

Afin de vous simplifier la tâche dans ce TP on va travailler avec `absolute layout`.

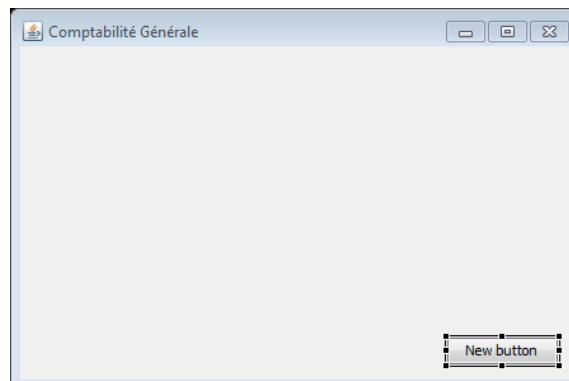


### 3.3. Ajouter un bouton Quitter

Dans la liste des composants sélectionnez `JButton` et dessinez ensuite un bouton sur la fenêtre.

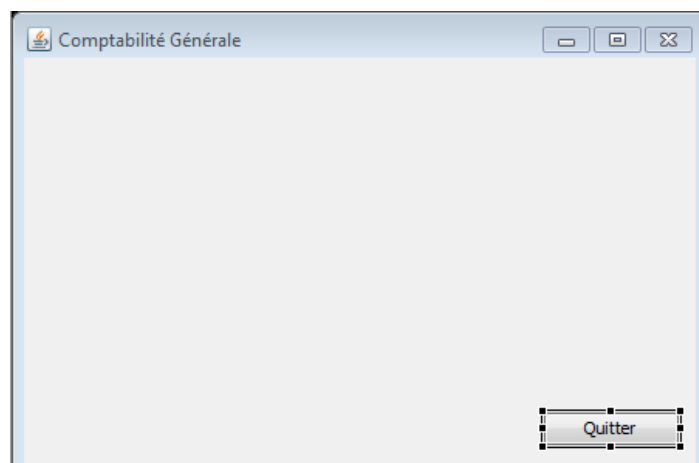
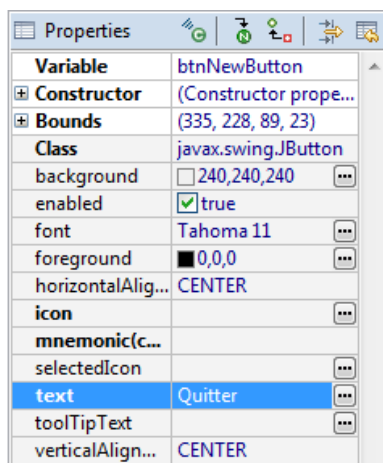


Vous devriez obtenir une fenêtre similaire à celle-ci :



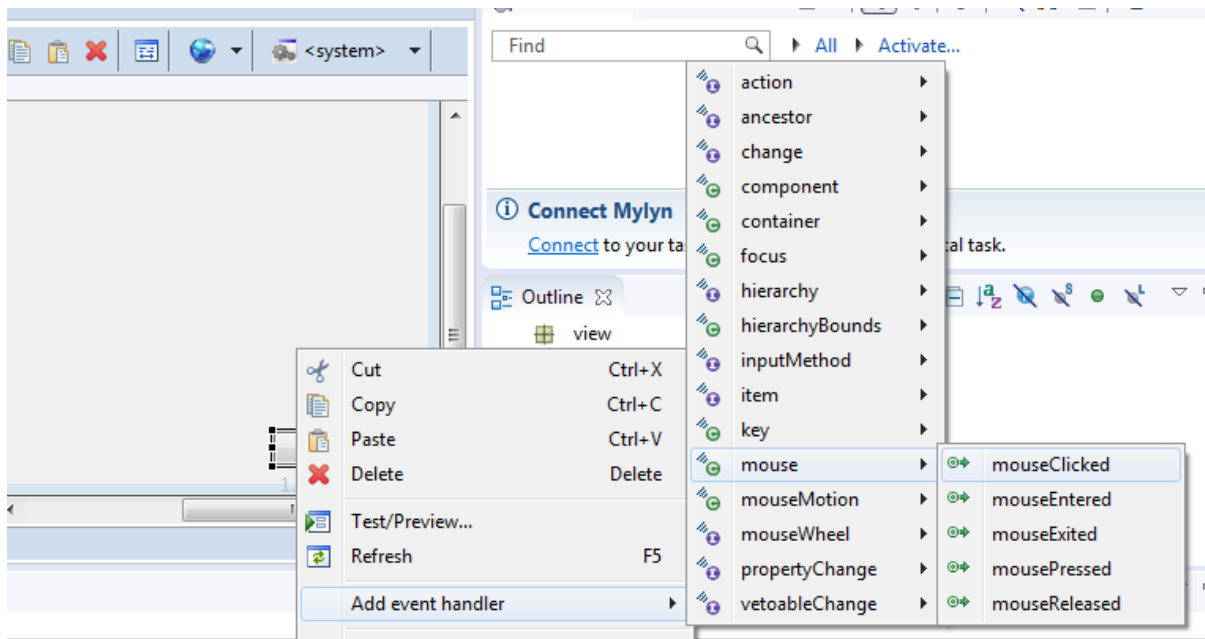
Nous allons changer l'apparence de ce bouton en faisant apparaître Quitter.

Sélectionnez le bouton à la souris, et examinez le panneau `Properties`. Modifiez ensuite le champ `Text`. Après avoir saisi Quitter et une fois validé par la touche Entrée, vous obtiendrez :



### 3.4. Attachez une action au bouton Quitter

Faire un click droit sur le bouton pour faire apparaître le menu contextuel. Et choisir ensuite : `Add event handler/mouse/mouseClicked`. Cela signifie que nous désirons attacher un événement lorsque l'événement click se produira sur le bouton.



Automatiquement, Eclipse vous propose de remplir le code qui sera associé à l'événement Click sur le bouton. Le nom de la procédure est clair de ce point de vue.

```

JButton btnNewButton = new JButton("Quitter");
btnNewButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        |
    }
});

```

Je vous propose de mettre le code suivant :

```

JButton btnNewButton = new JButton("Quitter");
btnNewButton.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        System.exit(0);
    }
});

```

Ce code permettra de terminer l'application sur l'événement Click. Je vous laisse vérifier que cela fonctionne (prenez le temps de compiler et de tester l'application).

## 4. Communiquer avec l'utilisateur

### 4.1. Lire les données saisies et afficher

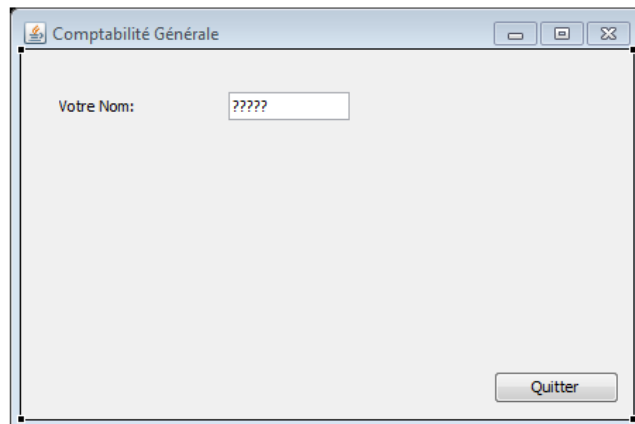
Utilisez les `JLabel` et les `JTextField` pour construire une fenêtre similaire à la fenêtre ci-dessous.



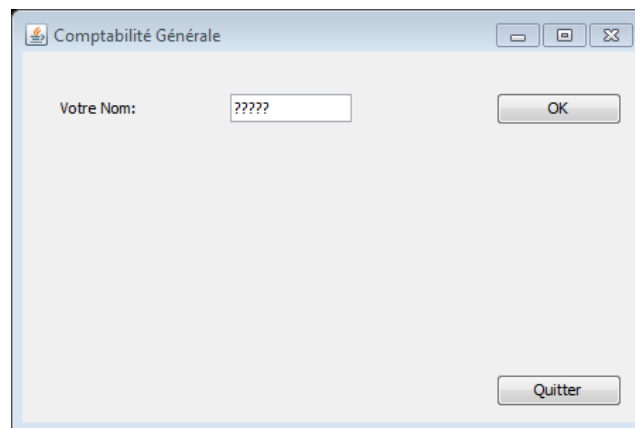


Je vous laisse deviner comment j'ai modifié le texte initial du `JLabel` et le contenu du `JTextField`.

Vous devriez obtenir ceci :

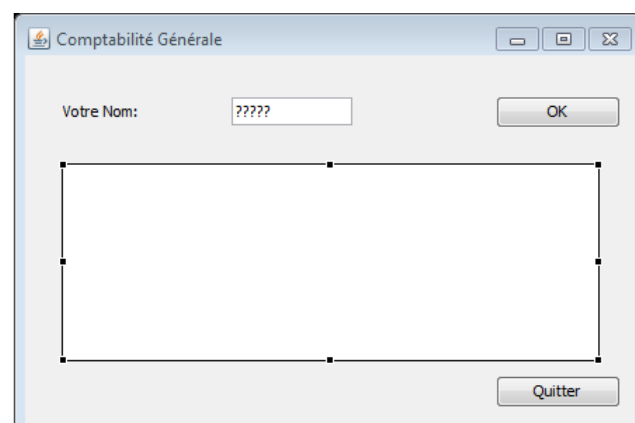


Ajoutez ensuite un `Jbouton` :



Et pour conclure ajoutez un `JTextArea`.

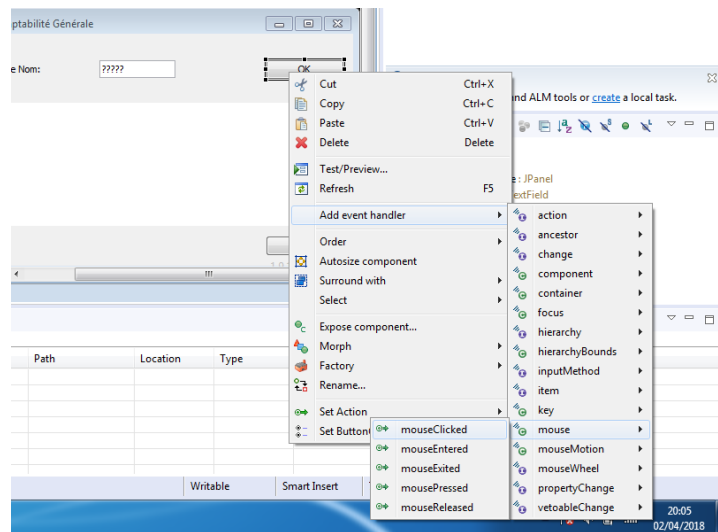
Au final voici à quoi pourrait ressembler votre interface :



Nous allons réaliser l'opération suivante : lorsque l'utilisateur « Click » sur le bouton ok, nous allons afficher dans le `JTextArea` (ce composant se comporte comme une console) le message suivant : « Bonjour XXXX ».

Pour cela faites comme précédemment en appelant le menu contextuel de votre bouton OK.





Eclipse devrait alors vous proposer ceci :

```

JButton btnNewButton_1 = new JButton("OK");
btnNewButton_1.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
    }
});

```

### Remarque :

Le texte sur le bouton est bien OK, mais d'un point de vue Objet, le bouton s'appelle `btnNewButton_1`.

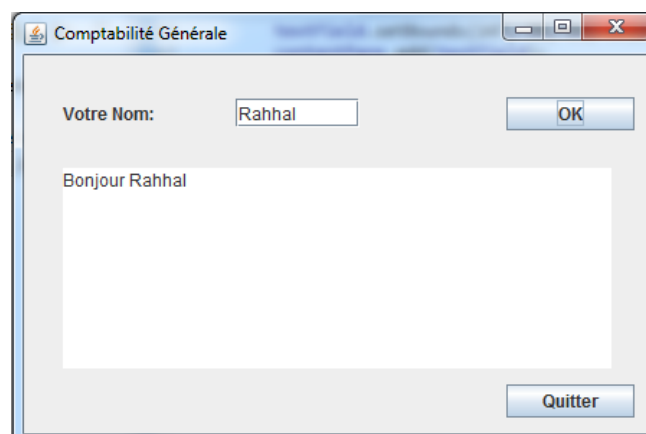
Maintenant ajoutez le code Java suivant :

```

JButton btnNewButton_1 = new JButton("OK");
btnNewButton_1.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        String nom;
        nom = textField.getText();
        textArea.setText("Bonjour " + nom);
    }
});

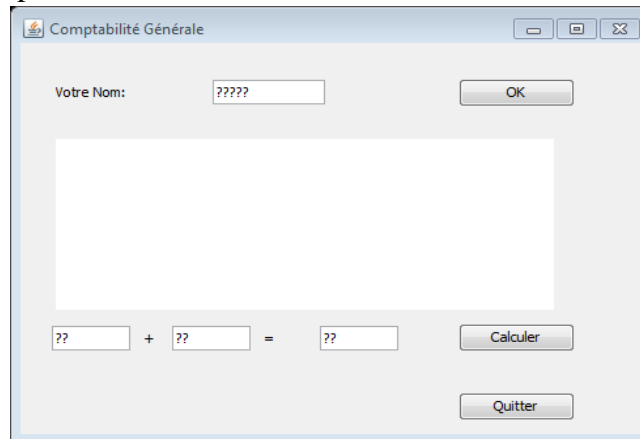
```

Vous obtiendrez à l'exécution ceci :



## 4.2. Les nombres réels

Modifier la fenêtre précédente Comme ceci :



Déclarez un événement sur le bouton que vous venez de définir, puis modifier le code Java suivant :

```
int i,j,resultat;
String Chaine1, Chaine2, Chaine3;

// récupération des deux données au format String...
Chaine1 = textField_1.getText();
Chaine2 = textField_2.getText();

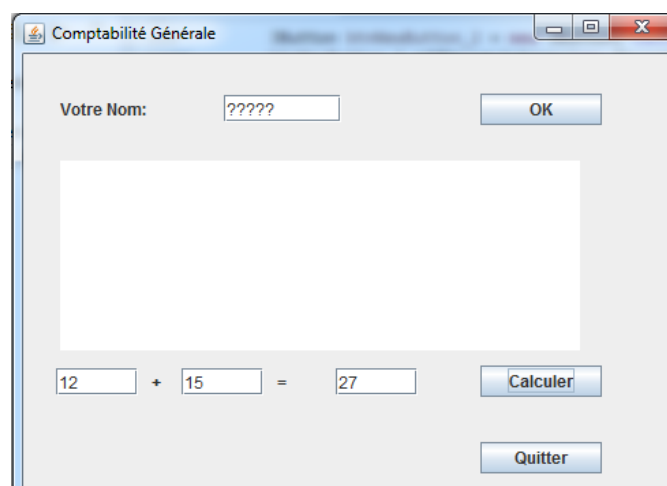
// conversion en int
i = Integer.parseInt(Chaine1);
j = Integer.parseInt(Chaine2);

// addition
resultat = i + j;

// conversion en chaîne
Integer rr = Integer.valueOf(resultat);
Chaine3 = rr.toString();

// on affiche le résultat
textField_3.setText(Chaine3);
```

Vous pourrez ensuite vérifier que les calculs sont justes :



## 5. Intégration d'une interface graphique pour l'application bibliothèque

- A. Créer une class **MainMenu** qui étend la classe **JFrame** (package `java javax.swing`) pour la fenêtre principale de l'application. Dans cette fenêtre placer les composants adéquats pour reproduire l'interface suivant:



- B. Ajouter l'événement de type **ActionListener** aux deux boutons au moyen de la méthode **addActionListener()** ; Cet **ActionListener** possède une méthode **actionPerformed** qui sera automatiquement appelée chaque fois que l'on cliquera sur un bouton.
- Le bouton **Chercher** permet de chercher dans la classe *bibliotheque* le document avec le nom d'auteur et le titre indiqués. L'objet retourne va être affiché dans la zone de texte au milieu.
  - Le bouton **Quitter** permet de fermer l'application.