I4 - Interface graphique avec Swing

N. Prcovic

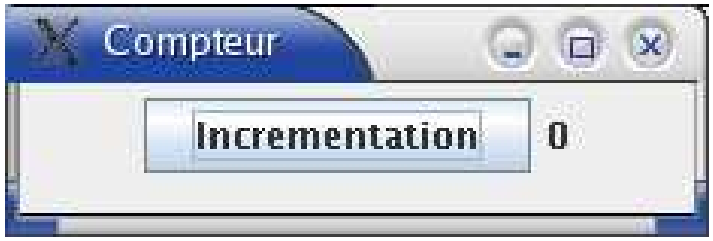
La bibliothèque Swing

- Swing est une bibliothèque de classes qui permet de créer une interface utilisateur graphique (GUI).
- `http://java.sun.com/docs/books/tutorial/uiswing/`

La bibliothèque Swing

- GUI $\rightarrow$ *programmation événementielle* :
  - On crée un programme par assemblage de différents composants possédant des capteurs d'événements.
  - On associe des méthodes aux différents événements pouvant survenir.
  - **Propriété** : on ne contrôle pas le flux d'entrée du programme (ensemble des événements et ordre).
- GUI :
  - Composants : fenêtres, boutons, champs textuels, menus, etc.
  - Evénements : appui sur un bouton, fermeture de fenêtre, entrée d'un texte dans un champ, etc).

Exemple



- A chaque fois qu'on clique sur le bouton "incrémentation", on augmente le nombre situé à droite d'une unité.
- Le programme s'arrête quand on fermera la fenêtre.

Exemple : programme (extrait)

```
class Fenetre implements ActionListener {
    private JLabel label = new JLabel("0");
    private int nbClics;

    public void actionPerformed(ActionEvent e) {
        nbClics++;
        label.setText("" + nbClics);
    }

    public void lanceFenetre() {
        JFrame frame = new JFrame("Compteur");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.getContentPane().setLayout(new FlowLayout());
        JButton bouton = new JButton("Incrementation");
        frame.getContentPane().add(bouton);
        frame.getContentPane().add(label);
        bouton.addActionListener(this);
        frame.pack();
        frame.setVisible(true);
    }
}
```

Squelette de programme conseillé

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

class Fenetre ... {
    ...
    public void lanceFenetre() {
        JFrame frame = new JFrame("Titre");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ...
        frame.pack();
        frame.setVisible(true);
    }
}
public class TesteFenetre {
    public static void main(String[] args) {
        Fenetre f = new Fenetre();
        f.lanceFenetre();
    }
}
```

Création et fermeture de fenêtre

- `JFrame frame = new JFrame("Titre");` permet de créer une fenêtre dont le titre sera "Titre".
- `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` indique que la fenêtre se fermera si on clique sur le bouton de fermeture standard des fenêtres.
- Après qu'on ait ajouté tous les composants de la fenêtre, `frame.pack();` calcule sa structure interne.
- `frame.setVisible(true);` permet de rendre la fenêtre visible.

Ajout de composants dans une fenêtre

```
frame.getContentPane().setLayout(new FlowLayout());
JButton bouton = new JButton("Incrementation");
frame.getContentPane().add(bouton);
frame.getContentPane().add(label);
```

- Les instances de `JFrame` sont composées notamment d'un *panneau "contenu"*, de type `Container`.
- Il est destiné à recevoir des composants, c'est-à-dire des instances de classes dérivées de `Component`.
- `Container` est elle-même une classe qui hérite de `Component`.
- L'accesseur `Container getContentPane()` retourne le panneau "contenu" de `frame`.

Le gestionnaire de mise en forme

- `frame.getContentPane().setLayout(new FlowLayout());` choisit le *gestionnaire de mise en forme (layout)* de type `FlowLayout` pour le panneau.
- Les gestionnaires de mise en forme déterminent la façon dont les composants seront disposés **automatiquement** sur le panneau.
- Ex : en alignant les composants horizontalement au fur et à mesure qu'ils sont ajoutés, en passant à la ligne suivante dès que le bord droit de la fenêtre est atteint.
- Lorsqu'une fenêtre est redimensionnée, les composants se repositionnent automatiquement suivant le même principe mais en fonction des nouvelles dimensions de la fenêtre.

Le gestionnaire de mise en forme

- Il existe d'autres types de "layout" :
  - `BoxLayout` : alignement de composants selon un axe horizontal ou vertical.
  - `GridLayout` : placement des composants dans une grille dont on a donné les dimensions.
  - `BorderLayout` : place les composants dans une des 5 régions (centre, nord, sud, ouest, est).
  - etc.
- La méthode `void add(Component)` permet de rajouter un composant au panneau en respectant le "layout".
- Ici, on a ajouté un bouton, puis un label, tous deux des instances de classes qui dérivent de `Component`.

Liaison événement-action

- Une interface graphique sert à :
  - recevoir des entrées via des *événements*
  - réagir à ces événements en exécutant certaines actions (des méthodes sont appelées)
  - afficher graphiquement le résultat de ces actions.
- Ex : on associe une méthode à un événement de type "action" pouvant survenir à une instance de `JButton` grâce à la méthode `void addActionListener(ActionListener)` (définie dans la classe `JButton`).
  - Le paramètre de cette méthode est une instance d'une classe implémentant l'interface `ActionListener` et définissant la méthode `public void actionPerformed(ActionEvent)`.
  - C'est cette méthode qui sera appelée lorsque qu'on appuiera sur le bouton.

Liaison événement-action

- Il est possible que plusieurs composants associent leur événement "action" à une même instance de `ActionListener`.
  - Il faut alors pouvoir déterminer quel est le composant source de l'événement à l'intérieur de la méthode `actionPerformed`.
  - C'est la méthode `Object getSource()` de la classe `ActionListener` retournant l'instance de l'objet sur lequel est apparu l'événement qui permet de le déterminer.

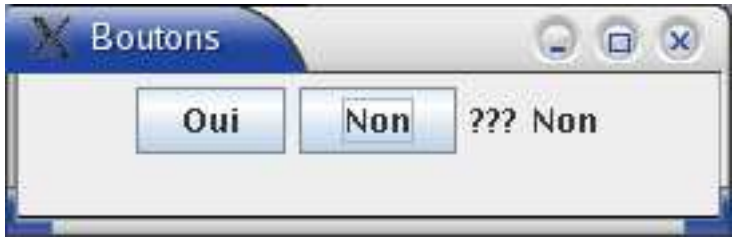
Ecouteurs d'événements

- `ActionListener` hérite de `EventListener`, qui est la racine de la hiérarchie des "event listeners".
- Ses autres sous-classes (interfaces) sont :
  - `ComponentListener` pour réagir quand un composant change de taille, de position ou de visibilité.
  - `FocusListener` pour réagir quand un composant obtient le focus ou le perd.
  - `KeyListener` pour que le composant qui a le focus réagisse à l'appui ou au relachement d'une touche.
  - `MouseListener` pour réagir quand la souris entre ou sort de l'aire d'affi chage du composant ou qu'on clique dessus, qu'on appuie ou qu'on relache un bouton de souris.
  - `MouseMotionListener` pour réagir au déplacement de la souris au dessus du composant, qu'un bouton de la souris

Ecouteurs d'événements

- Chacune de ces interfaces liste sa série de méthodes à définir en réaction aux événements.
- Le paramètre passé à l'exécution de ces méthodes est une instance d'une sous-classe de `EventObject` et permet de récupérer des informations complémentaires sur l'événement qui est survenu
  - Ex : coordonnées du pointeur de la souris lorsqu'on clique sur la souris.

Exemple



- Application graphique constituée d'une fenêtre contenant un bouton Oui, un bouton Non et deux labels.
  - Le premier label affiche le nom du bouton au-dessus duquel se trouve la souris.
  - Le second label affiche le nom du dernier bouton sur lequel on a cliqué.
- Etat actuel de l'application : le dernier bouton sur lequel on a appuyé est le bouton Non et le pointeur de la souris se trouve en dehors des zones occupés par les boutons.

Exemple

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

class Fenetre implements ActionListener, MouseListener {
    private JButton boutonOui = new JButton("Oui");
    private JButton boutonNon = new JButton("Non");
    private JLabel etat = new JLabel("????");
    private JLabel choix = new JLabel("???");

    // La méthode à défi nir quand
    // on implémente ActionListener :
    public void actionPerformed(ActionEvent e) {
        if(e.getSource() == boutonOui)
            choix.setText("Oui");
        if(e.getSource() == boutonNon)
            choix.setText("Non");
    }
```

Exemple

```
// Les 5 méthodes à défi nir quand
// on implémente MouseListener :
public void mouseEntered(MouseEvent e){
    if(e.getSource() == boutonOui)
        etat.setText("Oui");
    if(e.getSource() == boutonNon)
        etat.setText("Non");
}

public void mouseExited(MouseEvent e){
    etat.setText("???");
}

public void mouseClicked(MouseEvent e){}
public void mousePressed(MouseEvent e){}
public void mouseReleased(MouseEvent e){}
```

Exemple

```
public void lanceFenetre(){
    JFrame frame = new JFrame("Boutons");
    frame.setDefaultCloseOperation(
        JFrame.EXIT_ON_CLOSE);

    Container panneau = frame.getContentPane();
    panneau.setLayout(new FlowLayout());
    panneau.add(boutonOui);
    panneau.add(boutonNon);
    panneau.add(etat);
    panneau.add(choix);

    boutonOui.addActionListener(this);
    boutonOui.addMouseListener(this);
    boutonNon.addActionListener(this);
    boutonNon.addMouseListener(this);
```

Exemple

```
        frame.pack();
        frame.setVisible(true);
    }
}

public class TesteFenetre {
    public static void main(String[] args) {
        Fenetre f = new Fenetre();
        f.lanceFenetre();
    }
}
```