

# Gérer les événements I

- Le clic sur une souris, un bouton, ... constitue un événement qui est représenté par :
  - un objet du package **java.awt.event**
- Java gère les événements comme suit :
  - 1 l'objet **source** transmet l'événement à l'objet écouteur qui effectue le traitement approprié correspondant à l'événement.
  - 2 l'objet source doit être associé à l'objet écouteur **add..(...)**
  - 3 la classe écouteur doit implémenter l'une des interface **xxxListener** ou **xxx** renvoie à **Action,Mouse,Window...**

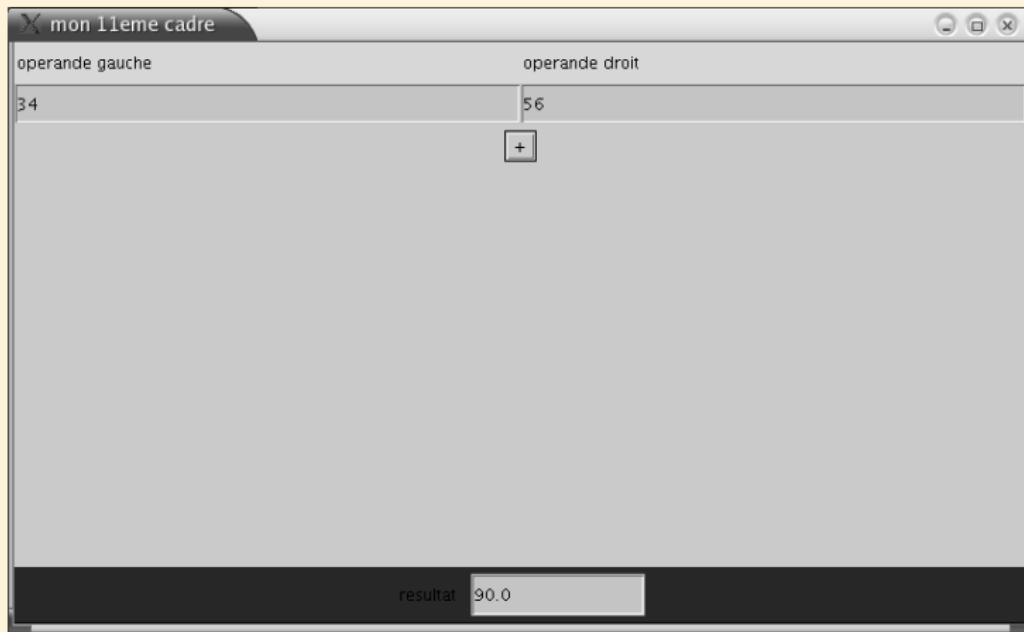
# Les événements familiers I

- Les événements les plus courants sont le clic sur :
  - 1 un bouton **JButton** correspond à l'événement **ActionEvent**
  - 2 une zone **JPanel** correspond à l'événement **MouseEvent**
  - 3 le clic sur le bouton d'ouverture et de fermeture de l'objet **JFrame** crée un événement de type **WindowEvent**
- Tous les événements héritent de la classe **EventObject**
  - **public void EventObject(Object source)** : un constructeur
  - **public Object getSource()** : rend l'émetteur.
  - **public String toString()** : nom de la classe

# Autre type d'événements I

- 1 **FocusEvent** ⇒ lorsqu'un composant gagne ou perd le focus
- 2 **KeyEvent** ⇒ lorsqu'une touche du clavier est enfoncée alors que le composant a le focus.
- 3 **ComponentEvent** ⇒ lorsqu'un composant est déplacé, redimensionné, rendu visible ou invisible.
- 4 **ItemEvent** ⇒ spécifique aux listes, *choice*, *checkBox* , signale si un élément est sélectionné ou pas.
- 5 **TextEvent** ⇒ spécifique au **TextField**, **TextArea** lorsque le texte est modifié

# Reprendons l'exemple 1



# Reprendons l'exemple I

- On veut que le cadre réagisse à certaines actions de l'utilisateur :
  - 1 lorsque l'utilisateur ferme la fenêtre grâce au commutateur,  
⇒ le programme doit se terminer
  - 2 lorsque l'utilisateur clique sur le bouton "+"  
⇒ le programme doit calculer la somme des nombres placés dans les champs opérande gauche et opérande droit et placer le résultat dans le champ de texte résultat.

# Gestion de la terminaison du programme I

Pour gérer la terminaison du programme, il faut :

- une classe **Terminaison** qui hérite de l'interface **WindowListener**
- dans la classe Terminaison donner un corps à la fonction **windowClosing(?)**

```
1 class Terminaison extends WindowAdapter{  
2     public Terminaison()  
3     {super();}  
4     public void windowClosing(WindowEvent e)  
5     {System.exit(0);}  
6 }
```

# Gestion de la terminaison du programme I

- instancier la classe Terminaison
- indiquer quelle instance doit écouter les évènements fenêtre du cadre (avec **addWindowListener(?)**)

```
1 class Cadre extends Frame{  
2     Terminaison fin;  
3     public Cadre(String titre){  
4         .....  
5         fin = new Terminaison();  
6         addWindowListener(fin);  
7     }
```

- Ou bien plus simplement

```
1 JFrame jf=new JFrame();  
2 jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

# Gestion du bouton I

Pour gérer les évènements du bouton "+", il faut :

- 1 Définir une classe qui **implémente** l'interface **ActionListener** ⇒ on choisit la classe Cadre
- 2 Indiquer quelle instance doit écouter les évènements du bouton avec la fonction **addActionListener()**
- 3 Donner un corps à la fonction **ActionPerformed()** dans la classe dérivée  
⇒ elle sera appelée lorsque l'évènement Bouton se produira

# Gestion du bouton I

- Mon cadre doit **implémenter** l'interface **ActionListener**

```
1 class Cadre extends Frame implements ActionListener{  
2     PanneauNord panneauNord;  
3     PanneauCentre panneauCentre;  
4     PanneauSud panneauSud;
```

- L'objet **source** d'événement est associé à l'objet **écouteur**.

```
1 public Cadre(String titre){  
    .....  
3     panneauCentre.bouton.addActionListener(this);  
}
```

# Gestion du bouton II

- **action exécutée** si un événement **ActionEvent** se produit

```
public void actionPerformed(ActionEvent e){  
    //recupere l'objet ayant produit l'évenement  
    Object s=e.getSource();  
    if (s==panneauCentre.bouton){  
        double xg, xd, r;  
        String sg, sd, sr;  
        sg=panneauNord.operandeGauche.getText().trim();  
        sd=panneauNord.operandeDroit.getText().trim();  
        xg=new Double(sg).doubleValue();  
        xd=new Double(sd).doubleValue();  
        r=xg+xd;  
        sr=new Double(r).toString();  
        panneauSud.resultat.setText(sr);}  
}
```

# Les événements de la souris : MouseEvent

- ⇒ Un événement souris produit une instance de **MouseEvent**.
- ⇒ Java distingue deux types d'évènements de souris :

## 1 Des événements transmis par les méthodes de l'interface **MouseListener**

- **mouseClicked** ⇒ clic souris sur le composant
- **mousePressed** ⇒ bouton de la souris appuyé
- **mouseReleased** ⇒ bouton de la souris relaché
- **mouseEntered** ⇒ la souris entre dans le composant
- **mouseExited** ⇒ la souris sort du composant

## 2 Des événements transmis par les méthodes de l'interface **MouseMotionListener**

- **mouseMoved** ⇒ souris déplacé sur le composant
- **mouseDragged** ⇒ souris déplacé avec un bouton enfoncé

# Gestion d'évènements souris I

Pour écouter les **événements** produits par la souris il faut

- 1 définir une classe qui **implémente** l'interface  
**MouseListener**
- 2 indiquer avec **addMouseListener(?)** quelle instance de cette classe va écouter la souris.
- 3 cette classe doit donner un corps aux cinq fonctions
  - **mouseClicked(MouseEvent e)**,
  - .....
  - **mouseExited(MouseEvent e)**

qui contiendront le traitement approprié à l'événement.

# Gestion d'évènements souris II

Pour écouter les **mouvements** de la souris il faut

- 1 définir une classe qui implémente l'interface  
**MouseMotionListener**
- 2 indiquer avec **addMouseMotionListener(?)** quelle instance de cette classe va écouter les mouvements de la souris.
- 3 cette classe doit donner un corps aux deux fonctions
  - **mouseDragged(MouseEvent e)** et
  - **mouseMoved(MouseEvent e)**qui contiendront le traitement approprié à l'événement.

# Exemple d'événement de la souris I

- Ma classe doit **implémenter** l'interface **MouseListener**

```
import java.awt.*;  
2 import java.awt.event.*;  
class Cadre extends Frame implements MouseListener{
```

- Un double click est intercepté avec clickcount à 1 puis le second à 2

```
1 public void mouseClicked(MouseEvent ev){  
    System.out.println("Souris clique\u00e7e en : " +  
3     +ev.getX() + " par " + ev.getY() + " avec "  
     +ev.getClickCount() + " click");  
5 }
```

# Exemple d'événement de la souris II

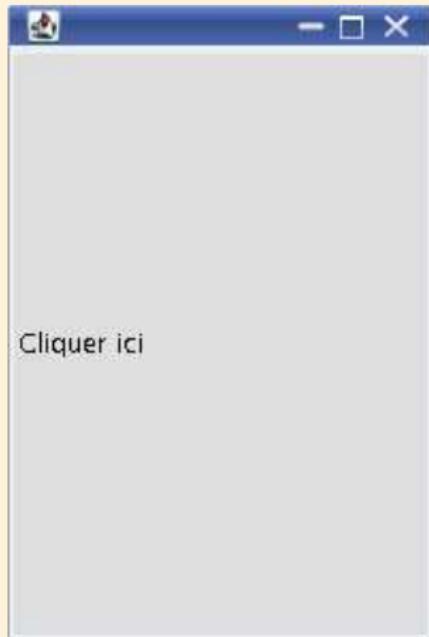
- Il faut **implémenter** toutes les autres méthodes

```
1 public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
3 public void mousePressed(MouseEvent e) {}
  public void mouseReleased(MouseEvent e) {}
```

- Mon **Cadre écoute les événements** produits par la souris

```
public static void main( String [] arg){
2 Cadre ts = new Cadre();
  Label l=new Label("Cliquer_ici");
4 ts.add(l);
  l.addMouseListener(ts);
6 ts.setBounds(0,0, 200, 300);
  ts.setVisible(true);}
8 }
```

# Exemple d'événement de la souris I



## Exemple d'événement de la souris I

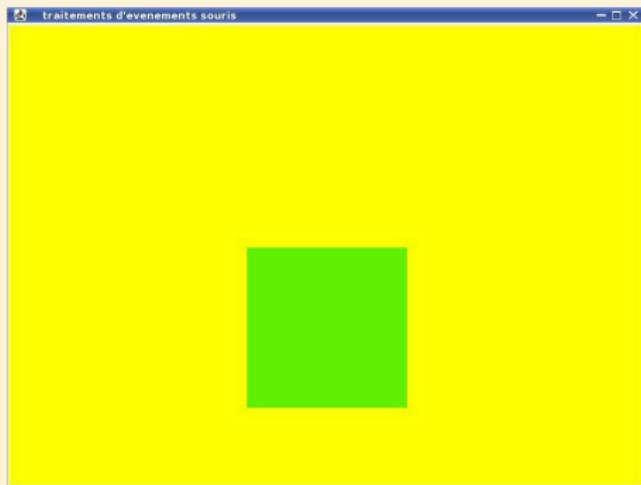
Le programme suivant gère que deux évènements souris :

- 1 lorsque l'utilisateur **clique sur la souris** dans une zone rectangulaire de la fenêtre, le **rectangle change de couleur** ;
- 2 lorsque la **souris se déplace** à l'intérieur de ce rectangle, le **curseur change de forme**.

2 évènements sont traités :

- un clic de souris
- la souris est déplacée

# Exemple d'événement de la souris I



# Exemple d'événement de la souris I

- le cadre **implémente** les interfaces **MouseListener** et **MouseMotionListener**

```
1 class CadreSouris extends Frame implements MouseListener,  
2 MouseMotionListener{  
    Rectangle rectangle;
```

- Et voici mon constructeur de cadre

```
1 public CadreSouris(String titre){  
    super(titre);  
3    this.setBackground(Color.yellow);  
    this.setLocation(100,100);  
5    this.setSize(800,600);  
    rectangle = new Rectangle(300,300, 200, 200);
```

- le cadre est un écouteur d'événements et de mouvements de la souris

# Exemple d'événement de la souris II

```
1 addMouseListener(this);  
2 addMouseMotionListener(this);  
}
```

- crée une **couleur au hasard et remplit le rectangle**

```
1 public void paint(Graphics g){  
2     g.setColor(new Color((float)Math.random(),  
3     (float)Math.random(), (float)Math.random()));  
4     g.fillRect(rectangle.x,rectangle.y,  
5                 rectangle.width,rectangle.height);  
6 }
```

- fonction **appelée lorsque l'utilisateur clique la souris**

```
1 public void mouseClicked(MouseEvent e){  
2     int xSouris, ySouris;  
3     xSouris = e.getX(); ySouris = e.getY();  
4     // l'utilisateur a cliqué à l'intérieur du rectangle  
5     if (rectangle.contains(xSouris,ySouris)) repaint();  
6 }
```

# Exemple d'événement de la souris III

- il est obligatoire de donner un corps à ces fonctions, sinon **CadreSouris** reste une classe abstraite

```
1 public void mouseEntered(MouseEvent e) {}  
2 public void mouseExited(MouseEvent e) {}  
3 public void mousePressed(MouseEvent e) {}  
4 public void mouseReleased(MouseEvent e) {}  
5 public void mouseDragged(MouseEvent e) {}
```

- cette fonction est **appelée** lorsque la **souris bouge**

```
1 public void mouseMoved(MouseEvent e){  
    int xSouris, ySouris;  
    xSouris = e.getX(); ySouris = e.getY();  
    if (rectangle.contains(xSouris,ySouris))  
        setCursor(new Cursor(Cursor.CROSSHAIR_CURSOR));  
    else setCursor(Cursor.getPredefinedCursor(Cursor.  
        DEFAULT_CURSOR));  
7 }
```

## Exemple d'événement de la souris IV

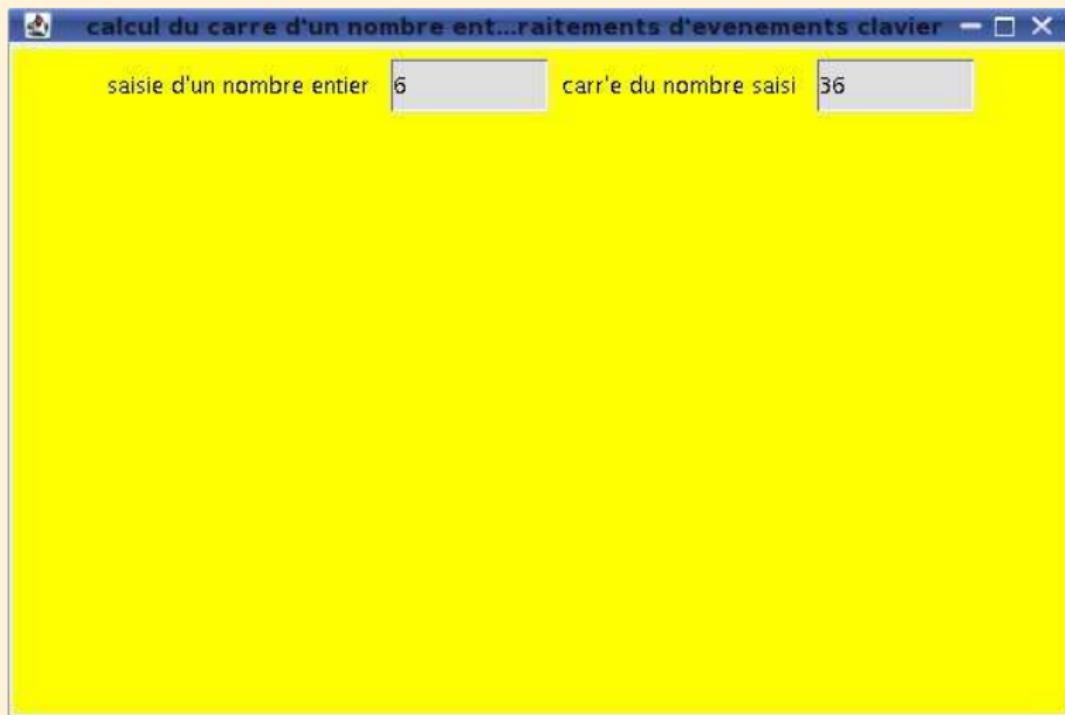
```
1 class TestSouris{  
2     public static CadreSouris cadre;  
3     public static void main( String [] arg){  
4         cadre = new CadreSouris("traitements_d'evenements_souris");  
5         cadre.setVisible(true);  
6     }  
7 }
```

# Gestion des évènements clavier I

On souhaite écrire un programme :

- calcule le carré d'un nombre entier. Il réagit si des chiffres sont tapés,
  - 1 si la touche **ENTREE** → le calcul est exécuté
  - 2 si la touche **Echap** → le programme s'arrête
  - 3 si la touche **Suppr** → les champs de texte sont vidés
- à un instant donné, seul le composant qui a le focus peut recevoir un événement clavier ; ⇒ il faut donc que chaque composant indique quel est son écouteur d'évènements clavier.
- La fenêtre principale est généralement choisie comme écouteur des évènements clavier.

# Exemple d'événement du clavier I



# Exemple :événements clavier I

- le cadre écoute les événements clavier implémente l'interface **KeyListener**

```
import java.awt.*;
2 import java.awt.event.*;
class CadreClavier extends Frame implements KeyListener{
4 Label etiquetteEntree, etiquetteSortie;
    TextField entree; //saisir le nombre
6 TextField sortie; //afficher le carré du nombre
public CadreClavier(String titre){
8     super(titre);
    this.setBackground(Color.yellow);
10    this.setLocation(100,100);
    this.setSize(600,400);
12    setLayout(new FlowLayout());
```

- création de l'étiquette du champ de saisie

```
etiquetteEntree=new Label("saisie_du_nombre_entier");
2 this.add(etiquetteEntree);
```

# Exemple :événements clavier II

## ■ création du champ de saisie

```
1 entree = new TextField("",10);  
2 this.add(entree);  
entree.addKeyListener(this);
```

## ■ création de l'étiquette du champ résultat

```
1 etiquetteSortie=new Label("carre du nombre saisi");  
this.add(etiquetteSortie);
```

## ■ création du champ résultat

```
1 sortie = new TextField("",10);  
2 sortie.setEditable(false);  
this.add(sortie);  
4 sortie.addKeyListener(this);
```

## ■ le cadre peut écouter ses propres événements clavier

## Exemple :événements clavier III

```
    addKeyListener(this);  
2 }
```

- pour que le cadre puisse recevoir le focus et donc les événements clavier

```
public boolean isFocusTraversable() {return true;}
```

# Exemple :événements clavier IV

## ■ appelée lorsque l'utilisateur tape un caractère

```
1 public void keyTyped(KeyEvent e){  
    //recupere le caractere tape  
3     char c = e.getKeyChar();  
4     if (('0' <= c) && (c <= '9')){  
5         String s;  
6         // je recuper la chaine dans entree  
7         s = entree.getText();  
8         // j'y ajoute le caractere chifre  
9         entree.setText(s+c);  
10    }  
11 }
```

```
1 public void keyReleased(KeyEvent e) {}
```

# Exemple :évènements clavier V

- fonction **appelée** lorsque l'utilisateur **frappe** une touche

```
1 public void keyPressed(KeyEvent e){  
2     int code;  
3     code=e.getKeyCode(); //recupere le code de la touche  
4     if (code==KeyEvent.VK_ESCAPE){System.exit(0);}  
5     else  
6         if (code==KeyEvent.VK_DELETE)  
7             {entree.setText(""); sortie.setText("");}  
8             if (code==KeyEvent.VK_ENTER){  
9                 String s1, s2; int x1,x2;  
10                s1=entree.getText();  
11                x1=new Integer(s1).intValue();  
12                x2=x1*x1;  
13                s2=Integer.toString(x2);  
14                sortie.setText(s2);}  
15 }
```

# Exemple :événements clavier VI

```
1 class TestClavier{
2     public static CadreClavier cadre;
3     public static void main( String [] arg){
4         cadre = new CadreClavier("calcul du carre d'un nombre
5                                     entier: traitements d'evenements clavier");
6         cadre.setVisible(true);
7     }
8 }
```

# Exemple : Gestion des cases à cocher : JCheckBox I

Voici un exemple simple de cadre qui comporte deux événements de type **ActionListener**

- 1 deux cases à cocher
- 2 un bouton qui donne l'état des cases à cocher ( true ou false)

```
1 import java.awt.*;
import javax.swing.*;
3 import java.awt.event.*;
5 class MaFenetre extends JFrame implements ActionListener{
    JCheckBox coche1, coche2;
    JButton etat;
```

## Exemple : Gestion des cases à cocher : JCheckBox II

```
1 public MaFenetre(){  
2     setTitle("Exemple_de_case_a_cocher");  
3     setSize(400,100);  
4     setLayout(new FlowLayout());  
5     coche1=new JCheckBox("case_1"); add(coche1);  
6     coche1.addActionListener(this);  
7     coche2=new JCheckBox("case_2"); add(coche2);  
8     coche2.addActionListener(this);  
9     etat=new JButton("ETAT"); add(etat);  
10    etat.addActionListener(this);  
11}
```

```
1 public void actionPerformed(ActionEvent ev){  
2     Object source = ev.getSource();  
3     if (source==coche1)System.out.println("action_case1");  
4     if (source==coche2)System.out.println("action_case2");  
5     if (source==etat) System.out.println("Etat_des_case:"+coche1  
6         .isSelected()+" "+coche2.isSelected());}  
7 }
```

# Exemple : Gestion des cases à cocher : JCheckBox III



# Exemple : Cases à cocher (suite) I

- Le cadre qui comporte deux types d' événements
  - 1 **ActionListener** pour le bouton
  - 2 **ItemListener** pour les cases à cocher

```
import java.awt.*;
2 import javax.swing.*;
import java.awt.event.*;
4 class MaFenetre extends JFrame implements ActionListener,
    ItemListener
{
6     JCheckBox coche1, coche2;
    JButton etat;
```

# Exemple : Cases à cocher (suite) II

## ■ Voici mon constructeur

```
1 public MaFenetre(){  
2     setTitle( "Exemple_de_case_a_cocher" );  
3     setSize(400,100);setLayout(new FlowLayout());  
4  
5     coche1=new JCheckBox( "case_1" ); add(coche1);  
6     coche1.addItemListener(this);  
7     coche1.setSelected(true);  
8  
9     coche2=new JCheckBox( "case_2" ); add(coche2);  
10    coche2.addItemListener(this);  
11  
12    etat=new JButton( "ETAT" ); add(etat);  
13    etat.addActionListener(this);  
14}
```

# Exemple : Cases à cocher (suite) III

## ■ fonction appelée lorsque une case est cochée

```
public void itemStateChanged(ItemEvent ev){  
1 Object source = ev.getSource();  
2 if (source==coche1) System.out.println("act_case1");  
3 if (source==coche2) System.out.println("act_case2");  
4 }
```

## ■ fonction appelée lorsque le bouton est appuyé

```
public void actionPerformed(ActionEvent ev){  
1 Object source=ev.getSource();  
2 if (source==etat) System.out.println("Etat_des_cases:"+  
    coche1.isSelected()+" "+coche2.isSelected());  
3 }  
4 }
```

# Exemple d'événement JCheckBox I



# Exemple 17 : Gestion des boutons radios I

Voici un exemple très simple de création d'un groupe de boutons radios :

- 1 3 boutons radios
- 2 un bouton pour l'état des boutons radios
- 3 deux événements se produisent à chaque sélection , un de type **ActionEvent** et un de type **ItemEvent**

```
import java.awt.*;  
2 import javax.swing.*;  
import java.awt.event.*;  
4 class MaFenetre extends JFrame implements ActionListener,  
    ItemListener  
{  
6 JRadioButton radiol, radio2, radio3;  
JButton etat;
```

# Exemple 17 : Gestion des boutons radios II

```
1 public MaFenetre(){  
    setTitle("Exemple_de_case_a_cocher");  
3    setSize(400,100);  
    setLayout(new FlowLayout());
```

- On Crée un ButtonGroup

```
ButtonGroup groupe=new ButtonGroup();
```

- on crée le premier bouton radio et on le met dans le groupe, on l'accroche à la fenêtre

```
1 radiol=new JRadioButton("Radio_1");  
groupe.add(radiol); // juste pour la desactivation  
3 add(radiol);
```

- deux événements ItemEvent et ActionEvent

# Exemple 17 : Gestion des boutons radios III

```
1 radio1.addItemListener(this);
  radio1.addActionListener(this);
3 radio1.setSelected(true);
```

## ■ On ajoute les autres boutons radio et le bouton

```
1 radio2 = new JRadioButton("Radio_2");
  .....
3 // un bouton pour l'etat
  etat=new JButton("ETAT");
5 add(etat);
  etat.addActionListener(this);
7 }
```

```
1 public void itemStateChanged(ItemEvent ev){
  Object s=ev.getSource();
3 if (s==radio1) System.out.println("changt_radio_1");
  if (s==radio2) System.out.println("changt_radio_2");
5 if (s==radio3) System.out.println("changt_radio_3");}
```

## Exemple 17 : Gestion des boutons radios IV

```
1 public void actionPerformed(ActionEvent ev){  
    Object s = ev.getSource();  
3    if (s==etat) System.out.println("Etat des radios:"+radio1.  
        isSelected()+" "+radio2.isSelected()+" "+radio3.  
        isSelected());  
    if (s==radio1) System.out.println("action_radio_1");  
5    if (s==radio2) System.out.println("action_radio_2");  
    if (s==radio3) System.out.println("action_radio_3");}  
7 }
```

```
1 class TestRadio1  
{ public static void main(String args[]){  
3     MaFenetre f=new MaFenetre();  
    f.setVisible(true);}  
5 }
```

# Exemple 18 : Gestion d'une Liste I

Voici un exemple très simple de gestion d'une liste : **JList**

```
1 import java.awt.*;
import javax.swing.*;
3 import java.awt.event.*;
import javax.swing.event.*;
5 class MaFenetre extends JFrame implements ListSelectionListener
{
7 String[] couleurs={"rouge","bleu","gris","vert","jaune","noir"};
JList Liste;
```

```
public MaFenetre(){
2 setTitle("Exemple_de_case_a_cocher");
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
3 setLayout(new FlowLayout());
// Cree la liste
6 Liste =new JList(couleurs);
add(Liste);
8 Liste.addListSelectionListener(this);
}
```

## Exemple 18 : Gestion d'une Liste II

```
1 public void valueChanged(ListSelectionEvent ev){  
2     if(!ev.getValueIsAdjusting()){  
3         System.out.println("action_Liste_-_valeurs_selectionnes");  
4         Object[] valeurs = Liste.getSelectedValues();  
5         for(int i=0; i<valeurs.length;i++)  
6             System.out.println((String) valeurs[i]);} }  
7 }
```

```
1 class TestList1{  
2     public static void main(String args[]){  
3         MaFenetre f=new MaFenetre();  
4         f.setVisible(true);}  
5     }
```

# Exemple d'événement JCheckBox I



## Exemple 18 : Gestion d'une boite Combo I

Voici un exemple très simple de gestion d'une **ComboBox** qui met en évidence les événements Action et Item générées par une boîte combo dont le champ de texte est editable.

```
1 import java.awt.*;
  import javax.swing.*;
3 import java.awt.event.*;
  import javax.swing.event.*;
5 class MaFenetre extends JFrame implements ActionListener,
    ItemListener
{
7 String[] couleurs={"rouge","bleu","gris","vert","jaune","noir"};
  JComboBox combo;
```

# Exemple 18 : Gestion d'une boite Combo II

## ■ Voici mon constructeur

```
public MaFenetre(){  
2   setTitle("Exemple_de_case_a_cocher");  
3   setSize(300,200);  
4   setLayout(new FlowLayout());  
5   // Cree le Combobox  
6   combo =new JComboBox(couleurs);  
7   combo.setEditable(true);  
8   add(combo);  
9   combo.addActionListener(this);  
10  combo.addItemListener(this);  
}
```

# Exemple 18 : Gestion d'une boite Combo III

## ■ fonction **actionPerformed**

```
1 public void actionPerformed(ActionEvent ev){  
    System.out.println("action_combo");  
3 Object valeur = combo.getSelectedItem();  
    System.out.println((String) valeur);}
```

## ■ fonction **itemStateChanged**

```
public void itemStateChanged(ItemEvent ev){  
2 System.out.println("item_combo");  
    Object valeur = combo.getSelectedItem();  
4 System.out.println((String) valeur);}  
    }
```

## Exemple d'événement JCheckBox I



## Exemple 19 : Gestion d'une boite de dialogue I

Voici un exemple très simple de gestion d'une boite de dialogue :

- L'action d'un bouton affiche une boite de dialogue pour saisir un texte
- la boite de dialogue est créée et libérée à chaque fois dans la méthode **actionPerformed**
- le déroulement du dialogue est géré par une méthode **lanceDialogue** à définir
- Les résultats (texte saisi ou dialogue abandonné s'affichent sur la console.

# Exemple 19 : Gestion d'une boite de dialogue II

```
1 import javax.swing.*;
import java.awt.*;
3 import java.awt.event.*;
class Fenetre extends JFrame implements ActionListener{
5 JButton lanceDial;
String texte;
```

## ■ constructeur

```
public Fenetre(){
2 setTitle("Essai_boite_de_dialogue");
setSize(400,200);
4 setLayout(new FlowLayout());
lanceDial=new JButton("lancement_dialogue");
6 add(lanceDial);
lanceDial.addActionListener(this);
8 }
```

## ■ fonction actionPerformed

# Exemple 19 : Gestion d'une boite de dialogue III

```
public void actionPerformed(ActionEvent ev){  
2   Dialogue bd = new Dialogue(this);  
    texte = bd.lanceDialogue();  
4   if(texte!=null) System.out.println("valeur"+texte);  
    else System.out.println("dialogue abandonné");  
6   bd.dispose(); }  
}
```

## ■ classe Dialogue

```
1 class Dialogue extends JDialog implements ActionListener{  
  boolean OK;  
3   JButton OKBouton;  
  JTextField champTexte;
```

## ■ constructeur de Dialogue

# Exemple 19 : Gestion d'une boite de dialogue IV

```
1 public Dialogue(JFrame f){  
2     super(f, "Dialogue_de_saisie", true);  
3     setSize(250,120);  
4     OKBouton=new JButton("OK");  
5     OKBouton.addActionListener(this);  
6     champTexte=new JTextField(20);  
7     setLayout(new FlowLayout());  
8     add(OKBouton);  
9     add(champTexte);  
10    }  
11}
```

## ■ fonction actionPerformed

```
1 public void actionPerformed(ActionEvent ev){  
2     if (ev.getSource()==OKBouton)  
3         OK=true; setVisible(false);  
4 }
```

## ■ fonction lanceDialogue

# Exemple 19 : Gestion d'une boite de dialogue V

```
public String lanceDialogue(){
2   OK= false;
    setVisible(true);
4   if(OK) return champTexte.getText();
    else return null;
6 }
```

## ■ classe TestDialogue

```
public class TestDialogue{
2 public static void main(String args[]){
    Fenetre fen=new Fenetre();
4   fen.setVisible(true);}}
```

# Exemple d'événement Dialogue I



# Exemple de Gestion d'un Menu I

Voici un exemple très simple de gestion d'un Menu :

- On crée une barre de menus comportant
  - menu **couleur**(Rouge et Vert)
  - menu **dimensions** ( longueur+largeur)
- On affiche en mode console les différentes actions de l'utilisateur sur les options en précisant la source et la chaîne de commande correspondante.

# Exemple de gestion de menu I



# Exemple de Gestion d'un Menu I

```
import javax.swing.*;  
2 import java.awt.*;  
 import java.awt.event.*;  
4 class Fenetre extends JFrame implements ActionListener{
```

## ■ Voici les champs dont j'ai besoin

```
JMenuBar barreMenus;  
2 JMenu couleur, dimensions;  
 JMenuItem rouge, vert, longueur, largeur;  
4 public Fenetre(){  
 setTitle( "Exemple_de_Menu" );  
6 setSize(300,150);
```

# Exemple de Gestion d'un Menu II

## ■ Création de barre des menus

```
barreMenus=new JMenuBar();  
2 setJMenuBar(barreMenus);
```

## ■ Creation menu couleurs et ses options rouge et vert

```
couleur=new JMenu( "Couleur" );  
2 barreMenus.add(couleur);  
rouge=new JMenuItem( "Rouge" );  
4 couleur.add(rouge);  
rouge.addActionListener(this);  
6 vert=new JMenuItem( "Vert" );  
couleur.add(vert);  
8 vert.addActionListener(this);
```

# Exemple de Gestion d'un Menu III

- Création du menu dimension et de ses item longueur et largeur

```
dimensions =new JMenu("Dimensions");
2 barreMenus.add(dimensions);
longueur=new JMenuItem("longueur");
4 dimensions.add(longueur);
longueur.addActionListener(this);
6 largeur=new JMenuItem("largeur");
dimensions.add(largeur);
8 largeur.addActionListener(this);
}
```

- Gestion des évènement

# Exemple de Gestion d'un Menu IV

```
1 public void actionPerformed(ActionEvent ev){  
    Object Ob= ev.getSource();  
    3 System.out.println("Action_= "+ev.getActionCommand());  
    if (Ob==rouge) System.out.println("Act_rouge");  
    5 if (Ob==vert) System.out.println("Act_vert");  
    if (Ob==longueur) System.out.println("Act_longueur");  
    7 if (Ob==largeur) System.out.println("Act_largeur");}  
}
```

```
public class TestMenu{  
    2 public static void main(String args[]){  
        Fenetre fen=new Fenetre();  
        4 fen.setVisible(true); }  
    }
```