Création de votre première interface graphique avec Swing

par Baptiste Wicht

Date de publication :

Dernière mise à jour :

Ce tutoriel vous explique la création d'une interface graphique basique avec Swing.

I - Introduction

Vous avez toujours voulu créer une application avec une interface graphique en java, mais vous n'avez réussi.

Ce tutoriel est fait pour vous. On va y découvrir les composants de base d'une interface graphique et comment les intégrer à votre interface et tout cela en utilisant Swing. A la fin de ce tutoriel, vous aurez créé une petite calculatrice, qui pourra vous servir de base à vos futurs programmes.

Ce tutoriel est divisé en différentes petites étapes, à la fin de chacun de celles-ci, vous pouvez télécharger la source de l'étape pour toujours avoir une base de comparaison en cas d'erreur. Vous pouvez aussi télécharger l'entier des sources à la fin du tutoriel.

II - Création de la fenêtre

En Swing, une fenêtre tel que la fenêtre de votre browser, s'appelle une JFrame. Pour créer une nouvelle fenêtre pour votre application, il faudra donc créer une nouvelle classe étendant JFrame. Il ne faudra bien entendu pas non plus oublier d'importer JFrame. Tous les composants dont nous allons parler dans ce tutoriel sont dans le package Swing, nous allons donc importer l'entier de ce package avec javax.swing.* pour plus de simplicité.

Ainsi, vous pourriez faire quelque chose dans ce gout-là :

```
import javax.swing.*;

public class SimpleFenetre extends JFrame{

	public SimpleFenetre(){
		super();
	}
}
```

Sauf qu'ainsi, on afficherait rien du tout, il va donc falloir ensuite configurer cette fenêtre pour qu'elle fasse ce que l'on veut. Pour cela, on va créer une méthode qui va initialiser notre fenêtre et appeler celle-ci depuis notre constructeur.

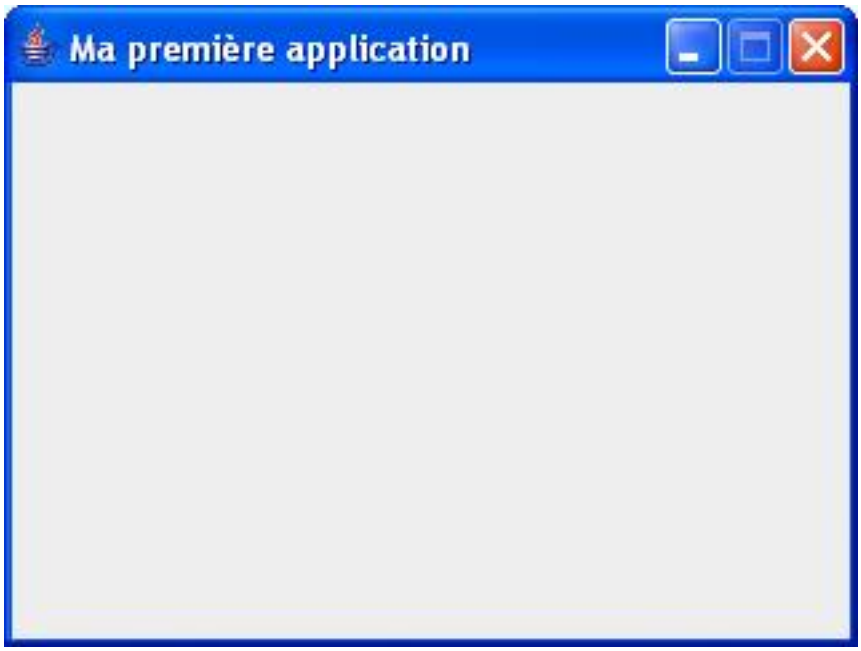
```
public class SimpleFenetre extends JFrame{

	public SimpleFenetre(){
		super();

		build();//On initialise notre fenêtre
	}

	private void build(){
		this.setTitle("Ma première application"); //On donne un titre à l'application
		this.setSize(320,240); //On donne une taille à notre fenêtre
		this.setLocationRelativeTo(null); //On centre la fenêtre sur l'écran
		this.setResizable(false) ; //On interdit la redimensionnement de la fenêtre
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); //On dit à l'application de se
fermer
		//lors du clic sur la croix
	}

}
```

Voilà, avec ce code, on a maintenant une fenêtre de 320 pixels sur 240 pixels ayant Ma première application comme titre, n'étant pas redimensionnable, centrée sur l'écran et qui ferme le programme lors de sa fermeture . Mais il faut pouvoir l'afficher maintenant.

Pour l'afficher, on va donc ajouter une méthode main à notre application et créer une nouvelle instance de notre fenêtre. Pour afficher une nouvelle fenêtre sur notre créer, il suffit de créer une nouvelle instance de celle-ci et ensuite d'appeler setVisible(true) sur cette instance pour afficher la fenêtre à l'écran.

```
public static void main(String[] args){
	//On crée une nouvelle instance de notre fenêtre
	SimpleFenetre gui = new SimpleFenetre();
	gui.setVisible(true);//On la rend visible
}
```

Lancer maintenant votre application. Vous verrez une fenêtre toute vide portant votre titre s'afficher au milieu de l'écran.



Votre première fenêtre Swing

Vous pouvez maintenant essayer de modifier les paramètres de cette fenêtre, pour vous familiariser avec celle-ci.

III - Afficher du texte dans la fenêtre

Maintenant que l'on a créé une fenêtre graphique pour notre application, il va bien falloir mettre quelque chose dedans, on va commencer par apprendre comment y afficher du texte.

Il y a plusieurs composants que l'on peut utiliser pour afficher du texte en Swing, on va ici apprendre à utiliser le plus simple, le JLabel, c'est une simple zone de texte.

Les méthodes intéressantes que l'on va employer sur notre JLabel, sont surtout la méthode setText(String texte) qui va changer le texte à l'intérieur du JLabel, la méthode setPreferedSize(Dimension dimension) qui va nous permettre de donner une taille à notre composant. Comme vous le voyez, on va devoir utiliser un objet de type Dimension, c'est juste un objet formé de 2 valeurs, une hauteur et une largeur. Il nous faudra donc importer java.awt.Dimension pour l'utiliser.

Mais avant d'ajouter quoi que ce soit à notre fenêtre, il faut définir son contentPane, c'est-à-dire le container principal de la fenêtre. Pour cela, on va utiliser un JPanel. C'est un composant tout simple, une sorte de panneau en fait. On n'emploie pas directement pour faire quelque chose, mais on l'emploie pour contenir d'autres composants.

On va ensuite définir un gestionnaire de composant (layout) pour gérer le positionnement des différents éléments au sein du JPanel. Il existe beaucoup de layout différents en java, certains permettant beaucoup de chose. Ici, on va utiliser un layout simple, mais néanmoins très pratique, le FlowLayout. Il permet de répartir les composants dans notre container, sans changer leur taille et permet de choisir l'alignement de ces composants. Si vous voulez plus d'infos sur les différents layouts, vous pourrez trouver des ressources très intéressantes sur developpez.com. les layouts ne viennent pas du package Swing, mais du package java.awt, on va donc importer java.awt.FlowLayout.

On va reprendre le code du chapitre précédent pour réaliser cela, on va juste renommer la classe. On va donc y créer une nouvelle méthode getContainer() qui va initialiser notre contentPane. Et ensuite appeler cette méthode dans notre méthode build pour le mettre en container principal de la fenêtre.

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

public class FenetreAvecTexte extends JFrame{
                private JPanel container = null;//Déclaration de l'objet JPanel
                private FlowLayout layout = null ;//Déclaration de notre layout

                public FenetreAvecTexte(){
                        super();

                        build();
                }

                private void build(){
                        ...
                        //On lui spécifie un container principal
                        this.setContentPane(getContainer());
                }

        private JPanel getContainer(){
                layout = new FlowLayout(); //Instanciation du layout
                layout.setAlignment(FlowLayout.CENTER);//On centre les composants

                container = new JPanel() ; //On crée notre objet
                container.setLayout(layout); //On applique le layout

                return container ;
        }
}
```

En l'étact actuel des choses, vous voyez que rien n'a changé au point de vue visuel, mais il faut s'imaginer un container prêt à accueillir d'autres composants.

Ensuite, on va déclarer et créer notre JLabel et y afficher un texte. Et ensuite, l'ajouter au container principal de la fenêtre pour qu'il soit visible, sans quoi il ne servirait à rien.

```
public class FenetreAvecTexte extends JFrame{
        private JLabel texte = null;//Déclaration de l'objet JLabel
        ...

        private void getContainer(){
        ...

                texte = new JLabel() ;//On crée notre objet
                texte.setPreferredSize(new Dimension(250,25)) ;//On lui donne une taille
                texte.setText("Bienvenue dans ma première application"); //On lui donne un texte

                container.add(texte);//On l'ajoute au container

                return container ;
        }
}
```

Si vous avez bien tout suivi jusqu'ici, lors du lancement de ce programme, vous devriez voir s'afficher une fenêtre avec un message de bienvenue affiché au milieu.

Vous avez maintenant réussi à afficher du texte dans votre fenêtre

Vous pouvez maintenant essayer de modifier les différents paramètres que nous avons définis pour ce label et essayer d'ajouter d'autres JLabel à votre fenêtre pour vous familiariser avec ce composant, qui reste très simple et basique.

IV - Ajouter des boutons dans la fenêtre

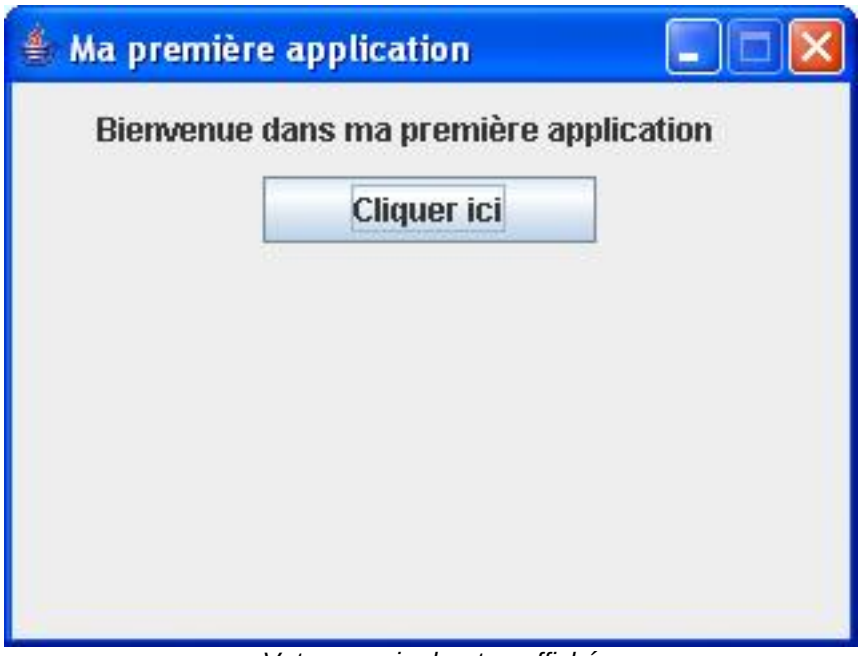
La principale fonctionnalité d'une application graphique est la programmation événementielle, c'est-à-dire le fait que l'utilisateur peut déclencher des événements et régir ce qui se passe dans la fenêtre. Au contraire d'un programme en mode console, dans lequel, c'est le programme qui régit les actions de l'utilisateur à sa guise.

Pour que l'utilisateur puisse interagir avec le programme, on dispose par exemple, des boutons. Vous allez donc apprendre dans ce chapitre, à vous en servir. Le composant pour créer un bouton est le JButton. On va donc créer un nouveau JButton et l'intégrer à notre application. Les méthodes qui vont nous intéresser pour le JButton, sont la méthode addActionListener() qui permettra d'ajouter un écouteur au bouton, setPreferredSize pour donner une taille au bouton et setText pour lui donner un texte.

On va reprendre à nouveau le code de la fenêtre précédente et y ajouter notre bouton.

```
public class FenetreAvecBouton extends JFrame{
        private JButton bouton = null;//Déclaration du bouton
        ...

        private void getContainer(){
                ...

                bouton = new JButton () ;//Création du bouton
                bouton.setPreferedSize(new Dimension(125,25)) ;//On lui donne une taille
                bouton.setText("Cliquer ici") ;//On lui donne un texte

                container.add(bouton);//On l'ajoute à la fenêtre

                return container ;
        }
}
```

Vous avez donc maintenant, un bouton qui va s'afficher sous votre texte de bienvenue.



Votre premier bouton affiché

C'est bien beau, vous me direz, mais comme ca, ca ne sert à rien. On va donc attribuer une action à ce bouton. Pour cela, il va falloir lui ajouter un listener, un écouteur d'action. En java, un écouteur d'action est un ActionListener. Il faudra donc définir qui va être l'écouteur de notre bouton, dans notre cas, on va dire que c'est la fenêtre elle-même, qui va écouter notre bouton pour ne rien compliquer. Car on peut aussi créer une autre classe qui ferait office d'écouteur. Pour qu'une classe puisse être écouteuse d'action, il faut qu'elle implémente, l'interface ActionListener. Pour cela, il vous suffira d'ajouter dans le signature de la classe implements ActionListener. Ainsi, la signature de notre classe deviendra :

```
public class FenetreAvecBouton extends JFrame implements ActionListener{
```

Sans oublier d'importer ActionListener et ActionEvent :

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
```

Si vous travaillez avec un EDI tel qu'eclise ou Netbeans, vous verrez que cette ligne, vous mettra des erreurs. Car, quand on implémente une interface, il faut qu'on redéfinisse ces différentes méthodes, dans notre cas, ActionListener, possède seulement la méthode actionPerformed(ActionEvent e) qui va être appelée à chaque fois que l'écouteur reçoit un événement. On va donc ajouter cette méthode dans notre classe :

```
public class FenetreAvecBouton extends JFrame implements ActionListener {
        ...

        public void actionPerformed(ActionEvent e) {

        }
}
```

Ensuite, on va ajouter l'écouteur à notre bouton :

```
bouton.addActionListener(this);//On ajoute la fenêtre en tant qu'écouteur du bouton
```

Avec ce code, il n'y a toujours rien qui se passe, mais c'est normal, puisque l'on ne rien fait dans la méthode actionPerformed.

Maintenant, il va falloir définir une action dans le listener pour notre bouton. On va donc mettre cette action dans la méthode actionPerformed. Dans notre cas, comme on n'a qu'un seul bouton qui est écouté par la méthode, on pourrait se permettre de mettre directement l'action dans la méthode, mais pour prendre de bonnes habitudes, on va d'abord contrôler que l'action émane bien de notre bouton. Pour savoir d'où vient l'action, on dispose de l'événement passé en paramètre de la méthode actionPerformed, cet événement possède une méthode getSource(), qui va nous dire d'où vient l'action.

Si par exemple, on veut qu'à chaque clic sur le bouton, ca affiche un nombre dans notre label et que ce nombre augmente à chaque clic, il va falloir faire quelque chose du genre :

```
public class FenetreAvecBouton extends JFrame implements ActionListener {
        private int nombre = 0; //Déclaration du chiffre
        ...

        public void actionPerformed(ActionEvent e) {
                if(e.getSource() == bouton){//Si l'action émane bien du bouton
                        nombre++;//On incrémente nombre de 1
                        texte.setText("Vous avez cliqué " + nombre + " fois sur le bouton"); //On
nomre dans le label
                }
        }
}
```

Ainsi, avec ce code, dès que vous cliquerez pour la première fois sur le bouton, le texte de bienvenue s'effacera, pour laisser place à un 0. Et ensuite, à chaque clic, le nombre augmentera.

Le texte s'incrémente à chaque pressée sur le bouton

Vous pouvez maintenant essayer de faire autre chose avec votre bouton. Et essayer de faire quelque chose avec plusieurs boutons, par exemple, un bouton qui augmente le nombre et un autre qui le diminue.

V - Demander du texte à l'utilisateur

Maintenant que vous savez afficher du texte à l'écran, et employer des boutons pour créer un événement. On va apprendre à demander à l'utilisateur du texte. Pour cela, il existe plusieurs composants :

- Le JTextField : très basique, il permet seulement d'entrer un texte sur une seule ligne. C'est sur celui-là que nous allons nous étendre maintenant.
- Le JTextArea : Il permet d'entrer un texte complet sur plusieurs lignes.
- Le JEditorPane : Très complet, vous pouvez modifier la police, la taille, la couleur de votre texte. Il permet même d'afficher des pages html.

Pour récupérer le texte entrée, il faut employer la méthode getText(). Sinon, pour employer un JTextField, c'est à peu près de la même manière que pour un autre composant. On va employer les méthodes setText et setPreferredSize comme sur un JLabel.

On va donc créer une petite application qui va prendre 2 nombre en entrées, les additionner sur clic d'un bouton et afficher le résultat et ajouter un autre bouton qui effacera les 2 champs.

On aura donc :

- Un bouton calculer
- Un label +
- Un label pour le résultat
- 2 Champs d'entrée pour nombre1 et nombre2

Voilà ce que sera la base de notre programme :

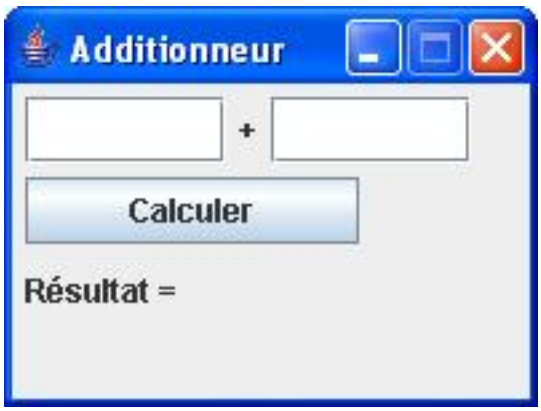
```
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;

public class Additionneur extends JFrame implements ActionListener{
        private JPanel container = null;//Déclaration de l'objet JPanel
        private FlowLayout layout = null ;//Déclaration de notre layout

        public Additionneur (){
                super();

                build();
        }

        private void build(){
                this.setTitle("Additionneur"); //On donne un titre à l'application
                this.setSize(200,150); //On donne une taille à notre fenêtre
                this.setLocationRelativeTo(null); //On centre la fenêtre sur l'écran
                this.setResizable(false) ; //On interdit la redimensionnement de l'écran
                this.setContentPane(getContainer()) ;//On lui dit de mettre le panel en fond
                this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                //On dit à l'application de se fermer lors du clic sur la croix
        }

        private JPanel getContainer(){
                layout = new FlowLayout() ; //Instanciation du layout
                layout.setAlignment(FlowLayout.LEFT) ;//On aligne à gauche

                container = new JPanel() ; //On crée notre objet
                container.setLayout(layout);

                return container ;
        }

        public static void main(String[] args){
                Additionneur gui = new Additionneur ();
```

```
                gui.setVisible(true);
        }

        public void actionPerformed(ActionEvent e){

        }
}
```

On va ensuite déclarer nos composants et les initialiser :

```
Public class Additionneur extends JFrame implements ActionListener{
        private JButton buttonCalcul = null ;
        private JLabel labelOperateur = null ;
        private JLabel labelResultat = null;
        private JTextField fieldNombre1 = null;
        private JTextField fieldNombre2 = null;

        ...

        private void getContainer(){
                ...

                fieldNombre1 = new JTextField();
                fieldNombre1.setPreferredSize(new Dimension(75,25));
                container.add(fieldNombre1);

                labelOperateur = new JLabel("+");
                labelOperateur.setPreferredSize(new Dimension(7,25));
                container.add(labelOperateur);

                fieldNombre2 = new JTextField();
                fieldNombre2.setPreferredSize(new Dimension(75,25));
                container.add(fieldNombre2);

                buttonCalcul = new JButton("Calculer");
                buttonCalcul.setPreferredSize(new Dimension(125,25));
                buttonCalcul.addActionListener(this) ;
                container.add(buttonCalcul);

                labelResultat = new JLabel("Résultat = ");
                labelResultat.setPreferredSize(new Dimension(100,25));
                container.add(labelResultat);

                return container ;
        }
}
```

Vous avez maintenant une interface avec un 2 champ d'entrée séparés par un + et un bouton calcul, mais tout cela, ne fait encore rien.



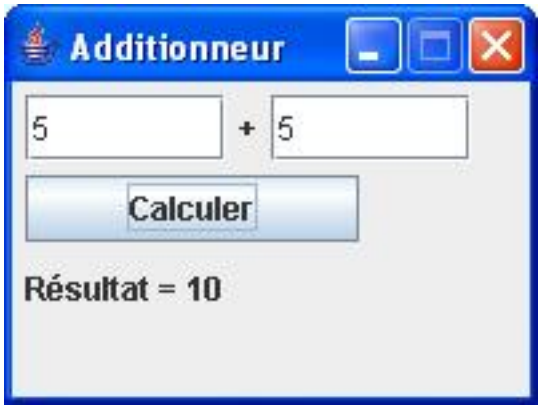
Vous avez maintenant conçu une interface un peu plus complexe.

Lors du clic sur le bouton calcul, il faudra tout simplement récupérer le premier chiffre, récupérer le deuxième et les additionner. Puis afficher le résultat dans le bon label.

Voila ce que va donc donner notre code de la méthode actionPerformed, à noter que cette fois, il ne sert à rien de vérifier que l'action émane du bouton puisque qu'il n'y a que lui qui est écouté par cette méthode.

```
public void actionPerformed(ActionEvent e){
        String nombre1String = fieldNombre1.getText();//On récupère la valeur dans le premier champ
        int nombre1 = Integer.parseInt(nombre1String);//On convertit cette valeur en un nombre

        String nombre2String = fieldNombre2.getText();//On récupère la valeur dans le deuxième champ
        int nombre2 = Integer.parseInt(nombre2String);//On convertit cette valeur en un nombre

        int resultat = nombre1 + nombre2;
        labelResultat.setText("Résultat = " + resultat);
}
```

Vous avez maintenant créé votre propre calculette.



Votre additionneur en action

Bon, c'est vrai qu'elle ne fait que des additions, mais avec le chapitre suivant, elle va évoluer. Vous pourriez aussi améliorer cette calculette en faisant des tests sur la grandeur des nombres, car ici, nous nous sommes contentés de les transformer en int, mais int ne gère par les nombres plus grands que 2147483647, dans le cas ou un nombre plus grand apparaitrait, une exception serait levée et le calcul ne serait pas effectuée.

VI - Proposer une liste de choix à l'utilisateur

Après avoir vu les champs de texte, on va maintenant voir un nouveau champ de saisie, les listes déroulantes. En Swing, il existe 2 sortes de listes déroulantes :

- JList, liste déroulante assez avancée qui permet d'afficher plusieurs éléments à la fois, on ne va pas en parler, car plus compliquée en prendre en main.
- JComboBox, c'est de celle-ci que nous allons parler dans ce chapitre, c'est une liste déroulante normale, de celle que vous voyez partout dans les applications.

Pour employer une JComboBox, on peut employer les méthodes suivantes :

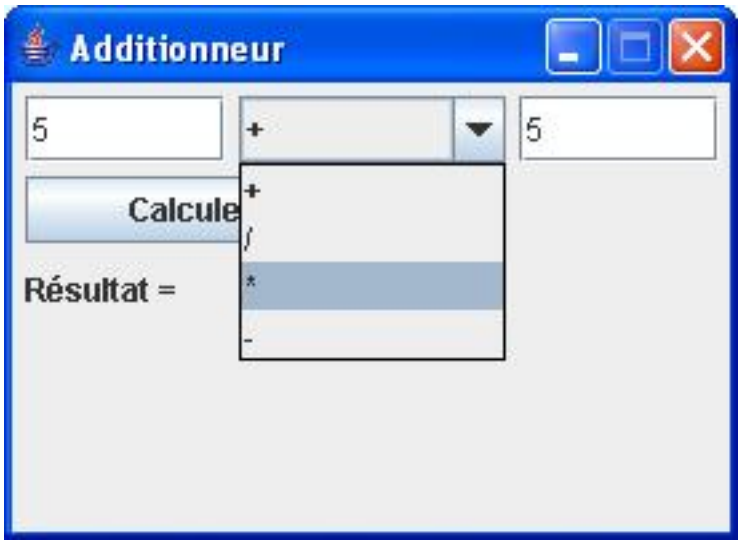
- addItem(Objet item), cette méthode va ajouter un nouvel objet à la liste
- removeItem(Objet item), cette méthode va enlever l'objet de la liste
- removeAllItems(), cette méthode va vider la liste déroulante
- getSelectedItem(), cette méthode retourne l'objet qui est actuellement sélectionné

Comme vous le voyez, l'emploi de ce composant est très simple, mais il permet encore d'autres choses qu'on ne verra pas dans ce chapitre.

Pour ce chapitre, on va reprendre le code de l'additionneur et on va en faire une calculatrice. On va juste modifier le label + par une liste déroulante contenant les 4 opérateurs de base. Ainsi, notre calculatrice, va maintenant avoir 4 opérateurs.

```
Public class Calculateur extends JFrame implements ActionListener{
        Private JComboBox listOperateurs = null ;
        Private Object[] operateurs = new Object[]{"+","/","*","-"} ;
        ...

        private void getContainer(){
                ...

                listOperateurs = new JComboBox(operateurs);//On crée la liste en lui donnant un
tableau d'opérateurs
                        listOperateurs.setPreferredSize(new Dimension(100,25));//On lui donne une
taille
                        container.add(listOperateurs); //on l'ajoute à la fenêtre
                ...
                return container ;
        }
}
```

Vous devriez maintenant avoir une liste entre vos deux champs texte permettant de choisir entre les quatre opérateurs de base. Comme vous l'avez vu cela déforme notre bel affichage, pour pallier à ceci, il faut mettre une taille de 275,200 à notre fenêtre.

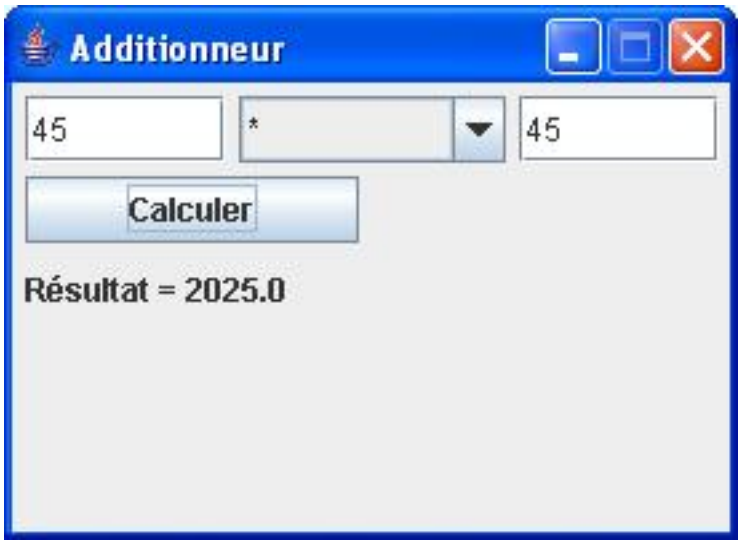


Vous avez maitenant afficher votre première liste déroulante

Mais il faut maintenant que notre bouton puisse effectuer la bonne action en fonction de l'opérateur sélectionné.

```
public void actionPerformed(ActionEvent e){
        String nombre1String = fieldNombre1.getText();//On récupère la valeur dans le premier champ
        double nombre1 = Double.parseDouble(nombre1String);//On convertit cette valeur en un nombre

        String nombre2String = fieldNombre2.getText();//On récupère la valeur dans le deuxième champ
        double nombre2 = Double.parseDouble(nombre2String);//On convertit cette valeur en un nombre

        Object operateur = listOperateurs.getSelectedItem();
        listOperateurs.getSelectedIndex();

        double resultat = 0;

        if(operateur.equals("+")){
                resultat = nombre1 + nombre2;
        }
        else if (operateur.equals("-")){
                resultat = nombre1 - nombre2;
        }
        else if (operateur.equals("*")){
                resultat = nombre1 * nombre2;
        }
        else if (operateur.equals("/")){
                resultat = nombre1 / nombre2;
        }

        labelResultat.setText("Résultat = " +  resultat);
}
```

Et voilà, vous avez maintenant une petite calculatrice, permettant d'additionner, de soustraire, de multiplier ou de diviser deux nombres entre eux.



Votre calculatrice en action.

Vous pouvez encore améliorer cette calculatrice, par exemple, en affichant un message d'erreur s'il y a une division par zéro ou alors vérifier s'il y a un opérateur sélectionné dans la liste.

VII - Utiliser des boîtes de dialogue

Dans les composants Swing, il existe un composant très intéressant, le JOptionPane qui possède plusieurs méthodes statiques permettant d'ouvrir d'afficher diverses boîtes de dialogue :

- Les boîtes de messages, ces boîtes de dialogue permettent d'afficher un message pour l'utilisateur.
- Les boîtes de saisie, avec cette boite de dialogue, on va pouvoir demander une chaîne de caractères à l'utilisateur
- Les boites de confirmation, avec ces boîtes de dialogues, on peut demander une confirmation à l'utilisateur.
- Les boîtes de choix, qui permettent de donner le choix entre plusieurs choses, une style déroulante en quelque sorte

Pour utiliser ces différentes boîtes de dialogue, on va maintenant recréer une calculatrice mais sans interface graphique, uniquement avec des boîtes de dialogue.

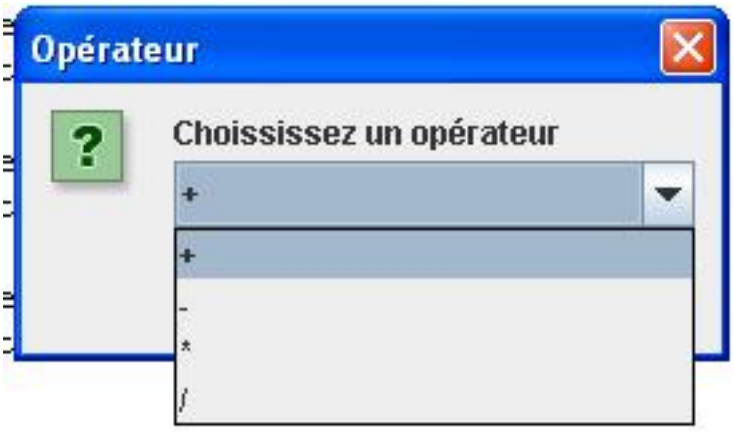
Voilà donc ce qui va se passer dans notre programme :

1 on va demander le premier nombre à l'utilisateur
2 On va demander un deuxième nombre à l'utilisateur
3 On va demander l'opérateur à l'utilisateur
4 On va afficher le résultat de l'addition de ces 2 nombres
5 On va demander à l'utilisateur s'il a apprécié ce programme
6 En fonction de sa réponse, on va afficher un message.

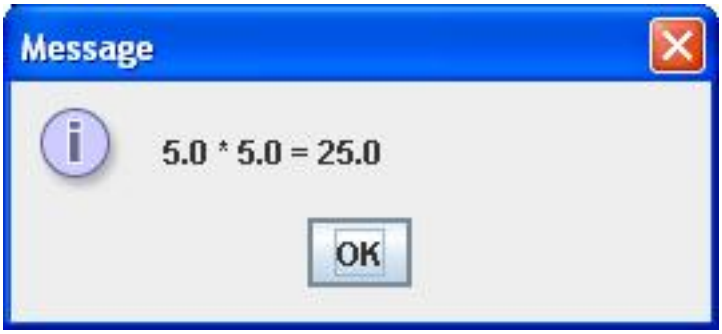
```
import javax.swing.JOptionPane;

public class CalcutateurDialogue {

	public static void main(String[] args){
		double resultat = 0;

		double nombre1 = Double.parseDouble(JOptionPane.showInputDialog(null, "Entrez un
premier nombre")) ;

		double nombre2 = Double.parseDouble(JOptionPane.showInputDialog(null, "Entrez un
second nombre")) ;

		String[] operateurs = new String[]{"+","-","*","/"};

		String operateur = (String)JOptionPane.showInputDialog(null,"Choississez un
opérateur",


			"Opérateur",JOptionPane.QUESTION_MESSAGE, null, operateurs, operateurs[0]);

		if(operateur.equals("+")){
			resultat = nombre1 + nombre2;
		}
		else if (operateur.equals("-")){
			resultat = nombre1 - nombre2;
		}
		else if (operateur.equals("*")){
			resultat = nombre1 * nombre2;
		}
		else if (operateur.equals("/")){
			resultat = nombre1 / nombre2;
		}

		JOptionPane.showMessageDialog(null, nombre1 + " " + operateur + " " + nombre2 + " =
" + resultat);

		int reponse = JOptionPane.showConfirmDialog(null, "Êtes-vous content de ce programme
?",
```

```
                        "Satisfaction", JOptionPane.YES_NO_OPTION);

                if(reponse == JOptionPane.YES_OPTION){
                        JOptionPane.showMessageDialog(null, "Cool :) A la prochaine") ;
                }else if(reponse == JOptionPane.NO_OPTION){
                        JOptionPane.showMessageDialog(null, "Dommage :( Peut-être trouverez-vous
votre bonheur" +


                                " dans les futurs programmes que je vais développer") ;
                }
        }
}
```

Demande de texte à l'utilisateur



Proposition de choix à l'utilisateur



Envoyer un message à l'utilisateur

Dans cet exemple-là, on employait seulement des boites de dialogue, mais c'est très rare, les boîtes de dialogue sont plutôt faite pour seconder l'interface graphique. Vous pouvez employer ces boîtes de dialogue exactement de la même manière dans une interface graphique que dans cet exemple.

VIII - Ajouter un menu à votre fenêtre

Maintenant que l'on a fait une calculatrice complète, on va y afficher un menu pour qu'elle fasse plus sérieuse. En Swing, pour mettre un menu sur votre JFrame, il faut employer une JMenuBar, qui sera composé de JMenu et de JMenuItem.

Une fois encore, pour savoir quand un élément du menu est cliqué, il faut avoir recours à un listener, dans notre cas, on va les faire écouter par notre classe Calculateur, il va donc falloir faire cohabiter les éléments du menu et le bouton calculer. Pour cela, on va de nouveau employer la méthode getSource() ;

Pour cette calculatrice, on va faire deux menus :

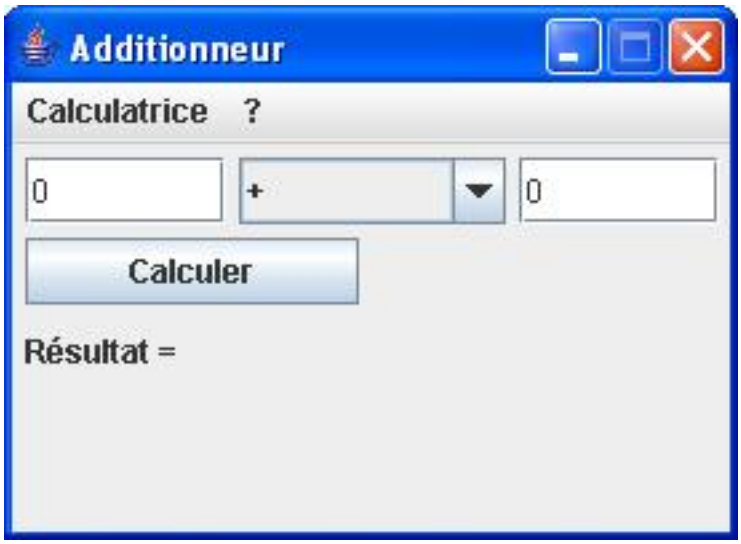
- Un menu "Calculatrice" avec un bouton calculer et un bouton quitter
- Un menu "?" qui contiendra un bouton "à propos"

On va donc reprendre le code de la calculatrice et y ajouter les éléments du menu.

```
Public class CalculateurAvecMenu extends JFrame implements ActionListener{
        ...

        private JMenuBar menuBar = null ; ;
        private JMenu menu1 = null ;
        private JMenuItem calculer = null ;
        private JMenuItem quitter = null ;
        private JMenu menu2 = null ;
        private JMenuItem aPropos = null ;

        public CalculateurAvecMenu(){
                super();

                build();
        }

        private void build(){
                menuBar = new JMenuBar() ;

                menu1 = new JMenu("Calculatrice") ;

                calculer = new JMenuItem("Calculer") ;
                calculer.addActionListener(this) ;
                menu1.add(calculer) ;

                quitter = new JMenuItem("Quitter") ;
                quitter.addActionListener(this) ;
                menu1.add(quitter) ;

                menuBar.add(menu1) ;

                menu2 = new JMenu("?") ;

                aPropos = new JMenuItem("A propos") ;
                aPropos.addActionListener(this) ;
                menu2.add(aPropos) ;

                menuBar.add(menu2) ;
                this.setJMenuBar(menuBar) ;
                ...
        }

        public void actionPerformed(ActionEvent e){
                if(e.getSource() == buttonCalcul || e.getSource() == calculer){
                        String nombre1String = fieldNombre1.getText();//On récupère la valeur dans
le premier champ
                        double nombre1 = Double.parseDouble(nombre1String);//On convertit cette
valeur en un nombre

                        String nombre2String = fieldNombre2.getText();//On récupère la valeur dans
le deuxième champ
                        double nombre2 = Double.parseDouble(nombre2String);//On convertit cette
```

```
valeur en un nombre

                        Object operateur = listOperateurs.getSelectedItem();
                        listOperateurs.getSelectedIndex();

                        double resultat = 0;

                        if(operateur.equals("+")){
                                resultat = nombre1 + nombre2;
                        }
                        else if (operateur.equals("-")){
                                resultat = nombre1 - nombre2;
                        }
                        else if (operateur.equals("*")){
                                resultat = nombre1 * nombre2;
                        }
                        else if (operateur.equals("/")){
                                resultat = nombre1 / nombre2;
                        }

                        labelResultat.setText("Résultat = " +  resultat);
                }else if(e.getSource() == quitter){
                        System.exit(0);
                }else if(e.getSource() == aPropos){
                        JOptionPane.showMessageDialog(null,"Ce programme a été développé par XXX");
                }
        }
}
```

Vous avez maintenant une jolie calculatrice avec un menu. Vous pourrez vous servir de ce menu en guise d'exemple pour toutes les autres applications que vous ferez ensuite.



Votre première application graphique complète, à quand les prochaines

Sachez que vous pouvez aussi créer des sous menus, pour cela, il vous suffit d'ajouter un JMenu à un autre JMenu et ensuite des JMenuItem dans le JMenu imbriqué. Vous pouvez continuer d'améliorer ce programme pour vous familiariser avec tous les composants de base de l'interfacage graphique en java.

IX - Conclusion

IX-A - Conclusion

Voilà, vous avez maintenant réalisé votre première application avec interface graphique en java. Nous arrivons donc à la fin de ce tutoriel, mais avant d'aller plus loin, je vous conseille d'essayer de faire des modifications et amélioration sur ce programme. Ainsi, vous apprendrez de vous-mêmes des nouvelles choses. Pour des infos sur des méthodes des classes que je vous ai présentées, le mieux est toujours d'aller sur la javadoc et si vous avez des problèmes sur certains points, il ne faut pas hésiter, si vos recherches ont été infructueuses, à poser la question sur les forums de developpez.com.

IX-B - Remerciements

J'aimerais remercier toute l'équipe java pour ses commentaires sur ce tutoriel, qui n'ont fait qu'en améliorer la qualité.

IX-C - Les sources

Vous pouvez télécharger toutes les sources de ce tutoriel, l'archive zip contient tous les fichiers .java. Elle contient un fichier par étape. Si vous voulez seulement le dernier fichier, c'est à dire le fichier complet, il vous suffit de choisir la dernière étape de chaque chapitre.

- Archive zip