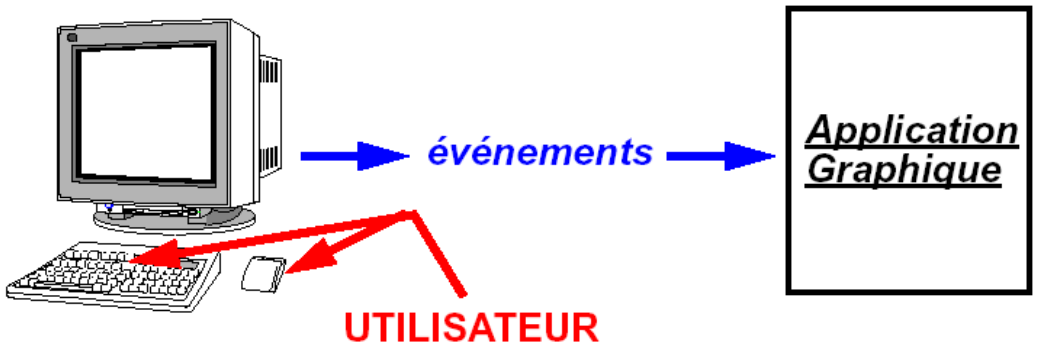
Interface graphiques

La gestion des événements



Les événements graphiques

- L'utilisateur effectue
  - une action au niveau de l'interface utilisateur (clic souris, sélection d'un item, etc)
  - alors un **événement graphique** est émis.
- Lorsqu'un événement se produit
  - il est reçu par le composant avec lequel l'utilisateur interagit (par exemple un bouton, un curseur, un champ de texte, etc.).
  - Ce composant transmet cet événement à un autre objet, un **écouteur** qui possède une méthode pour traiter l'événement
    - ➢ cette méthode reçoit l'objet événement généré de façon à traiter l'interaction de l'utilisateur.

Les événements graphiques

- La gestion des événements passe par l'utilisation d'objets (***Listener***) et d'objets sources d'événements.
  - Un objet écouteur est l'instance d'une classe implémentant l'interface ***EventListener***.
  - Une source d'événements est un objet pouvant recenser des objets écouteurs et leur envoyer des objets événements.
- Lorsqu'un événement se produit,
  - la source d'événements envoie un objet événement correspondant à tous ses écouteurs.
  - Les objets écouteurs utilisent alors l'information contenue dans l'objet événement pour déterminer leur réponse.

Les événements graphiques

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

class MonAction implements ActionListener {
        public void actionPerformed (ActionEvent e)  {
                System.out.println ("Une action a eu lieu") ;
        }
}

public class TestBouton {
    public TestBouton(){
        JFrame f = new JFrame ("TestBouton");
        JButton b = new JButton ("Cliquer ici");
        f.add (b) ;
        f.pack (); f.setVisible (true) ;
        b.addActionListener (new MonAction ());
    }
    public static void main(String args[]) {
        TestBouton test = new TestBouton();}
}
```

Les événements graphiques

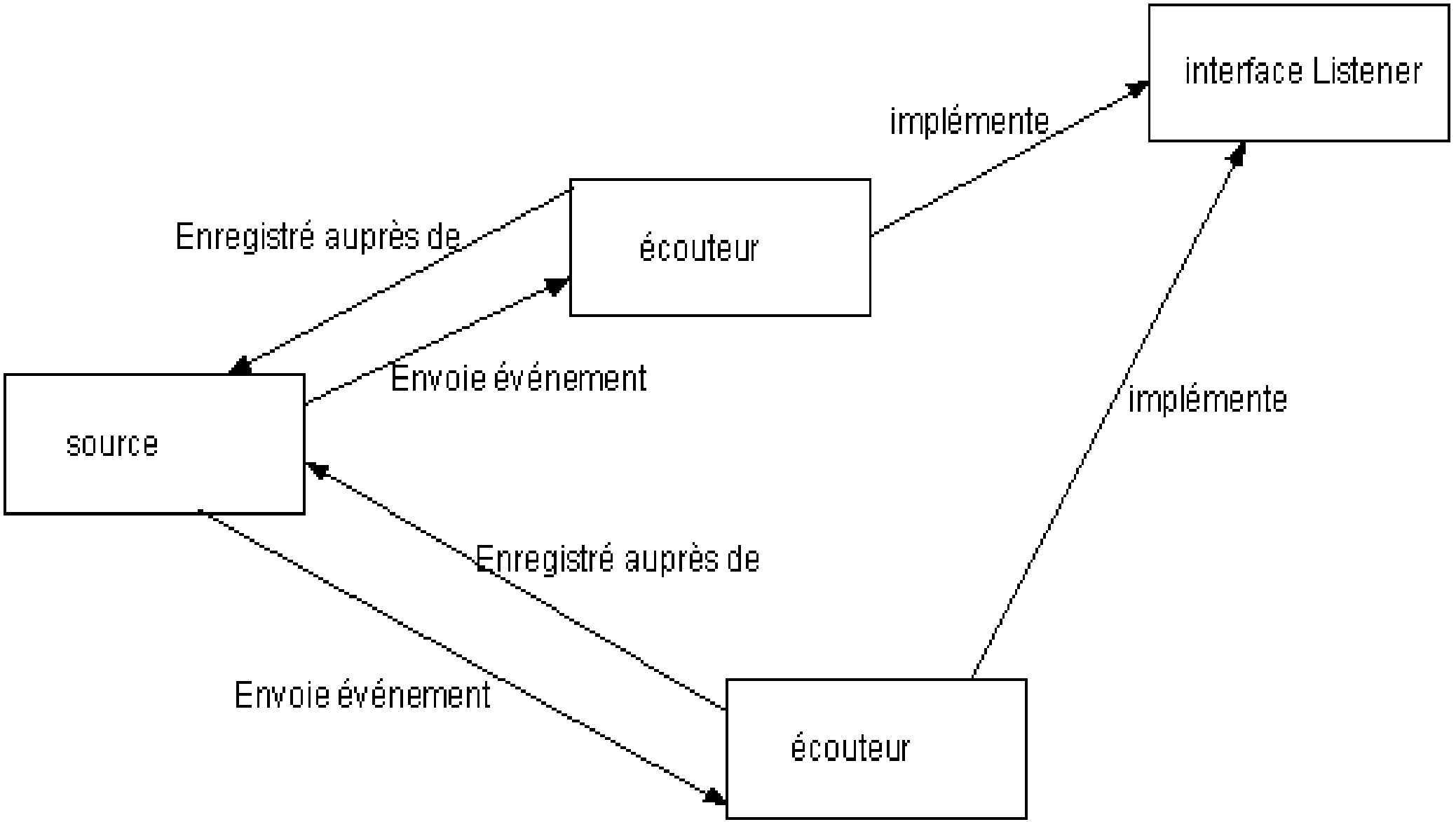
- Les écouteurs sont des interfaces
- Donc une même classe peut implémenter plusieurs interfaces écouteur.
  - ♦ Par exemple une classe héritant de Frame implémentera les interfaces **MouseMotionListener** (pour les déplacements souris)

    MouseListener (pour les clics souris).
- Chaque composant de l'swing est conçu pour être la source d'un ou plusieurs types d'événements particuliers.
  - ♦ Cela se voit notamment grâce à la présence dans la classe de composant d'une méthode nommée **addXXXListener()**.

Les événements graphiques

- L'objet événement envoyé aux écouteurs et passé en paramètres des fonctions correspondantes peut contenir des paramètres intéressants pour l'application.
  - Par exemple, **getX()** et **getY()** sur un **MouseEvent** retournent les coordonnées de la position du pointeur de la souris.
  - Une information généralement utile quelques soit le type d'événement est la source de cet événement que l'on obtient avec la méthode **getSource()**.

Les événements graphiques



Catégories d'événements graphiques

- Plusieurs types d'événements sont définis dans le package **java.awt.event.**
- Pour chaque catégorie d'événements, il existe une interface qui doit être définie par toute classe souhaitant recevoir cette catégorie d'événements.
  - ♦ Cette interface exige aussi qu'une ou plusieurs méthodes soient définies.
  - ♦ Ces méthodes sont appelées lorsque des événements particuliers surviennent.

Catégories d'événements graphiques

Catégorie	Interface	Méthodes
Action	**ActionListener**	**actionPerformed (ActionEvent)**
Item	**ItemListener**	**itemStateChanged (ItemEvent)**
Mouse	**MouseMotionListener**	**mouseDragged (MouseEvent)** **mouseMoved (MouseEvent)**
Mouse	**MouseListener**	**mousePressed (MouseEvent)** **mouseReleased (MouseEvent)** **mouseEntered (MouseEvent)** **mouseExited(MouseEvent)** **mouseClicked(MouseEvent)**
Key	**KeyListener**	**keyPressed (KeyEvent)** **keyReleased (KeyEvent)** **keyTyped (KeyEvent)**
Focus	**FocusListener**	**focusGained (FocusEvent)** **focusLost (FocusEvent)**

Catégories d'événements graphiques

AdjustmentEvent	AdjustmentListener	adjustmentValueChanged (AdjustmentEvent)
ComponentEvent	ComponentListener	componentMoved (ComponentEvent) componentHidden (ComponentEvent) componentResized (ComponentEvent) componentShown (ComponentEvent)
WindowEvent	WindowListener	windowClosing (WindowEvent) windowOpened (WindowEvent) windowIconified (WindowEvent) windowDeiconified (WindowEvent) windowClosed (WindowEvent) windowActivated (WindowEvent) windowDeactivated (WindowEvent)
ContainerEvent	ContainerListener	componentAdded (ContainerEvent) componentRemoved (ContainerEvent)
TextEvent	TextListener	textValueChanged (TextEvent)

Catégories d'événements graphiques

ActionListener : Action clic sur un bouton, retour chariot dans une zone de texte, « tic d'horloge »

WindowListener : Fermeture, iconisation, etc. des fenêtres

TextListener : Changement de valeur dans une zone de texte

ItemListener : Sélection d'un item dans une liste

FocusListener: Pour savoir si un élément a le "focus"

KeyListener: Pour la gestion des événements clavier

Catégories d'événements graphiques

MouseListener :Clic, enfoncement/relâchement des boutons de la souris, etc.

MouseMotionListener :Déplacement de la souris, drag&drop avec la souris, etc.

AdjustmentListener:Déplacement d'une échelle

ComponentListener: Savoir si un composant a été caché, affiché …

ContainerListener: Ajout d'un composant dans un Container

Catégories d'événements graphiques

```
import java.swing.*;
import java.awt.event.*;
public class Test extends JFrame implements ActionListener
{
        public void actionPerformed( ActionEvent e )
        {
                setTitle("bouton cliqué !");
         }

        public Test()
        {
                super("Utilisation d'un ActionEvent");
                JButton b = new JButton("action");
                b.addActionListener(this);
                add(BorderLayout.CENTER,b);pack();show();
        }

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

Catégories d'événements graphiques

```
public class Test extends JFrame implements ActionListener{
        private JButton b1,b2;
        public static void main(String args[]){
                Test f= new Test();
        }
        public Test(){
                super("Utilisation d'un ActionEvent");
                b1 = new JButton("action1");
                b2 = new JButton("action2");
                b1.addActionListener(this);
                b2.addActionListener(this);
                add(BorderLayout.CENTER,b1);
                add(BorderLayout.SOUTH,b2);
                pack();show();
        }
        public void actionPerformed( ActionEvent e ) {
                if (e.getSource() == b1) setTitle("action1 cliqué");
                if (e.getSource() == b2) setTitle("action2 cliqué");
        }
}
```

Catégories d'événements graphiques

```
import java.swing.*;
import java.awt.event.*;
public class Test extends JFrame  implements WindowListener{
        public static void main(String[] args) {
                Test f= new Test();
        }
        public WinEvt() {
                super("Cette fenêtre se ferme");
                addWindowListener(this);
                pack();show();
        }
        public void windowClosing(WindowEvent e){
                System.exit(0);
        }
        public void windowOpened(WindowEvent e){}
        public void windowClosed(WindowEvent e){}
        public void windowIconified(WindowEvent e){}
        public void windowDeiconified(WindowEvent e){}
        public void windowActivated(WindowEvent e){}
        public void windowDeactivated(WindowEvent e){} }
```

Les adapteurs

Les classes « **adapteur** » permettent une mise en œuvre simple de l'écoute d'événements graphiques.

Ce sont des classes qui implémentent les écouteurs d'événements possédant le plus de méthodes, en définissant un corps vide pour chacune d'entre elles.

Plutôt que d'implémenter l'intégralité d'une interface dont une seule méthode est pertinente pour résoudre un problème donné, une alternative est de sous-classer l'adapteur approprié et de redéfinir juste les méthodes qui nous intéressent.

- Par exemple pour la gestion des événements fenêtres...

Les adapteurs

Solution en implémentant l'interface

```
class Test implements WindowListener
{
        public void windowClosing (WindowEvent e)
        {
                System.exit(0);
        }
        public void windowClosed (WindowEvent e)
        {}
        public void windowIconified (WindowEvent e)
        {}
        public void windowOpened (WindowEvent e)
        {}
        public void windowDeiconified (WindowEvent e)
        {}
        public void windowActivated (WindowEvent e)
        {}
        public void windowDeactivated (WindowEvent e)
        {}
}
```

Solution en utilisant un WindowAdapter

```
class Test extends WindowAdapter
{
        public void windowClosing (WindowEvent e)
        {
                System.exit(0);
        }
}
```

Les adapteurs

Il existe 7 classes d'adapteurs (autant que d'interfaces d'écouteurs possédant plus d'une méthode) :

1. **ComponentAdapter**
2. **ContainerAdapter**
3. **FocusAdapter**
4. **KeyAdapter**
5. **MouseAdapter**
6. **MouseMotionAdapter**
7. **WindowAdapter**

Les adapteurs

En pratique, et notamment avec la classe WindowAdapter, on utilise très souvent une classe anonyme

```
JFrame f = new JFrame("TEST");

f.addWindowListener(
        new WindowAdapter()
        {
                public void windowClosing(WindowEvent e)
                {
                        System.exit(0);
                }
        }
);
```

Maquette, prototype et interface finale (gestionnaire de paquets Ubuntu)