Programmation d'Interfaces Graphiques en JAVA
Source : cours de l'école de printemps "Initiation au langage Java" présenté par Anne-Marie Leclech-Déplanche

Paul CHECCHIN
Christophe BLANC

Préambule

- Pré-requis
  - connaissance des concepts de l'approche Objet
  - expérience minimum de programmation en Java
- Objectifs
  - découverte des éléments essentiels à la conception d'interface Homme-Machine
  - pas d'examen complet des multiples classes et méthodes concernées
  - acquisition de connaissances de base avant un futur perfectionnement …

Qu'est-ce qu'une interface graphique ? (1)

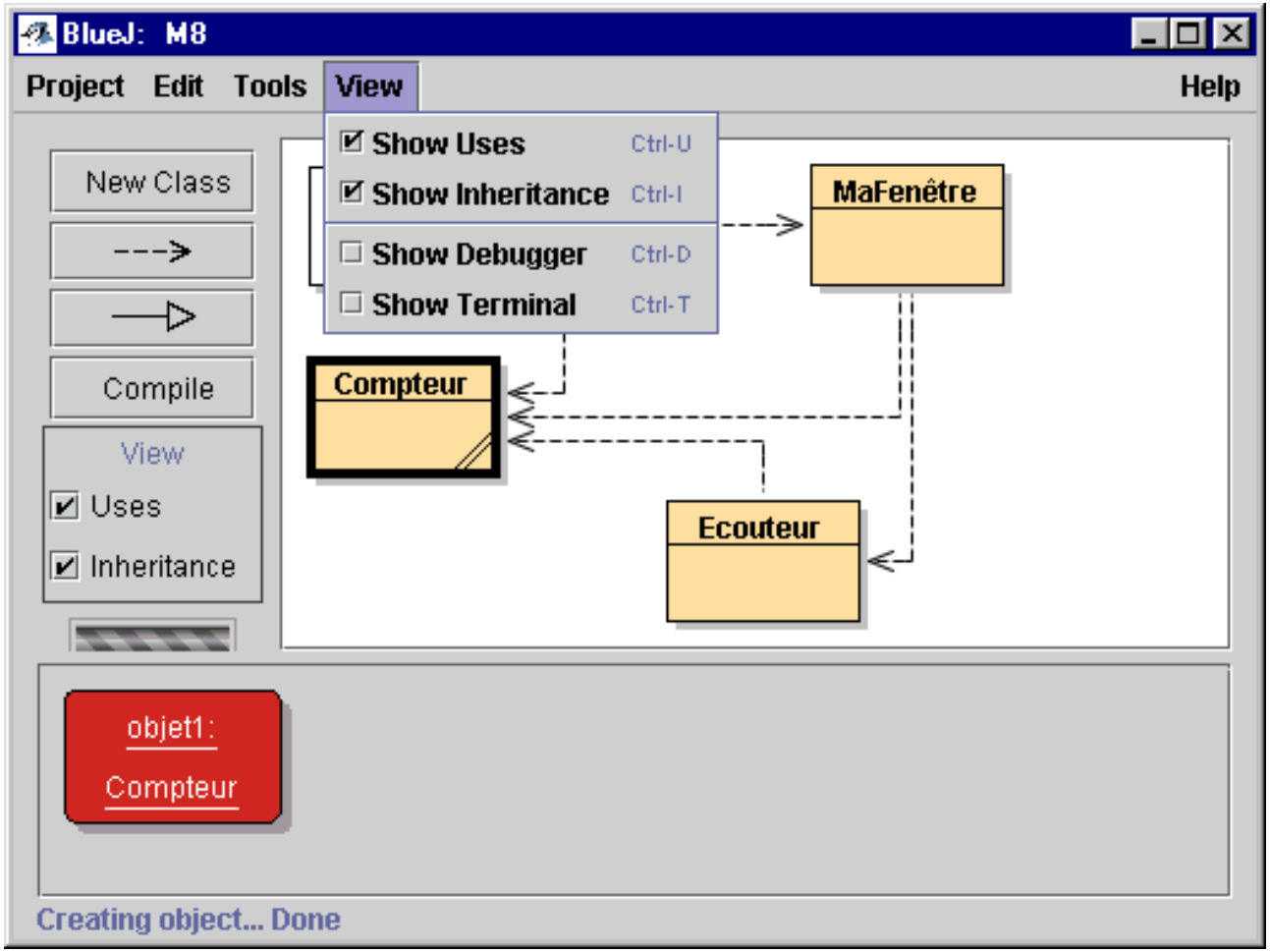
- Communication avec l'utilisateur dans applications informatiques
  ⇒ interface utilisateur = interface homme-machine

- Interface homme-machine
  - mode texte
  - mode graphique ⇒ interface graphique

- Interface graphique
  = GUI, « Graphical User Interface »

Qu'est-ce qu'une interface graphique ? (2)

- Par exemple…



Les API utilisées en Java (1)

- Deux bibliothèques
  - AWT, « Abstract Window Toolkit » : Java 1.0 et Java 1.1
  - Swing : Java 2 (≥ JDK/SDK 1.2)

- Mise à disposition d'un ensemble de classes pour la création d'interfaces graphiques

- Paquetages incontournables
  - java.awt
  - javax.swing
  - java.awt.event

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

Les API utilisées en Java (2)

- AWT, « Abstract Window Toolkit »
  - composants graphiques « lourds »
    - utilisent les composants natifs de l'OS
    - place mémoire importante
    - aspect changeant selon la plateforme
    - possibilités limitées
- Swing
  - composants graphiques « légers »
    - intègrent toutes les informations nécessaires à leur représentation
    - place mémoire moindre
    - aspect indépendant de la plateforme
    - nombreuses possibilités
    - moins efficaces à l'exécution
    - permettent la séparation entre le modèle et la vue
  - il est fortement conseillé d'utiliser les composants Swing…
    sauf si écriture d'applet limitée aux fonctionnalités Java 1.0 ou 1.1

Diagramme de classes (1)

- Les classes de Swing héritent de classes de AWT

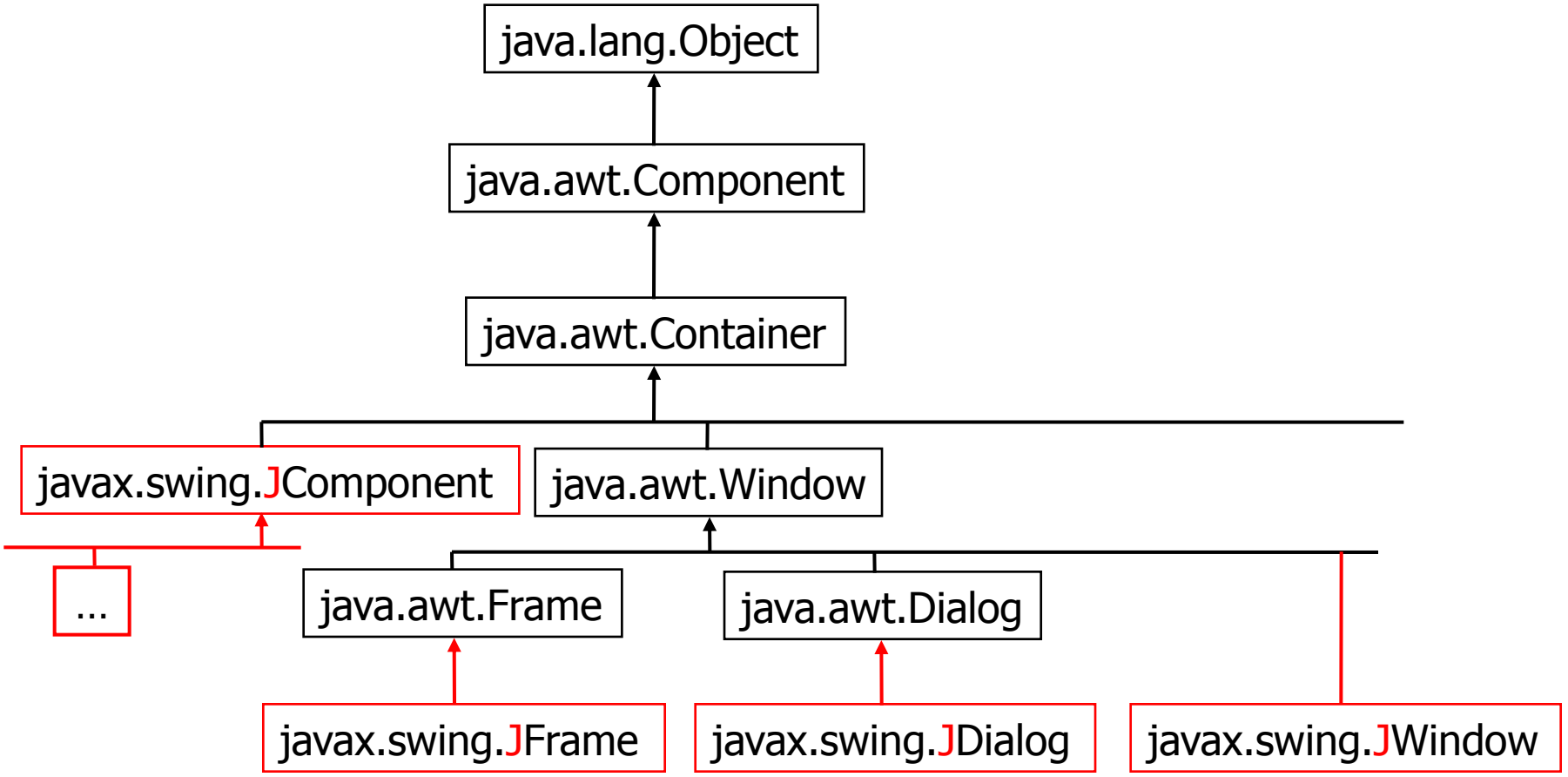
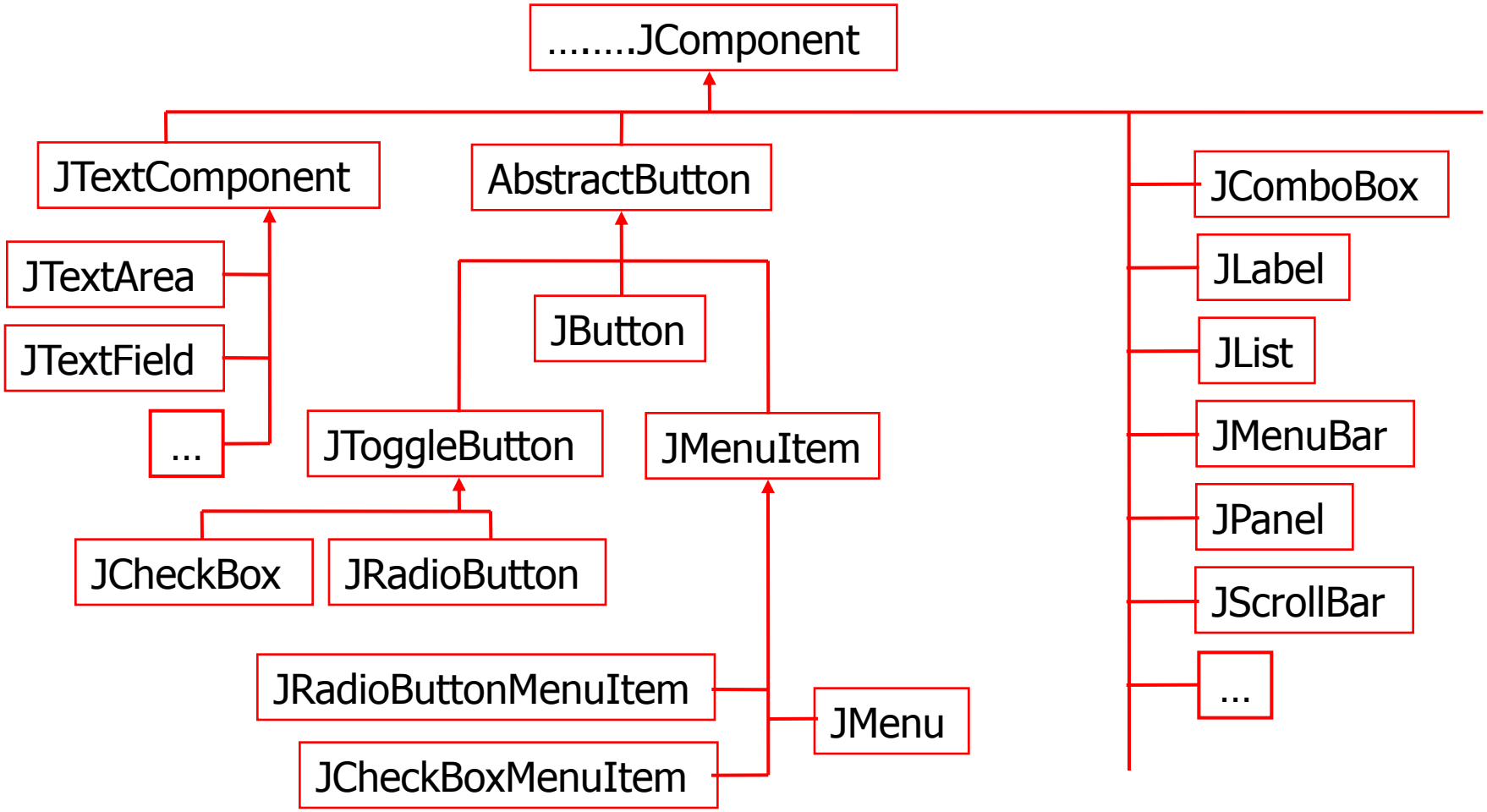


Diagramme de classes (2)



Construire une interface graphique avec Java

⌘ Assembler des composants graphiques

- manipuler principalement trois types d'objets :
  - des composants graphiques
  - des conteneurs
  - des gestionnaires de positionnement

⌘ Gérer des événements

- prendre en compte les actions de l'utilisateur sur les composants graphiques ⇒ événements
- réagir par des traitements adaptés

Mise en place d'une fenêtre ou cadre (1)

- Classe JFrame
  - par instanciation directe

Exemple1.java

```
import javax.swing.*;

public class Exemple1
{
    public static void main(String[] args)
    {
        JFrame fenêtre = new JFrame();

fenêtre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        fenêtre.setSize(400,300);
        fenêtre.setLocation(10,20);
        fenêtre.setTitle("Fenêtre principale");
        fenêtre.setVisible(true);
    }
}
```

Mise en place d'une fenêtre ou cadre (2.a)

- Classe JFrame
  - par héritage

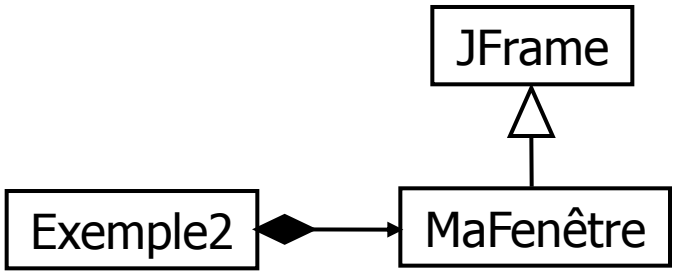
MaFenêtre.java

```
import javax.swing.*;

public class MaFenêtre extends JFrame
{
    public MaFenêtre()
    {
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(10, 20, 400, 300);
setTitle("Fenêtre principale");
    }
}
```

Mise en place d'une fenêtre ou cadre (2.b)

Exemple2.java

```
public class Exemple2
{
    public static void main(String[] args)
    {
MaFenêtre fenêtre = new MaFenêtre();
fenêtre.setVisible(true);
    }
}
```



Exercice 1

1. Écrire un programme qui affiche deux fenêtres…

MaFenêtre1.java

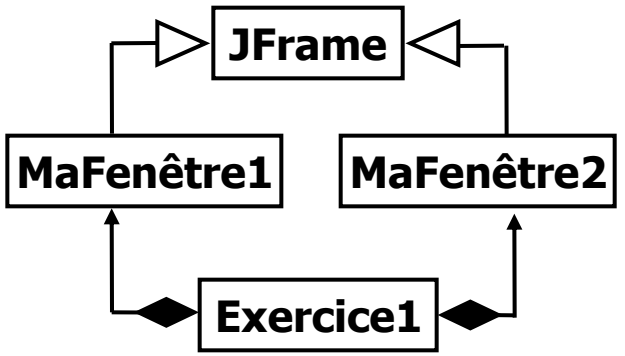
```
import javax.swing.*;

public class MaFenêtre1 extends JFrame
{
   public MaFenêtre1()
   {
setBounds(10,20,400,300);
setTitle("Fenêtre numéro 1");
   }
}
```

MaFenêtre2.java

```
import javax.swing.*;

public class MaFenêtre2 extends JFrame
{
   public MaFenêtre2()
   {
setBounds(350,370,400,300);
setTitle("Fenêtre numéro 2");
   }
}
```

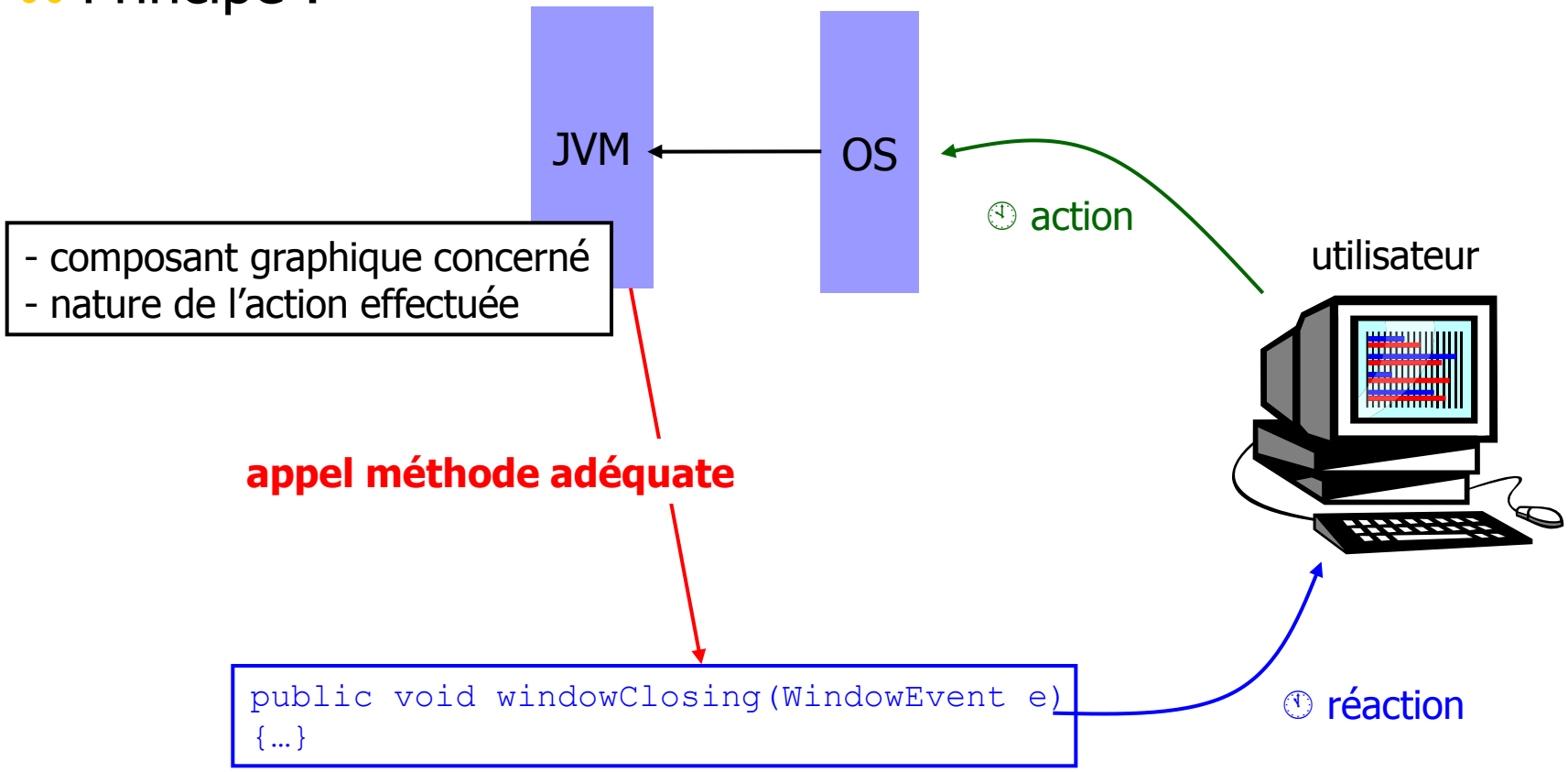
Exercice1.java

```
public class Exercice1
{
   public static void main(String[] args)
   {
MaFenêtre1 fenêtre1 = new MaFenêtre1();
MaFenêtre2 fenêtre2 = new MaFenêtre2();
fenêtre1.setVisible(true);
fenêtre2.setVisible(true);
   }
}
```



Programmation événementielle (1)

- Programmation « non modale »
- Principe :

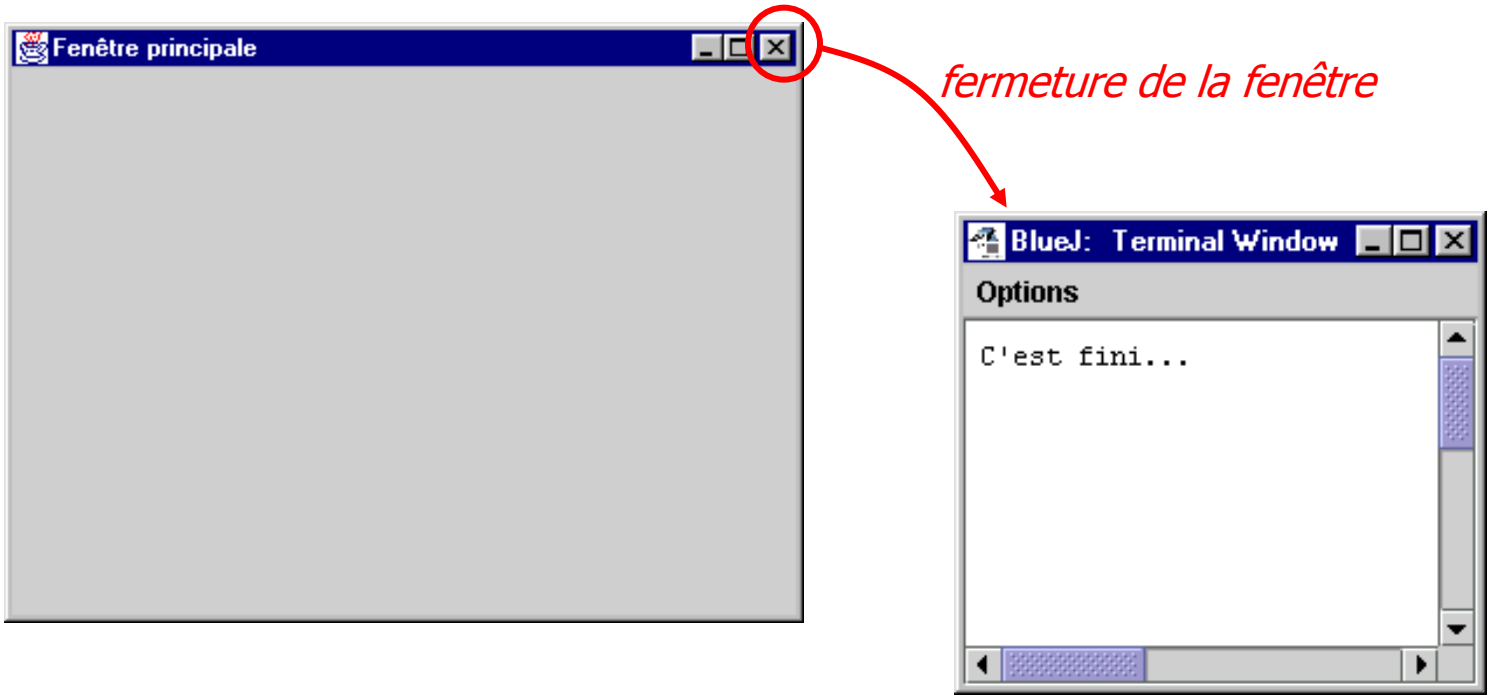


Programmation événementielle (2)

- Mécanisme de « call-back »
  - objet source de l'événement = émetteur ou observé
  - objet intéressé par l'événement = observateur ou écouteur (« listener »)
  - un objet écouteur s'enregistre auprès d'un (ou plusieurs) émetteur(s) pour certaine(s) catégorie(s) d'événements
  - un objet émetteur s'engage à notifier tous ses objets écouteurs enregistrés à chaque fois qu'un événement donné se produit
  - diverses catégories d'événements en programmation graphique
    - événements souris
    - événements clavier
    - événements liés aux composants graphiques
    - etc…

Gestion des événements-fenêtre (1)

- Comportement attendu…



Gestion des événements-fenêtre (2.a)

- Spécification d'un écouteur
  - par implémentation de l'interface `WindowListener`

MonEcouteur.java

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

public class MonEcouteur implements WindowListener
{
    public void windowClosing(WindowEvent e)
    {   System.out.println("C'est fini...");
        System.exit(0);
    }
    public void windowActivated(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowIconified(WindowEvent e) {}
    public void windowOpened(WindowEvent e) {}
}
```

La catégorie d 'événements

L'unique événement traité

Gestion des événements-fenêtre (2.b)

- Spécification d'un écouteur
  - par héritage de la classe `WindowAdapter`

MonEcouteur.java

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

public class MonEcouteur extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {   System.out.println("C'est fini...");
        System.exit(0);
    }
}
```

La catégorie d 'événements

L'unique événement traité

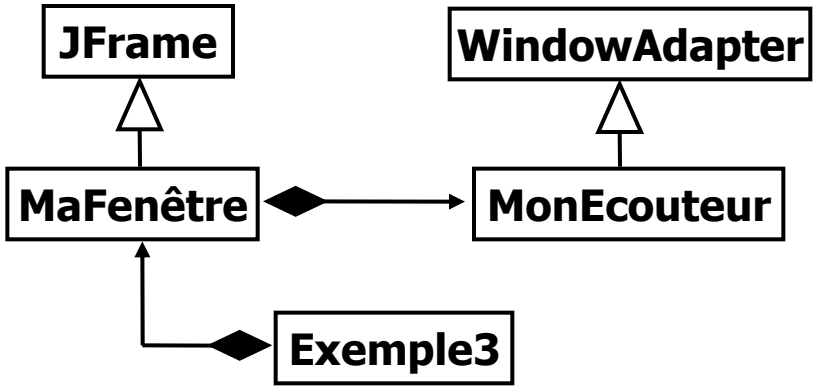
Gestion des événements-fenêtre (3)

- Enregistrement d'un objet écouteur auprès de la fenêtre

MaFenêtre.java

```
import javax.swing.*;

public class MaFenêtre extends JFrame
{
   public MaFenêtre()
   {
setBounds(10,20,400,300);
setTitle("Fenêtre principale");
MonEcouteur écouteur = new MonEcouteur();
addWindowListener(écouteur);
   }
}
```

Instanciation et abonnement d'un écouteur



Gestion des événements-fenêtre (4)

- Fonctionnement
  - clic sur case de fermeture ⇒ l'objet `fenêtre` devient source d'un événement-fenêtre
  - un objet de type `WindowEvent` est créé dans lequel diverses informations sont rangées (source de l'événement, coordonnées graphiques, etc)
  - l'objet `écouteur` étant enregistré comme écouteur des événements-fenêtre de cet émetteur, sa méthode `windowClosing` est activée

- N.B.
  - plusieurs écouteurs peuvent être enregistrés sur un même émetteur ⇒ ordre d'exécution sans déterminisme
  - un même écouteur peut être enregistré sur plusieurs émetteurs

Gestion des événements-fenêtre (5)

- Les méthodes de `WindowListener` et événements-fenêtre associés :
  - `windowClosing()` : la fenêtre est en cours de fermeture
  - `windowClosed()` : la fenêtre a été fermée
  - `windowOpened()` : la fenêtre a été rendue visible
  - `windowActivated()` : la fenêtre devient la fenêtre active
  - `windowDeactivated()` : la fenêtre est sur le point de devenir inactive
  - `windowDeiconified()` : la fenêtre a été agrandie
  - `windowIconified()` : la fenêtre a été réduite

2. Écrire un programme qui réagit dans la fenêtre terminal à ces différents événements-fenêtre…

Ajout d'un composant dans une fenêtre

⌘ Classe JButton

MaFenêtre.java

```
import javax.swing.*;

public class MaFenêtre extends JFrame
{
   private JButton bouton;

   public MaFenêtre()
   {
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(10,20,400,300);
setTitle("Fenêtre principale");
getContentPane().setLayout(null);

this.bouton = new JButton();
this.bouton.setText("BOUTON 1");
this.bouton.setBounds(30,30,100,30);

getContentPane().add(this.bouton);
   }
}
```

Exemple4.java

Désactivation du placement automatique

Instanciation d'un objet bouton

Insertion du bouton dans le conteneur attaché à la fenêtre

Gestion de l'événement-bouton (1)

- Un seul événement émis : celui lié au clic sur le bouton
- Evénement de catégorie `ActionEvent`

- Source = le bouton

- Ecouteur = par implémentation de l'interface `ActionListener` ou par héritage de la classe `ActionAdapter`
- Une seule méthode pour l 'écouteur = `actionPerformed(ActionEvent e)`

- Enregistrement de l 'écouteur par la méthode `addActionListener(écouteur)`

Gestion de l'événement-bouton (2)

Exemple5.java

```
import javax.swing.*;

public class MaFenêtre extends JFrame
{
   private JButton bouton;

   public MaFenêtre()
   {
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(10,20,400,300);
setTitle("Fenêtre principale");
getContentPane().setLayout(null);

this.bouton = new JButton();
this.bouton.setText("BOUTON 1");
this.bouton.setBounds(30,30,100,30);
this.bouton.addActionListener(new MonEcouteur());

getContentPane().add(this.bouton);
   }
}
```

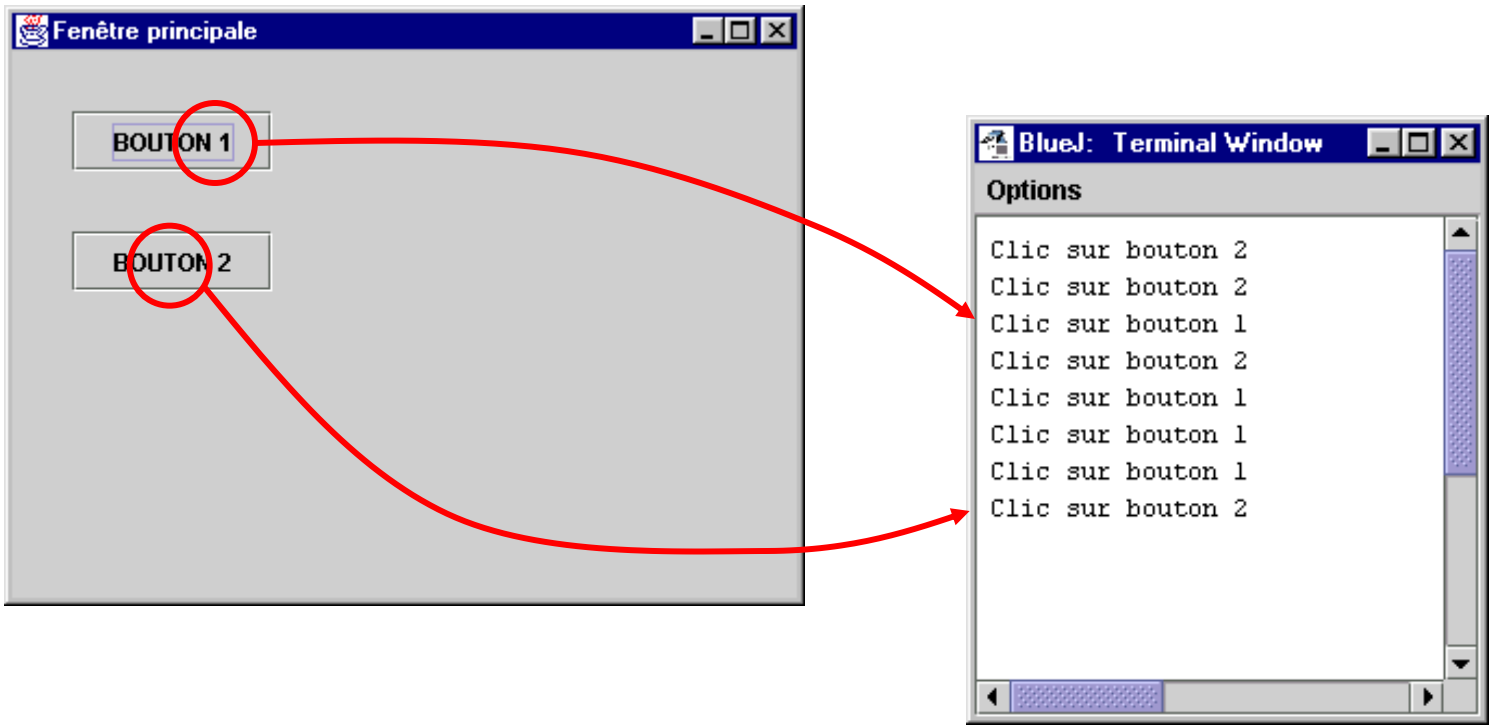
MaFenêtre.java

MonEcouteur.java

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

public class MonEcouteur implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        System.out.println("Clic sur bouton 1");
    }
}
```

Ajout d'un deuxième composant dans une fenêtre (1)

3. Écrire un programme qui réalise l'interface suivante



- Méthode `getSource()` de la classe `EventObject`...

Ajout d'un deuxième composant dans une fenêtre (2)

```
import javax.swing.*;

public class MaFenêtre extends JFrame
{
   private JButton bouton1;
   private JButton bouton2;

   public MaFenêtre()
   {
…
this.bouton1 = new JButton();
this.bouton1.setText("BOUTON 1");
this.bouton1.setBounds(30,30,100,30);
this.bouton2 = new JButton();
this.bouton2.setText("BOUTON 2");
this.bouton2.setBounds(30,90,100,30);

MonEcouteur écouteur =
          new MonEcouteur(this.bouton1, this.bouton2);
this.bouton1.addActionListener(écouteur);
this.bouton2.addActionListener(écouteur);

getContentPane().add(this.bouton1);
getContentPane().add(this.bouton2);
   }
}
```

Exercice3.java

MaFenêtre.java

Ajout d'un deuxième composant dans une fenêtre (3)

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

public class MonEcouteur implements ActionListener
{
    private JButton réf1;
    private JButton réf2;

    public MonEcouteur(JButton b1,JButton b2)
    {
        this.réf1 = b1;
        this.réf2 = b2;
    }

    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == this.réf1)
            System.out.println("Clic sur bouton 1");
        else if (e.getSource() == this.réf2)
            System.out.println("Clic sur bouton 2");
    }
}
```

MonEcouteur.java

Exercice3.java

Agencement des composants : « layouts »

- Gestionnaires de mise en forme, « layout managers » : objets qui s'occupent de la mise en page des composants graphiques dans un conteneur

- Avantage : la mise en page est spécifiée de manière relative et s'adapte aux modifications de taille subies par l'interface

- Lorsqu'un conteneur est créé, un gestionnaire de mise en forme lui est implicitement assigné
  - par défaut
    - pour une JFrame : `BorderLayout`
    - pour un JPanel : `FlowLayout`

- On le modifie par méthode `setLayout(…)` :

```
… FlowLayout flo = new FlowLayout();
  getContentPane().setLayout(flo);
…
```

Agencement des composants : `FlowLayout` (1)

- Classe `FlowLayout`

  - gestionnaire le plus simple
  - composants à la suite, de gauche à droite, puis à la ligne
  - par défaut, les composants sont centrés sur chaque ligne et espacement de 3 pixels au sein d'une ligne et entre lignes

```
FlowLayout flo = new FlowLayout();
```

  - si autre alignement souhaité

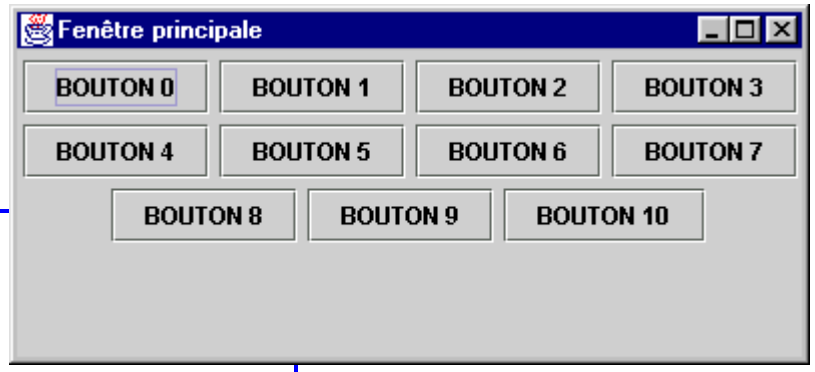
```
FlowLayout flo_R = new FlowLayout(FlowLayout.RIGHT);
```

  - si autre espacement souhaité

```
FlowLayout flo_LE = new
     FlowLayout(FlowLayout.LEFT,30,10);
```

Agencement des composants : `FlowLayout` (2)

- Un exemple...




```
…
  private JButton bouton[];

  public MaFenêtre()
  {
    …
    getContentPane().setLayout(new FlowLayout());

    this.bouton = new JButton[11];
    for (int i = 0; i < 11; i++) {
      this.bouton[i] = new JButton();
      this.bouton[i].setText("BOUTON "+ i);
      getContentPane().add(this.bouton[i]);
    }
  }
…
```

Agencement des composants : `GridLayout` (1)

- Classe `GridLayout`

  - division du conteneur selon une grille (virtuelle)
  - composants à la suite, de gauche à droite sur une ligne, puis ligne suivante + ajustés pour remplir tout l'espace alloué à chaque cellule
  - le nombre de lignes et colonnes doit être spécifié à l'instanciation

```
GridLayout gr = new GridLayout(10,3);
// 10 lignes et 3 colonnes
```

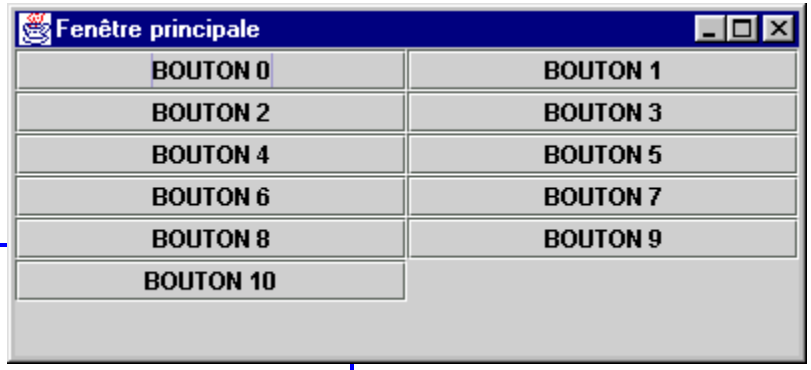
  - par défaut, espacement de 0 pixel horizontalement et verticalement

  - si autre espacement souhaité

```
GridLayout gr_E = new GridLayout(10,3,10,10);
```

Agencement des composants : GridLayout (2)

- Un exemple...




```
…
  private JButton bouton[];

  public MaFenêtre()
  {
    …
    getContentPane().setLayout(new GridLayout(7,2));

    this.bouton = new JButton[11];
    for (int i = 0; i < 11; i++) {
      this.bouton[i] = new JButton();
      this.bouton[i].setText("BOUTON "+ i);
      getContentPane().add(this.bouton[i]);
    }
  }
…
```

Agencement des composants : `BorderLayout` (1)

- Classe `BorderLayout`

  - division du conteneur en 5 sections selon points cardinaux
  - la section doit être spécifiée lors de l'ajout d'un composant au conteneur

```
getContentPane().add(bouton1, BorderLayout.WEST);
```

  - par défaut, espacement de 0 pixel horizontalement et verticalement + les composants des 4 coins cardinaux reçoivent autant d'espace que nécessaire, le centre reçoit ce qui reste

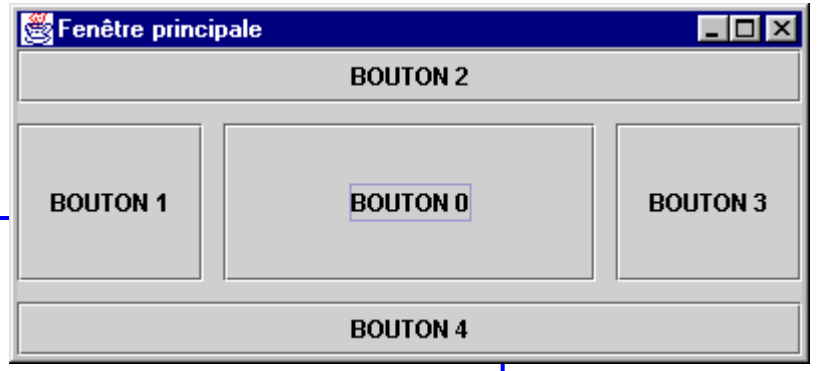
```
BorderLayout bo = new BorderLayout();
```

  - si autre espacement souhaité

```
BorderLayout bo_E = new BorderLayout(10,6);
```

Agencement des composants : BorderLayout (2)

- Un exemple…

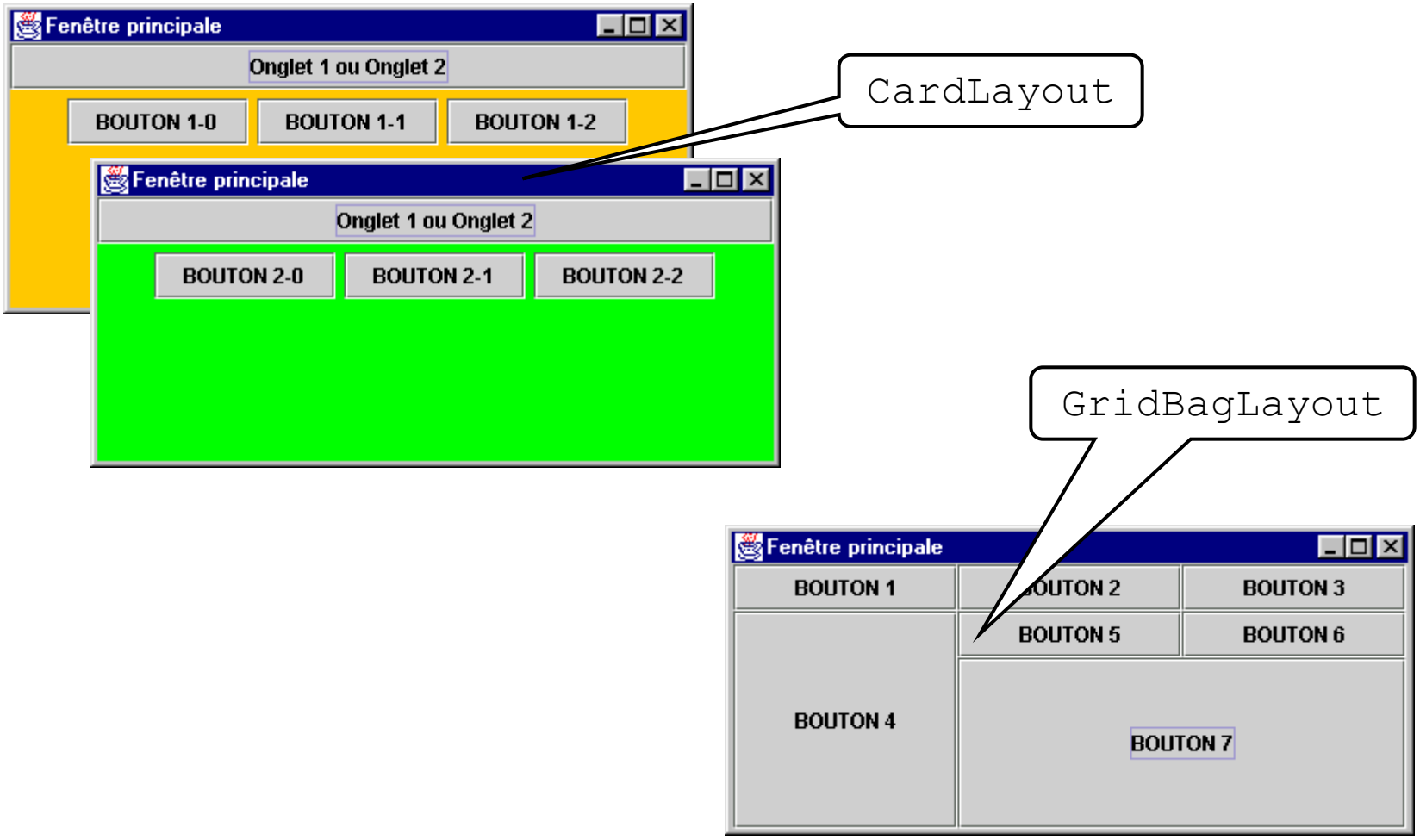



```
…
  private JButton bouton[];

  public MaFenêtre()
  {
     …
     getContentPane().setLayout(new BorderLayout(10,10));

     this.bouton = new JButton[5];
     …
     getContentPane().add(this.bouton[0],BorderLayout.CENTER);
     getContentPane().add(this.bouton[1],BorderLayout.WEST);
     getContentPane().add(this.bouton[2],BorderLayout.NORTH);
     getContentPane().add(this.bouton[3],BorderLayout.EAST);
     getContentPane().add(this.bouton[4],BorderLayout.SOUTH);
  }
…
```

Agencement des composants : « layouts » (5)

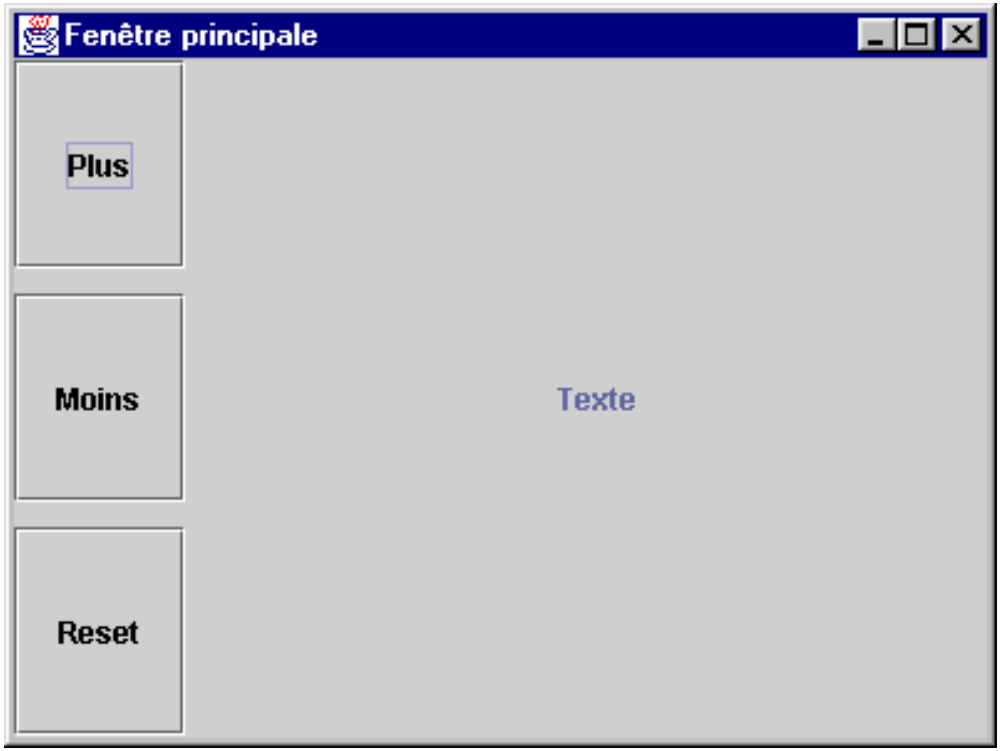
- Et aussi…
- Classe `BoxLayout`
  - proche du GridLayout mais avec une seule dimension
- Classe `CardLayout`
  - groupe de conteneurs ou composants affichés un par un
  - ≈ boîte à onglets
- Classe `GridBagLayout`
  - extension de GridLayout mais…
    - un composant peut occuper plusieurs cellules
    - les proportions entre les lignes et colonnes n'ont pas à être égales
    - les composants dans les cellules peuvent s'agencer de diverses manières
  - à tout composant à ajouter est associé un objet de type `GridBagConstraints` spécifiant sa taille et sa position

Agencement des composants : « layouts » (6)



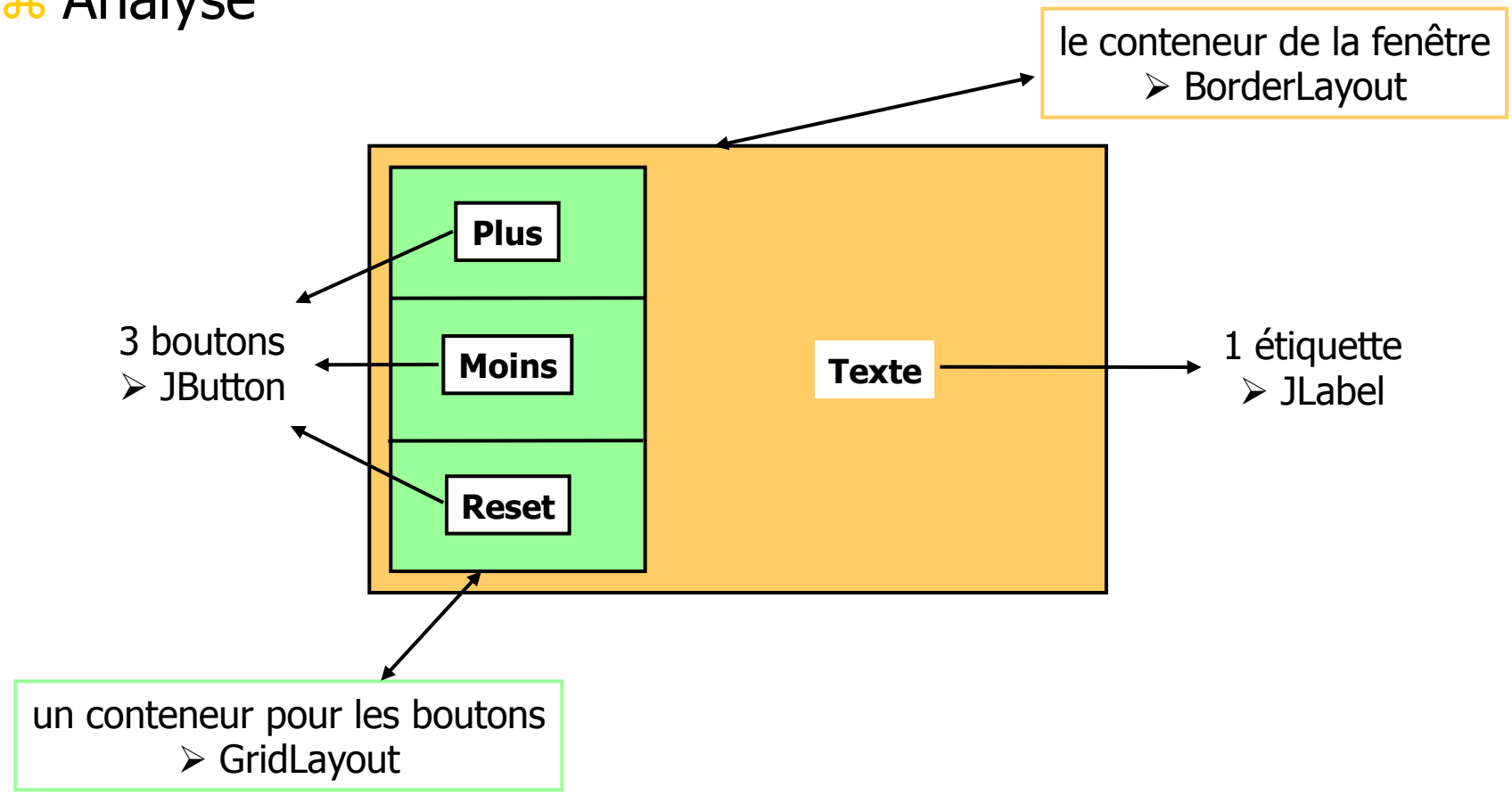
Combiner les gestionnaires de mise en forme (1)

- Résultat attendu…



Combiner les gestionnaires de mise en forme (2)

⌘ Analyse



Combiner les gestionnaires de mise en forme (3)

MaFenêtre.java

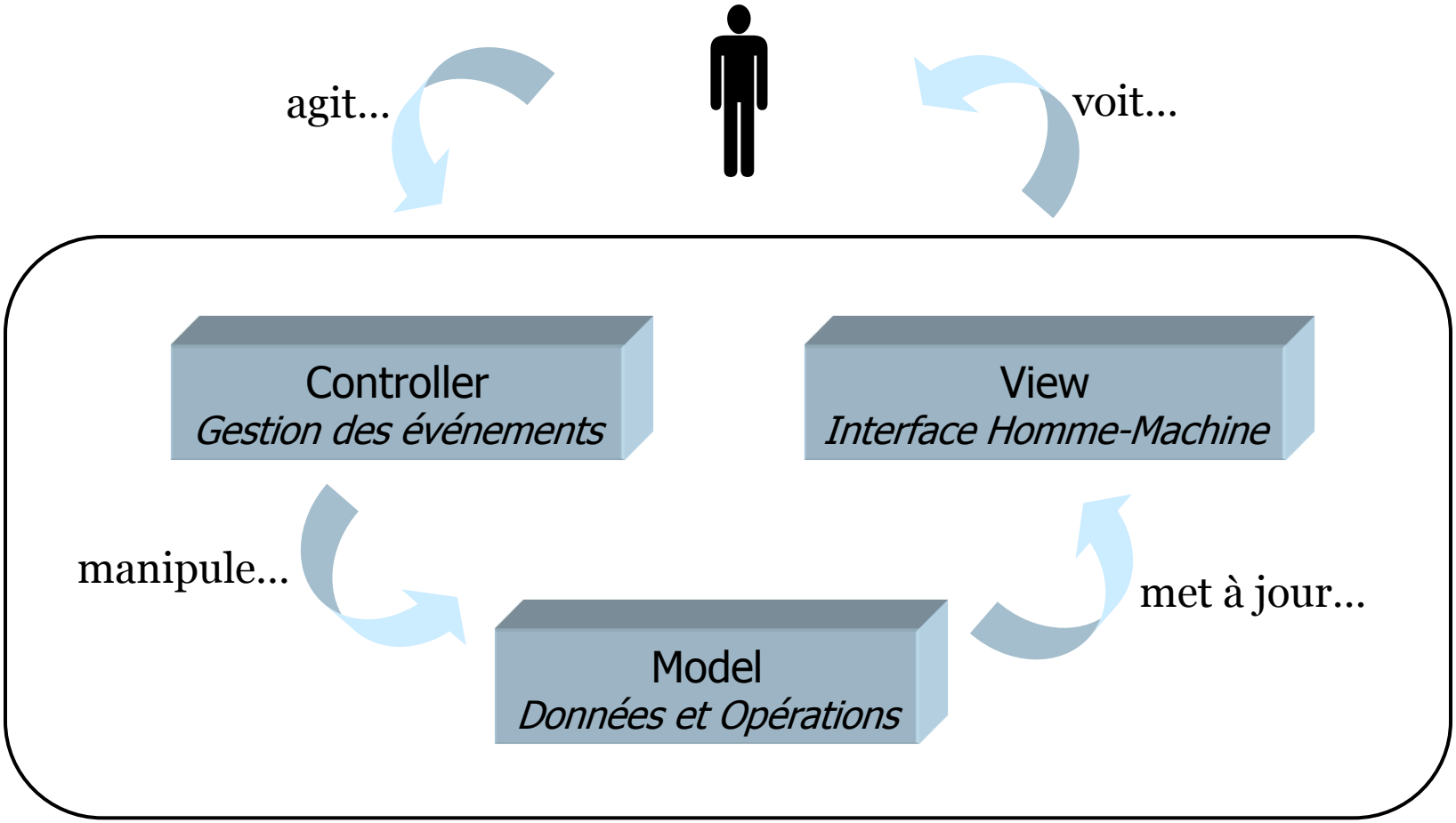
```
import javax.swing.*;
import java.awt.*;

public class MaFenêtre extends JFrame
{
   private JButton bPlus;
   private JButton bMoins;
   private JButton bRaZ;
   private JLabel  étiq;
   private JPanel  pBoutons;

                                public MaFenêtre()
                                {
                                    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                                    setBounds(10,20,400,300);
                                    setTitle("Fenêtre principale");
                                    getContentPane().setLayout(new BorderLayout(10,10));

                                    this.bPlus = new JButton("Plus");
                                    this.bMoins = new JButton("Moins");
                                    this.bRaZ = new JButton("Reset");
                                    this.pBoutons = new JPanel(new GridLayout(3,1,10,10));
                                    this.pBoutons.add(this.bPlus);
                                    this.pBoutons.add(this.bMoins);
                                    this.pBoutons.add(this.bRaZ);
                                    this.getContentPane().add(this.pBoutons,BorderLayout.WEST);

                                    this.étiq = new JLabel("Texte",SwingConstants.CENTER);
                                    getContentPane().add(this.étiq,BorderLayout.CENTER);
                                }
                            }
```

Exemple6.java

Le modèle « MVC » (1)

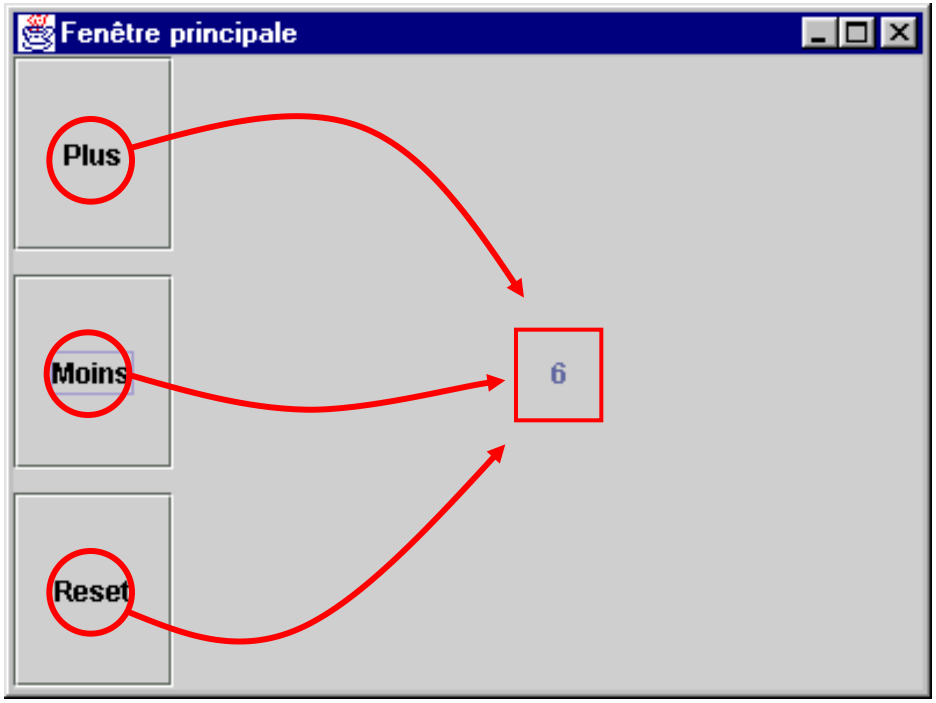
- MVC = « Model-View-Controller »
- modèle créé dans les années 80, pour SmallTalk
- modèle de conception (« design pattern ») qui sépare :
  - le modèle : les données + la logique de manipulation des données
  - la vue : la présentation accessible à l'utilisateur
  - le contrôleur : les mécanismes d'interaction entre l'utilisateur et l'application
- avantage : meilleure séparation entre les composantes
  - réutilisabilité
  - évolutivité
- inconvénient : complexité de la conception
  - recommandé pour les "grosses" applications

Le modèle « MVC » (2)



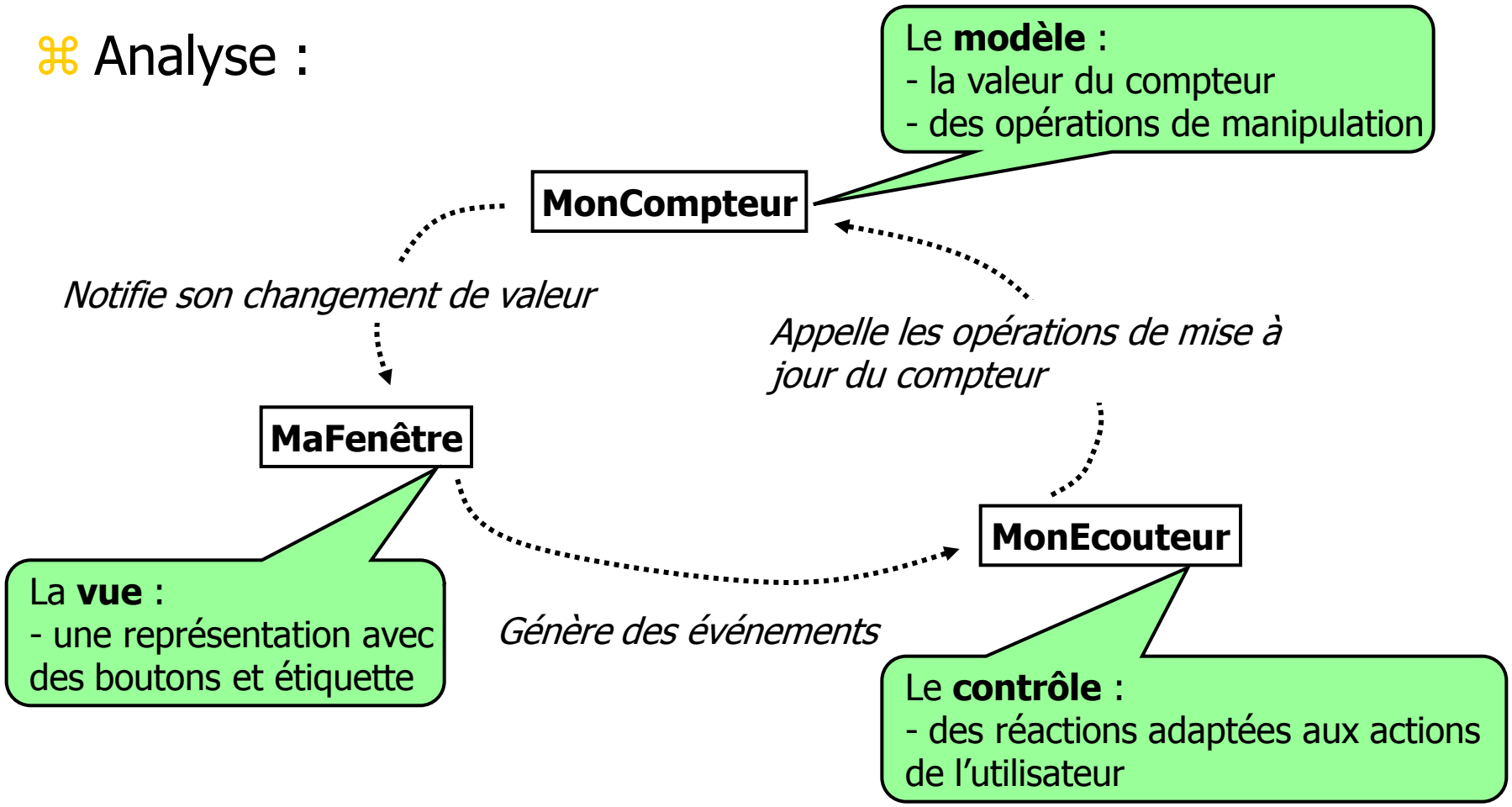
Vers le modèle « MVC » : un exemple… (1)

- Un exemple : un compteur graphique…
- Résultat attendu…



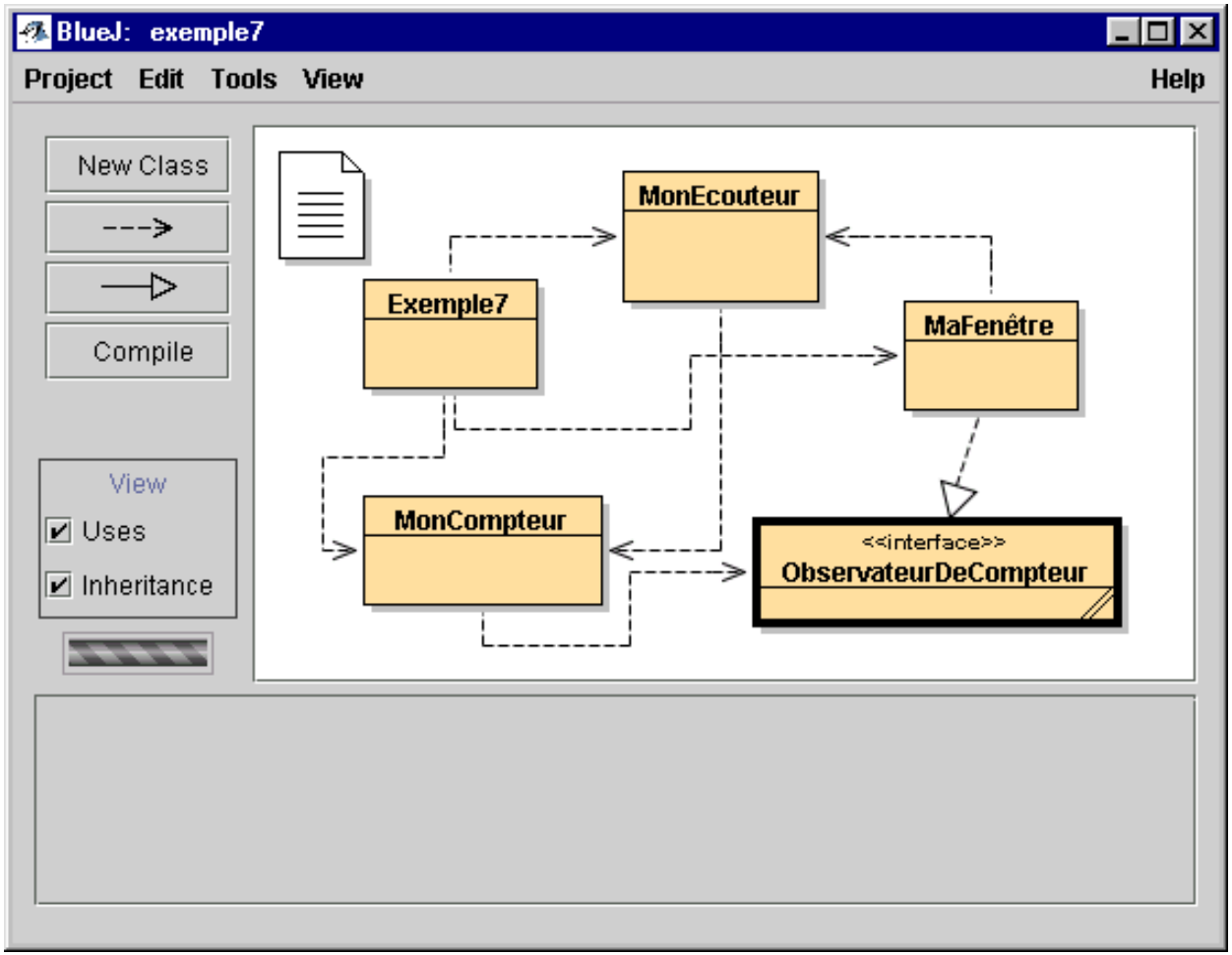
Vers le modèle « MVC » : un exemple… (2)

⌘ Analyse :



Vers le modèle « MVC » : un exemple… (3)

- Mise en œuvre :



Vers le modèle « MVC » : un exemple… (4)

```
public class MonCompteur
{
  private int valMax = 10;
  private int valInit;
  private int valCou;
  private ObservateurDeCompteur obs ;

  public MonCompteur(int init)
  {
     this.valInit = init;
     this.valCou = init;
  }
```

```
  public void inc() {
     this.valCou++;
     if (this.valCou > this.valMax) this.valCou = 0;
     obs.nouvelleValeur(this.valCou);
  }
  public void dec() {
     if (this.valCou > 0) this.valCou--;
     obs.nouvelleValeur(this.valCou);
  }
  public void raz() {
     this.valCou = this.valInit;
     obs.nouvelleValeur(this.valCou);
  }
  public int getValCou() { return this.valCou; }

  public void enregistrerANotifier(ObservateurDeCompteur réf) {
     this.obs = réf;
  }
}
```

MonCompteur.java

Vers le modèle « MVC » : un exemple… (5)

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

public class MonEcouteur implements ActionListener
{
    private MonCompteur leModèle;

    public void actionPerformed(ActionEvent e)
    {   if (e.getActionCommand() == "Plus") leModèle.inc();
        else if (e.getActionCommand() == "Moins") leModèle.dec();
             else if (e.getActionCommand() == "Reset") leModèle.raz();
    }

    public void enregistrerModele(MonCompteur réf)
    {
        leModèle = réf;
    }
}
```

MonEcouteur.java

ObservateurDeCompteur.java

```
public interface ObservateurDeCompteur
{
    public void nouvelleValeur(int val);
}
```

Vers le modèle « MVC » : un exemple… (6)

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

public class MaFenêtre extends JFrame implements ObservateurDeCompteur
{
    …
    public MaFenêtre(int val, MonEcouteur réf)
    {
        …
        bPlus = new JButton("Plus");
        bPlus.setActionCommand("Plus");
        …
        étiq = new JLabel(Integer.toString(val),SwingConstants.CENTER);

        bPlus.addActionListener(réf);
        …
        getContentPane().add(pBoutons,BorderLayout.WEST);
        getContentPane().add(étiq,BorderLayout.CENTER);
    }

    public void nouvelleValeur(int val)
    {
        étiq.setText(Integer.toString(val));
    }
}
```

MaFenêtre.java

Vers le modèle « MVC » : un exemple… (7)

```
public class Exemple7
{
   public static void main(String[] args)
   {
     int valDépart = 2;
     MonCompteur leCompteur = new MonCompteur(valDépart);
     MonEcouteur écouteur = new MonEcouteur();
     MaFenêtre fenêtre = new MaFenêtre(valDépart,écouteur);

     écouteur.enregistrerModele(leCompteur);
     leCompteur.enregistrerANotifier(fenêtre);

     fenêtre.setVisible(true);
   }
}
```

Exemple7.java

Et bien d'autres composants graphiques… (1)

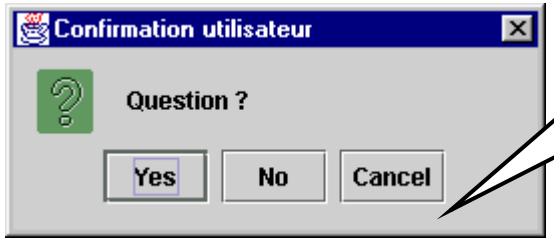
- `JLabel` : intitulé
  - non modifiable par l'utilisateur
- `JTextField` : champ de texte
  - modifiable par l'utilisateur
  - une seule ligne
- `JTextArea` : champ de texte
  - plusieurs lignes
- `JScrollPane` : volet défilant
  - conteneur pouvant contenir n'importe quel composant à faire défiler
- `JScrollBar` : ascenseur
- `JCheckBox` : case à cocher
- `JCheckBoxMenu` : menu de cases à cocher
- `JRadioButton` : bouton radio
- `ButtonGroup` : groupe de boutons radio

Et bien d'autres composants graphiques… (2)

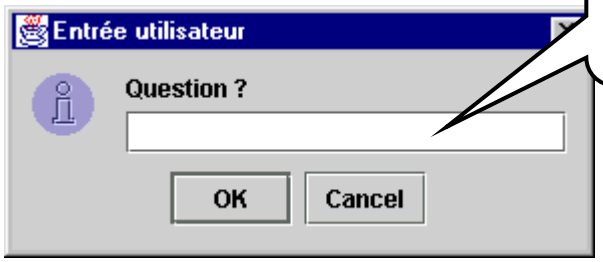
- `JOptionPane` : boîte de dialogue standard
  - boîte de dialogue de confirmation (Yes/No/Cancel)
  - boîte de dialogue d'entrée de texte
  - boîte de dialogue de message
  - boîte de dialigue d'options
- `JSlider` : curseur
- `JProgressBar` : barre de progression
- `JToolBar` : barre d'outils
- `JToolTip` : bulle d'aide
- `JMenu` : menu
- `JMenuItem` : item de menu
- `JMenuBar` : barre de menus d'une fenêtre
- `JList` : liste de choix
- `JComboBox` : bouton de choix accédant à une liste
  - simple ou avec champ de saisie possible

- … Etc…

Et bien d'autres composants graphiques… (3)

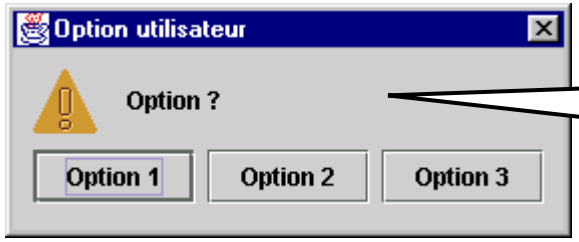


Boîte de dialogue
de confirmation
`…showConfirmDialog(…)`



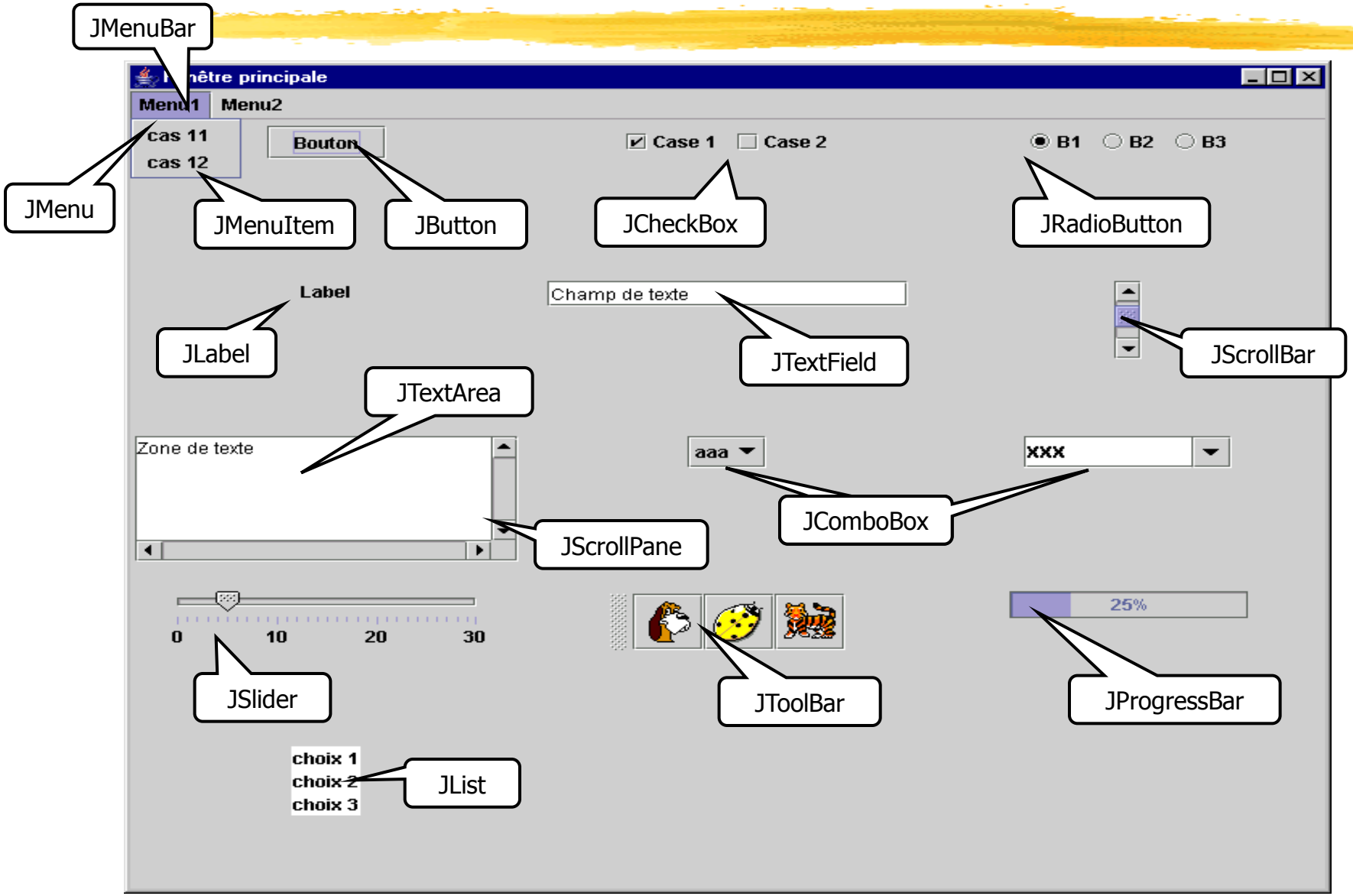
Boîte de dialogue
d'entrée de texte
`…showInputDialog(…)`

Boîte de dialogue
de message
`…showMessageDialog(…)`



Boîte de dialogue
d'options
`…showOptionDialog(…)`

Et bien d'autres composants graphiques… (4)



Et bien d'autres événements et écouteurs… (1)

- Désignation standard
  - écouteur : `XXXListener`
  - adaptateur : `XXXAdapter`
  - type d'événement : `XXXEvent`

- `…Action…`
  - événements liés à l'action de l'utilisateur sur un composant
  - exemples : clic bouton, touche entrée, sélection
  - composants concernés : JButton, JCheckBox, JRadioButton, JTextField, JMenu, JMenuItem, etc.

- `…Adjustment…`
  - événements liés à l'ajustement d'un composant
  - exemple : déplacement curseur d'une barre de défilement
  - composants concernés : JScrollBar et tout composant implémentant l'interface Adjustable

Et bien d'autres événements et écouteurs… (2)

- …Focus…
  - événements liés à l'obtention ou la perte du focus
  - exemple : un champ de texte reçoit le focus
  - composants concernés : tous

- …Item…
  - événements d'élément
  - exemples : sélection dans une liste ou un groupe de cases à cocher
  - composants concernés : JButton, JCheckBox, JRadioButton, JMenu, JMenuItem, Jlist, JComboBox, etc. et tout composant implémentant l'interface ItemSelectable

- …Key…
  - événements clavier
  - exemple : l'utilisateur saisit du texte au clavier
  - composants concernés : tous

Et bien d'autres événements et écouteurs… (3)

- …`Mouse`…
  - événements de souris
  - exemple : entrée du pointeur de souris dans la zone d'un composant
  - composants concernés : tous

- …`MouseMotion`…
  - événements de mouvement de souris de type `MouseEvent`
  - exemple : translation de la souris
  - composants concernés : tous

- …`Window`…
  - événements de fenêtre
  - exemple : réduction d'une fenêtre
  - composants concernés : JWindow, JFrame, JDialog, etc.

- … Etc…

Et bien d'autres événements et écouteurs… (4)

4. Écrire un programme qui affiche les événements de type :

- `FocusEvent`,
- `KeyEvent`,
- `MouseEvent`
- `MouseMotionEvent`

survenant sur le bouton de l'exemple 4

`Exercice4.java`

Dessiner (1)

- Classe `Graphics` de `java.awt`

- Surface de dessin obligatoire
  = composant de type `JPanel`

- Crayon pour dessiner dans un panneau
  = contexte graphique
  = objet de type `java.awt.Graphics`
  - retourné par la méthode `public Graphics getGraphics();` (méthode redéfinie dans la classe `JComponent`)
  - offre des méthodes pour tracer des segments, des figures, du texte, des images, etc…
  - a une couleur associée : méthode `setColor(Color couleur)`
  - a une police associée : méthode `setFont(Font police)`

Dessiner (2)

- Méthode `paintComponent(Graphics g)`
  - disponible pour tout objet `JPanel`
  - définie dans la classe `JComponent`

- Méthode `paintComponent()` à rédéfinir dans le composant si l'on veut compléter par des tracés personnalisés

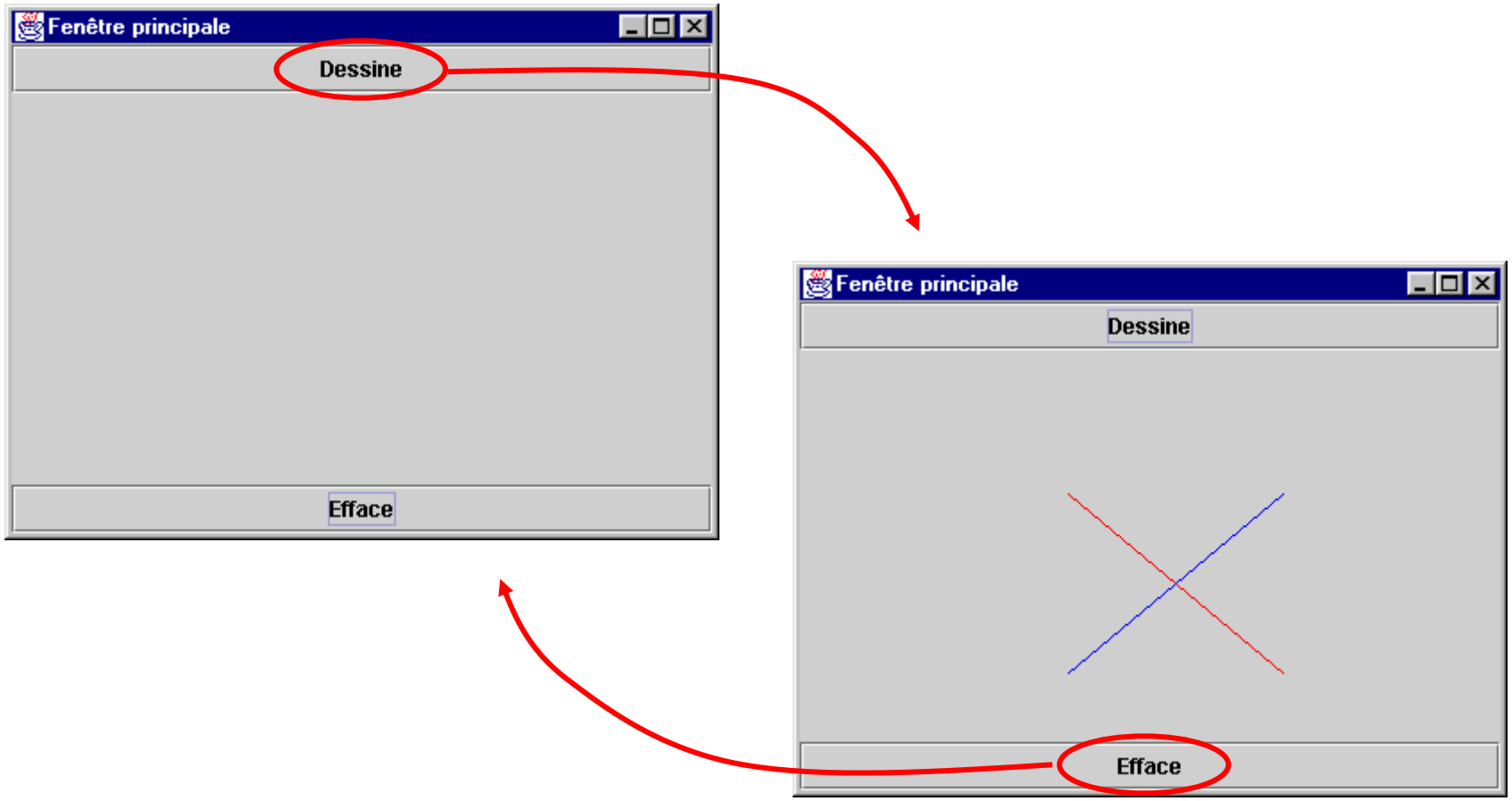
```
class MonPanneau extends JPanel {
…
        public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawline(10,10,120,230);
        }
…
}
```

Dessiner (3)

- La méthode `paintComponent` ne doit pas être directement appelée
  - appel automatique après que le composant ait été momentanément rendu invisible
  - appel explicite par l'intermédiaire de l'appel à la méthode `repaint()`

Dessiner (4)

- Un exemple…



Dessiner (5)

MaFenêtre.java

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

public class MaFenêtre extends JFrame
{
  private JButton bDessine;
  private JButton bEfface;
  private MonPanneau panneau;

  public MaFenêtre() {
     …
     this.bDessine = new JButton("Dessine");
     this.bDessine.setActionCommand("Dessine");
     this.bEfface = new JButton("Efface");
     this.bEfface.setActionCommand("Efface");
     this.panneau = new MonPanneau();

     MonEcouteur écouteur = new MonEcouteur(panneau);
     this.bDessine.addActionListener(écouteur);
     this.bEfface.addActionListener(écouteur);

     getContentPane().add(this.bDessine,BorderLayout.NORTH);
     getContentPane().add(this.bEfface,BorderLayout.SOUTH);
     getContentPane().add(this.panneau,BorderLayout.CENTER);

  }
}
```

Exemple8.java

Dessiner (6)

MonPanneau.java

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

public class MonPanneau extends JPanel
{
    public boolean dessin = false;

    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        if (dessin) {
          g.setColor(Color.red);
          g.drawLine(150,70,270,170);
          g.setColor(Color.blue);
          g.drawLine(150,170,270,70);
        }
}
```

MaFenêtre.java

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

public class MonEcouteur implements ActionListener
{
  private MonPanneau lePanneau;

  public MonEcouteur(MonPanneau réf){
     this.lePanneau = réf;
  }

  public void actionPerformed(ActionEvent e) {
     if (e.getActionCommand() == "Dessine")
        this.lePanneau.dessin = true;
     else if (e.getActionCommand() == "Efface")
        this.lePanneau.dessin = false;
     this.lePanneau.repaint();
    }
}
```

Et bien d'autres méthodes de dessin...

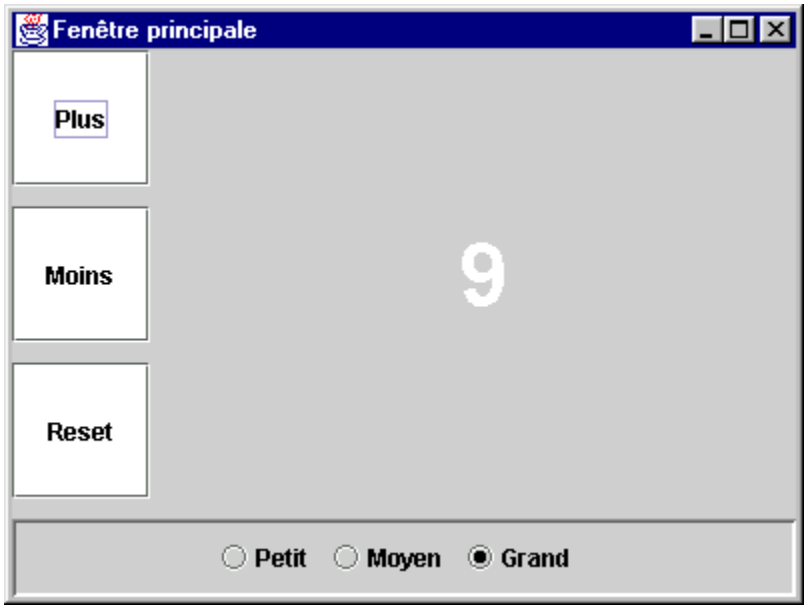
- `drawRect()` : rectangle vide
- `fillRect()` : rectangle plein
- `drawRoundRect()` : rectangle à bords arrondis
- `drawPolygon()` : polygone vide
- `fillPolygon()` : polygone plein
- `drawOval()` : ovale vide
- `fillOval()` : ovale plein
- `drawArc()` : ovale partiel vide
- `fillArc()` : ovale partiel plein
- `drawString()` : dessin de caractères
- `copyArea()` : copie une région rectangulaire sur une autre
- `clearArea()` : supprime une région rectangulaire

- ... Etc...

Un compteur graphique plus élaboré… (1)

- Avec des boutons radio…

5. Analyser le programme de l'exemple 9 (version 1), puis le compléter pour obtenir le résultat suivant

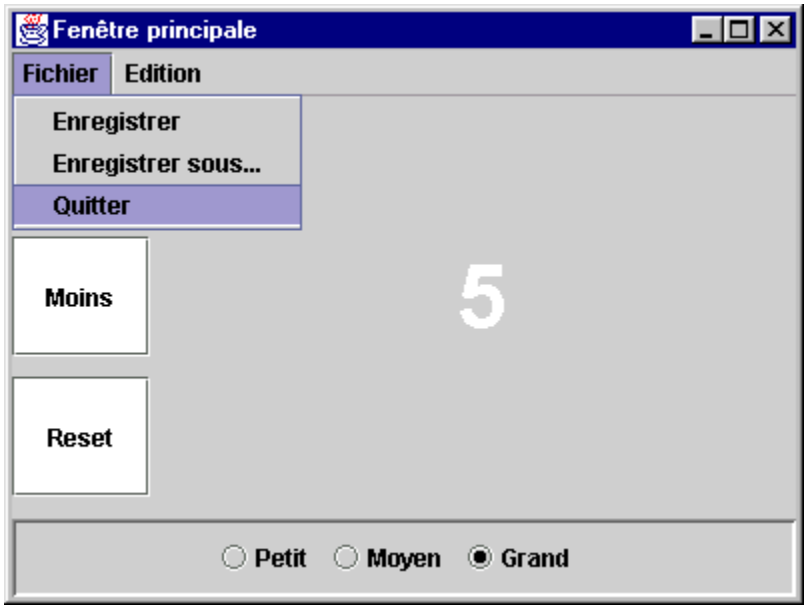


`JRadioButton`
`ButtonGroup`

Un compteur graphique plus élaboré… (2)

- Avec des menus…

6. Analyser le programme de l'exemple 9 (version 2), puis le compléter pour obtenir le résultat suivant

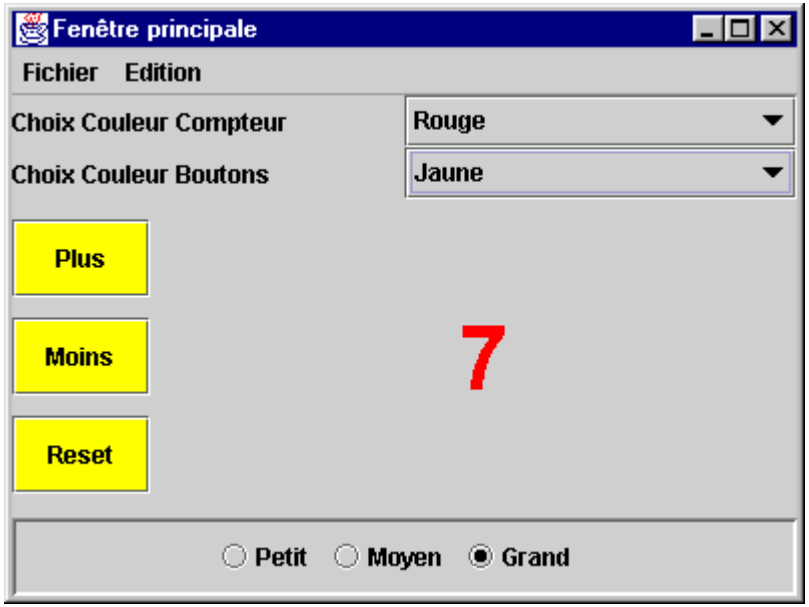


- `JMenuBar`
  `JMenu`
  `JMenuItem`

Un compteur graphique plus élaboré… (3)

- Avec des boutons de choix…

7. Analyser le programme de l'exemple 9 (version 3), puis le compléter pour obtenir le résultat suivant



JComboBox