Leçon N°4

Les interfaces graphiques

Introduction

Dès la version Java 1.0, SUN a introduit une bibliothèque de classes nommée **AWT** (Abstract Window Toolkit) pour la programmation de base de l'interface graphique utilisateur (*Graphical User Interface)*.

L'inconvénient majeur de la bibliothèque AWT est qu'elle n'est pas 100% Java. En effet, AWT s'appuie entièrement sur le gestionnaire graphique du système d'exploitation pour représenter les composants à l'écran. Ce qui fait que d'un système d'exploitation à un autre, l'interface graphique AWT peut changer énormément au niveau de la taille et de la disposition des composants.

Depuis Java SE 1.2, Sun a intégré une nouvelle bibliothèque d'interfaces utilisateur portant le nom de code **SWING**.

D'une façon générale, une interface utilisateur graphique (GUI) est composée d'éléments de base comme des boutons, des étiquettes (labels), des champs de texte, des cases à cocher (check box), des ascenseurs, des menus déroulants, etc. Ces éléments de base sont appelés « Composants » en Java. Dans certains environnements, ces éléments sont appelés « Widgets » ou « Contrôles ».

En Java, on distingue des **composants légers** (Lightweight) et des **composants lourds** (Heawyweight). Les composants lourds sont basés sur du code natif de la plate-forme d'exécution (Peer Component). Les composants légers sont entièrement écrits en Java et sont donc indépendants de la plate-forme d'exécution. Excepté quelques composants de haut-niveau, pratiquement tous les composants Swing sont des composants légers (contrairement à AWT).

I. Structure d'une interface graphique (GUI)

Il existe deux principaux types de composants susceptibles d'intervenir dans une interface graphique :

- Les conteneurs qui sont destinés à contenir d'autres composants, comme par exemple les fenêtres (frame, window), les boites de dialogue (dialog box), les panneaux (panels), etc. ;

- Les composants atomiques qui sont des composants qui ne peuvent pas contenir d'autres, comme les boutons (buttons), les cases à cocher (checkbox), les boutons radio (radiobuttons), etc.

Pour pouvoir utiliser ces composants, nous devons importer leurs classes des packages spécifiques **javax.swing** et **java.awt**.

Remarque : dans une interface GUI on peut avoir des composants swing et AWT en même temps, mais ceci est déconseillé.

II. Création d'une fenêtre

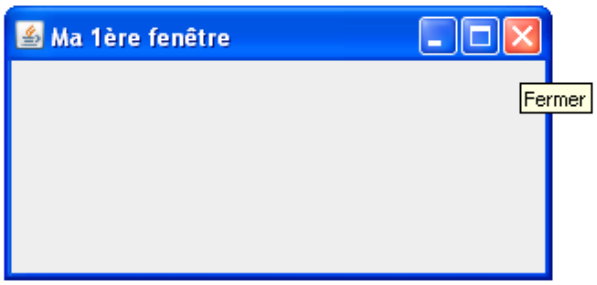
Une interface graphique swing est un arbre qui part d'un objet système lourd (fenêtre, boite de dialogue, applet, …).

Le programme suivant montre comment créer une fenêtre (JFrame) sous Java :

```
import javax.swing.JFrame;

public class Fenetre {
      public static void main(String[] args){
            JFrame maFenetre = new JFrame();
            maFenetre.setTitle("Ma 1ère fenêtre");
            maFenetre.setSize(300, 150);
            maFenetre.setLocationRelativeTo(null);
            maFenetre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            maFenetre.setVisible(true);
      }
}
```

Output



Remarques

- Par défaut, une fenêtre n'a pas de titre. La méthode **setTitle(String)** permet d'ajouter un titre à notre fenêtre dans la barre de titre.

- Par défaut, une fenêtre a une taille de 0 × 0 pixel. La méthode **setSize(largeur, hauteur)** permet d'affecter une taille à notre fenêtre (300 × 150 pixels dans l'exemple).

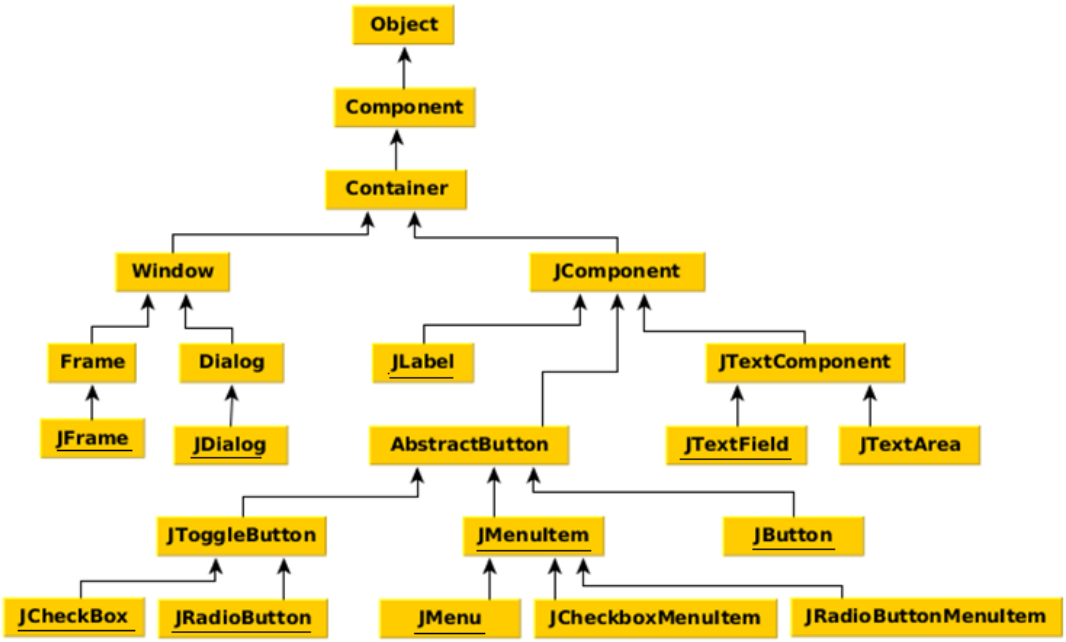
- L'instruction **maFenetre.setLocationRelativeTo(null)** permet d'afficher la fenêtre au milieu de l'écran.

- Par défaut, l'appui sur le bouton de fermeture de la fenêtre permet de la cacher (hide). Les options possibles sont :

    - **EXIT_ON_CLOSE** : fermer la fenêtre et arrêter le programme (le processus).
    - **HIDE_ON_CLOSE** : cacher la fenêtre.
    - **DO_NOTHING_ON_CLOSE** : ne rien faire.
    - **DISPOSE_ON_CLOSE** : détruire la fenêtre.

- Par défaut, une fenêtre est invisible. La méthode **setVisible(true/false)** permet de modifier l'option de visibilité par défaut. Elle doit être utilisée après le remplissage de la fenêtre par les différents composants graphiques.

III. Hiérarchie des classes



Les classes dont le nom commence par « J » comme JFrame, JLabel, JTextArea, JButton, JcheckBox, JTextField, JMenu, JradioButton, … font partie du package **javax.swing** alors que les autres font partie de package **java.awt**.

III.1. La classe JFrame

La classe JFrame propose différentes méthodes pour la création et la personnalisation d'une fenêtre :

Méthode	Rôle
`setTitle(titre)`	définir le titre de la fenêtre
`getTitle()`	récupérer le titre de la fenêtre
`setSize(largeur, hauteur)`	définir la taille de la fenêtre
`setLocation (int x, int y)`	définir la position de la fenêtre
`setBounds(int x, int y, int w,int h)`	définir à la fois la position et la taille de la fenêtre
`setIconImage(image)`	ajouter une icône dans la barre de titre
`setResizable(booléen)`	autoriser ou interdire le redimensionnement d'une fenêtre.
`Pack()`	calculer automatiquement la taille idéale pour afficher tous les composants d'une fenêtre.
`setContentPane()`	Affecte un conteneur de type panel à la fenêtre courante.
`getContentPane()`	fournit une référence, de type « Container », à la surface de contenu (panel) de la fenêtre courante.

III.2. La classe JPanel

Un panel (panneau) est essentiellement un objet de rangement pour d'autres composants. La classe JPanel possède plusieurs constructeurs parmi eux :

- `JPanel()` : panel avec un gestionnaire de placement par défaut (FlowLayout)

- `JPanel(LayoutManager l)` : panel avec le gestionnaire de placement l.

Un panel seul ne peut être affiché, il doit être inséré dans un composant lourd (JFrame ou autre).

Pour ajouter des composants à un panel p, on utilise la syntaxe suivante :

```
JPanel p = new JPanel() ;    // création du panel
p.add(composant1);
p.add(composant2);
…
```

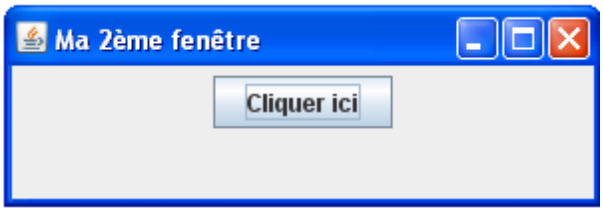
Par la suite, il faut rattacher ce panel à la fenêtre dans laquelle il s'affichera. Deux solutions sont possibles :

```
fenetre.setContentPane(p) ;
ou
fenetre.getContentPane().add(p) ;     // une fenêtre peut contenir plusieurs panels.
```

Exemple 1 : Ajout d'un bouton à la fenêtre

```
import javax.swing.*;
public class Fenetre extends JFrame {
      public Fenetre(){
            this.setTitle("Ma 2ème fenêtre");
            this.setSize(300, 100);
            this.setLocationRelativeTo(null);
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            JPanel pano = new JPanel();
            JButton monBouton = new JButton("Cliquer ici");
            pano.add(monBouton);
            this.setContentPane(pano);
            this.setVisible(true);
      }
      public static void main(String[] args){
            new Fenetre();
      }
}
```

Output

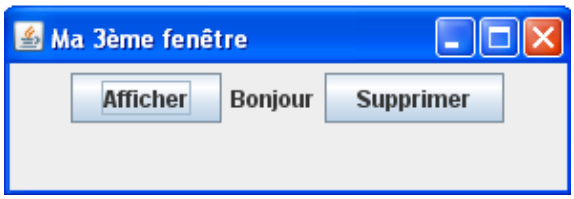


Exemple 2 : Le programme suivant permet de créer une fenêtre contenant : deux boutons et un label au milieu qui contient le texte « Bonjour ».

```
import javax.swing.*;
public class Fenetre  extends JFrame {
      JButton affiche, supprime;
      JLabel message;
      public Fenetre (){
```

```
            setTitle("Ma 3ème fenêtre");
            setSize(300, 100);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            affiche = new JButton("Afficher");
            message = new JLabel("Bonjour");
            supprime = new JButton("Supprimer");
            JPanel pano = new JPanel();
            pano.add(affiche);
            pano.add(message);
            pano.add(supprime);
            setContentPane(pano);
            setVisible(true);
      }
      public static void main(String[] args){
            new Fenetre ();
      }
}
```

Output



Remarque : Dans le programme précédent, un clic sur les boutons n'a aucun effet, pour activer ces boutons de façon qu'ils affichent ou suppriment respectivement le message « Bonjour », il faut associer un écouteur (**ActionListener**) à chaque bouton et développer une méthhode « actionPerformed » qui gère ces événements.

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

public class FenetreAvecEcouteur extends JFrame implements ActionListener
{
      JButton affiche, supprime;
      JLabel message;

      public FenetreAvecEcouteur(){
            setTitle("Ma 4ème fenêtre");
            setSize(300, 150);
            setLocationRelativeTo(null);
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            affiche = new JButton("Afficher");
            affiche.addActionListener(this);
            message = new JLabel("");
            supprime = new JButton("Supprimer");
            supprime.addActionListener(this);
            JPanel pano = new JPanel();
            pano.add(affiche);
            pano.add(message);
            pano.add(supprime);
            setContentPane(pano);
            setVisible(true);
      }
```

```
    public void actionPerformed(ActionEvent e){
       if(e.getSource() == affiche){
              message.setText("Bonjour");
          } else if(e.getSource() == supprime){
              message.setText("       ");
          }
    }

    public static void main(String[] args){
         new FenetreAvecEcouteur();
    }
}
```

Exercice : Ecrire un programme qui convertit une température en °C vers le Fahrenheit en utilisant la formule F = (9/5) C + 32

Exemple


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

public class Fenetre extends JFrame implements ActionListener
{
      JLabel celLabel, fahLabel;
      JTextField celTextField, fahTextField;
      JButton clearButton, calcButton, closeButton;

      public Fenetre()        //constructeur
      {
            JLabel celLabel = new JLabel("Temp en °C :");
            JLabel fahLabel = new JLabel("Temp en °F :");
            celTextField = new JTextField(10);
            fahTextField = new JTextField(10);
            clearButton = new JButton("Nettoyer");
            clearButton.addActionListener(this);
            calcButton = new JButton("Convertir");
            calcButton.addActionListener(this);
            closeButton = new JButton("Fermer");
            closeButton.addActionListener(this);
            JPanel pano = new JPanel();
            pano.add(celLabel);  pano.add(celTextField);
            pano.add(fahLabel); pano.add(fahTextField);
            pano.add(clearButton); pano.add(calcButton);
            pano.add(closeButton);
            this.setContentPane(pano);
            this.setSize(400,100);
            this.setTitle("Conertisseur de températures");
            this.setLocationRelativeTo(null);
            this.setVisible(true);
    }

      public void actionPerformed(ActionEvent e)
    {
          if(e.getSource() == clearButton){
              celTextField.setText("  ");
              fahTextField.setText(" ");
          }
```

```
            if(e.getSource() == calcButton){
               double c =
Double.parseDouble(celTextField.getText().toString());
               double f = ((double)9/5) * c + 32;
               fahTextField.setText(String.valueOf(f));
            }
            if(e.getSource() == closeButton){
                this.dispose();
                System.exit(0);
            }
      }

      public static void main(String args[])
      {
                new Fenetre();
      }
}
```

III.3. Autres composants graphiques

Composant	Constructeurs	Principales méthodes	Gestion des événements
JButton	btn = new JButton(); btn = new JButton(“texte”);	- btn.setText(“texte”); - btn.setBorder(null);	1. implémenter l’interface **ActionListener** 2. ajouter un écouter sur le bouton : btn.**addActionListener**(this) ; 3. développer une méthode **ctionPerformed**(ActionEvent e)
JLabel	label = new JLabel(); label = new JLable(“texte”);	- label.setText("texte"); - label.setFont(Font f);	
JTextField	t = new JTextField(20);	- t.setText("texte"); - t.setFont(new Font("Arial",Font.BOLD,14));	
JTextArea	ta = new JTextArea(); ta = new JTextArea("texte");	- ta.add("texte"); - v = ta.getText();	
JCheckBox	chx = new JCheckBox("texte");	- public boolean isSelected()	1. implémenter l’interface **ItemListener** 2. ajouter un écouteur sur la case à cocher : chx.**addItemListener**(this) 3. développert une méthode: **itemStateChanged**(ItemEvent e)
JComboBox	cb = new JComboBox(); cb = new JComboBox(tabElts);	- String[] operateurs = {"+","/","*","-"} ; - listeOperateurs = new JComboBox(operateurs); - listeOperateurs.addItem("%"); - Object op = listeOperateurs.getSelectedItem(); - listeOperateurs.setEditable(true);	
JRadioButton	br = new JRadioButton("texte"); br = new JRadioButton("texte", true);	- public boolean isSelected()	1. implémenter l’interface **ItemListener** 2. ajouter un écouteur sur la case à cocher : chx.**addItemListener**(this) 3. développert une méthode: **itemStateChanged**(ItemEvent e)
ButtonGroup	bg = new ButtonGroup();	- public void add(AbstractButton b) - public void remove(AbstractButton b)	
JPasswordField	pas = new JPasswordField()	- pas.setEchoChar(char c) ;	
JList	liste = new JList(objet[] data); liste = new JList(Vector<?> v)	- String couleurs[] = {"Rouge", "Vert", "Bleu"}; - liste = new JList(couleurs); - v = liste.getSelectedValue()	1. implémenter l’interface **ListSelectionListener** 2. ajouter un écouteur sur la liste : liste.**addListSelectionListener**(this); 3. developer une méthode valueChanged(ListSelectionEvent e)
JMenu	menu = new JMenu();	- jm.setText("nomMenu") - jm.add(item)	
JMenuItem	item = new JMenuItem("Item");	- item.setText("texte"); - boolean isSelected()	1. implémenter l’interface **ActionListener** 2. ajouter un écouter sur le bouton : btn.**addActionListener**(this) ; 3. développer une méthode **actionPerformed**(ActionEvent e)
JMenuBar	bm = new JMenuBar();	- bm.add(menu);	
JOptionPane		- V = JOptionPane.showInputDialog("texte") ; - JOptionPane.showMessageDialog(null, "msg") ;	

Exercice

1. Ecrire un programme qui fait la mise en forme d'un texte selon les préférences de l'utilisateur (Gras, italique, …) faites à partir de cases à cocher.

2. Réécrire le programme précédent en utilisant 3 boutons radio (rassemblées dans un même groupe) à la place des cases à cocher.

3. Réécrire le programme précédent en utilisant une liste (de type JList) avec 4 options.