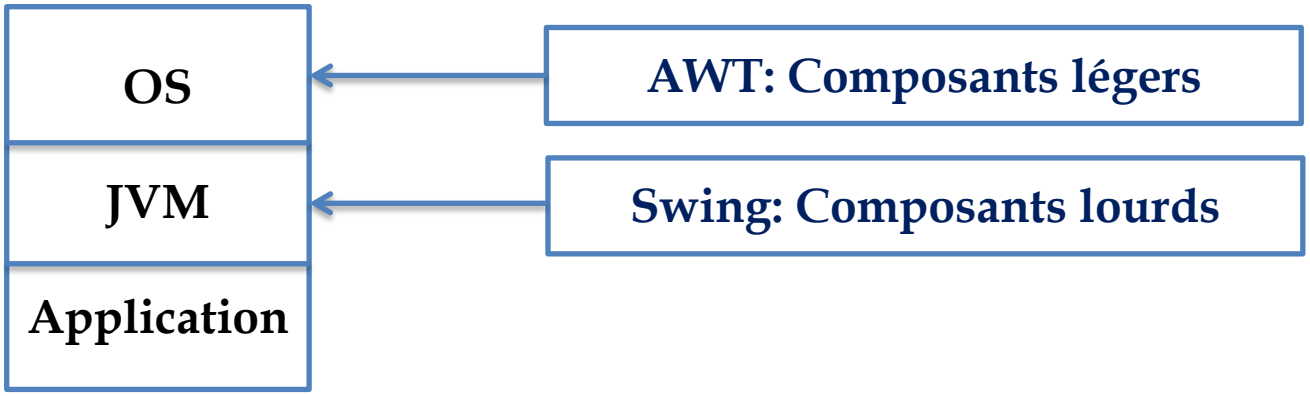
Programmation orientée objet

Chapitre10: Les interfaces graphiques en JAVA

Interfaces graphiques

En java, il existe deux types de composants graphiques:

- Composants **AWT** (Abstract Window ToolKit): Composants qui font appel aux composants graphiques de l'OS
- Composants **SWING** : Composants écrit complètement avec java et sont indépendant de l'OS



Les composants AWT

Nous avons à notre disposition principalement trois types d'objets :

- Les **Components** qui sont des composants graphiques.

Exemple (Button, Label, TextField, …)

- Les **Containers** qui contiennent les Components.

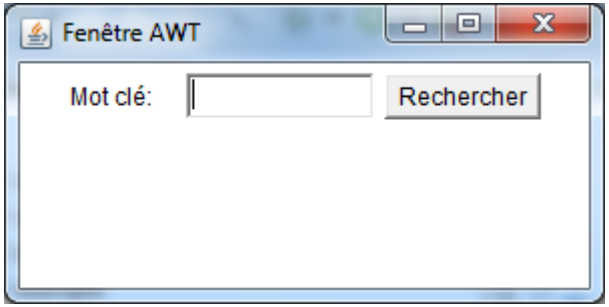
Exemple (Frame, Pannel, ….)

- Les **Layouts** qui sont en fait des stratégies de placement de Components pour les Containers.

Exemple (FlowLayout, BorderLayout, GridLayout, …)

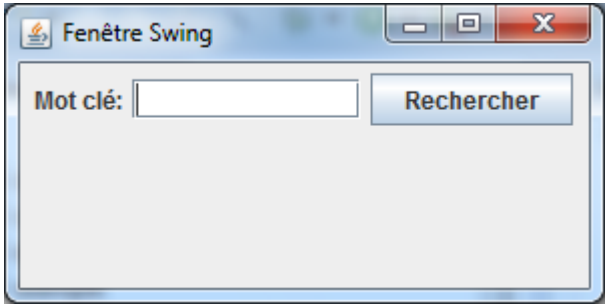
Exemple AWT

```
import java.awt.*;
public class AppAWT {
public static void main(String[] args) {
  Frame f=new Frame("Fenêtre AWT"); //créer une fenêtre
  Label mc=new Label("Mot clé:"); //créer un label
  TextField t=new TextField(10); //créer une zone de texte
  Button b=new Button("Rechercher"); //créer un bouton
  f.setLayout(new FlowLayout()); //définir une gestionnaire de placement
  //ajout des composants à la frame
  f.add(mc);
  f.add(t);
  f.add(b);
  //définir la position et les dimessions de la fenêtre
  f.setBounds(50,50,300,150);
  f.setVisible(true); //afficher la frame}
}
```



Exemple SWING

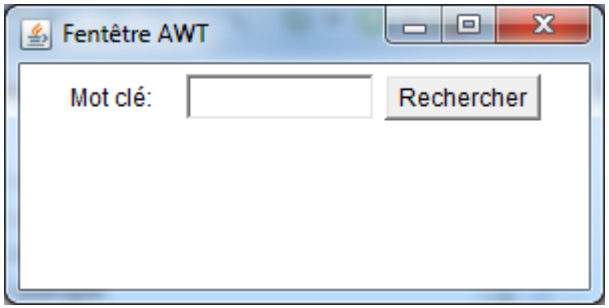
```
import java.awt.FlowLayout;
import javax.swing.*;
public class AppSwing {
public static void main(String[] args) {
  JFrame f=new JFrame("Fenêtre Swing");
  JLabel mc=new JLabel("Mot clé:");
  JTextField t=new JTextField(10);
  JButton b=new JButton("Rechercher");
  f.setLayout(new FlowLayout());
  f.add(mc); f.add(t); f.add(b);
  f.setBounds(50,50,300,150);
  f.setVisible(true);
  }
}
```



Autre manière de définir une fenêtre

Il est plus pratique d'hériter de de la classe Frame:

```
public class FenetreAWT extends Frame{
TextField t=new TextField(10); //créer une zone de texte
Label mc=new Label("Mot clé:"); //créer un label
public FenetreAWT() {
this.setLayout(new    FlowLayout());    //définir    une
gestionnaire de placement
this.setTitle("Fentêtre AWT");
Label mc=new Label("Mot clé:"); //créer un label
//ajout des composants à la frame
this.add(mc);
this.add(t);
this.add(b);
//définir la position et les dimessions de la fenêtre
this.setBounds(50,50,300,150);
this.setVisible(true); //afficher la frame
}
public static void main(String[] args) {
new FenetreAWT();
}
}
```



Les composants Swing

Les principaux composants swing sont :

- Les fenêtres (**JFrame**),
- Les panneau (**JPanel**), et les panneau avec variateur (**JScrollPane**),
- Les boutons (**JButton**),
- Les cases à cocher (**JCheckBox**),
- Les boutons radio (**JButtonRadio**),
- Les listes déroulantes (**JComboBox**),
- Les listes (**JList**),
- Les tables (**JTable**),
- Les variateurs (**JScrollBar**),
- Les étiquettes (**JLabel**),
- Les boîtes de saisie mono ligne (**JTextField**) ou multi lignes (**JTextArea**),
- Les menus (**JMenuBar**), (**JMenu**) les éléments de menu (**JMenuItem**),
- Les boites de dialogues (**JOptionPane**).

Gestion des événements

Dans java, pour qu'un objet puisse répondre à un événement, il faut lui attacher un **écouteur** (**Listener**).

Il existe différents types de Listeners:

- ✓ WindowsListener : pour gérer les événement sur la fenêtre
- ✓ ActionListener :pour gérer les événements produits sur les composants graphiques.
- ✓ KeyListener : pour gérer les événements du clavier
- ✓ MouseListener : pour gérer les événements de la souris.
- ✓ ….

Gestionnaire ActionListener

ActionListener est une interface qui définit une seule méthode:

```
public void actionPerformed(ActionEvent e);
```

L'événement **actionPerformed** est produit quand on valide une action par un clique ou par la touche de validation du clavier.

Pour gérer cet événement dans une application, il faut créer un classe implémentant l'interface ActionListener et redéfinir la réponse aux événements utilisateur, produits dans l'interface, dans la méthode actionPerformed.

La classe implémentant cette interface s'appelle un listener (écouteur) ou un gestionnaire d'événements.

Quand on clique, par exemple, sur un bouton qui est branché sur cet écouteur, la méthode actionPerformed de l'écouteur s'exécute.

Exemple: en implémentant ActionListener

```
public class TestEvent extends JFrame implements ActionListener{
private JLabel jLabel=new JLabel("Saisir nom:");
private JTextField jTextField=new JTextField(10);
private JButton jButton=new JButton("Ajouter");
private JButton jButton2=new JButton("Quitter");
private DefaultListModel<String> model=new DefaultListModel<>();
private JList<String> jList=new JList<>(model);
public TestEvent() {
this.setLayout(new FlowLayout());
this.add(jLabel);this.add(jTextField);
this.add(jButton);this.add(jButton2);
JScrollPane jScrollPane=new JScrollPane(jList);
this.add(jScrollPane);
jButton.addActionListener(this);
jButton2.addActionListener(this);
this.setSize(400,250);
this.setVisible(true);}
public static void main(String[] args) {
new TestEvent();}
public void actionPerformed(ActionEvent e) {
if(e.getSource()==jButton){model.addElement(jTextField.getText());}
if(e.getSource()==jButton2){System.exit(0);}
}}
```

Exemple: en créant un objet de ActionListener

```
public class TestEvent2 extends JFrame {
  private JLabel jLabel=new JLabel("Saisir nom:");
  private JTextField jTextField=new JTextField(10);
  private JButton jButton=new JButton("Ajouter");
  private JButton jButton2=new JButton("Quitter");
  private DefaultListModel<String> model=new DefaultListModel<>();
  private JList<String> jList=new JList<>(model);
  public TestEvent2() {
       this.setLayout(new FlowLayout());
       this.add(jLabel);this.add(jTextField);this.add(jButton);this.add(jButton2);
       JScrollPane jScrollPane=new JScrollPane(jList);this.add(jScrollPane);
       jButton.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
       model.addElement(jTextField.getText());
       }});
       jButton2.addActionListener(new ActionListener() {
       public void actionPerformed(ActionEvent e) {
       System.exit(0);
       }});
       this.setSize(400,250); this.setVisible(true);}
  public static void main(String[] args) {
       new TestEvent2();}
}
```

Gestionnaire WindowListener

Pour fermer une JFrame on peut ajouter après sa création l'instruction suivante :
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
Mais on peut faire mieux en utilisant l'interface **WindowListener**:
L'implémentation de WindowListener nécessite la redéfinition de de sept méthodes:

Exemple:

```
public class Fenetre extends JFrame{
public Fenetre() {
super("Fenêtre");
setSize(400,400);
setVisible(true);
WindowListener wl=new WindowListener() {
public void windowOpened(WindowEvent e) {}
public void windowIconified(WindowEvent e) {}
public void windowDeiconified(WindowEvent e) {}
public void windowDeactivated(WindowEvent e) {}
public void windowClosing(WindowEvent e) {
System.exit(0);
}
public void windowClosed(WindowEvent e) {}
public void windowActivated(WindowEvent e) {}
};
this.addWindowListener(wl);
}
public static void main(String[] args) {
new Fenetre();
}
}
```

Gestionnaire WindowListener

On utilise la classe **WindowAdapter** qui implémente l'interface WindowListener afin de redéfinir les méthodes qui nous intéressent:

Exemple:

```
public class Fenetre extends JFrame{
  public Fenetre() {
       super("Fenêtre");
       setSize(400,400);
       setVisible(true);
       WindowAdapter wl=new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
       };
       this.addWindowListener(wl);
  }
  public static void main(String[] args) {
       new Fenetre();
  }
}
```

Les adaptateurs

Les adaptateurs sont des implémentations toutes prêtes des interfaces d'auditeurs événements, entièrement faites de méthodes vides. Chaque interface **XxxListener** possède une classe **XxxAdapter** correspondante. Pour écrire le traitement des événements qui nous intéressent, il suffit alors de définir une sous classe de l'adaptateur concerné, dans laquelle seules les méthodes pertinentes sont définies.

MouseListener

Exemple:

```
public class FenetreSouris extends JFrame{
  public FenetreSouris() {
       super("Un cadre sensible");
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       JPanel panneau = new JPanel();
       DetecteurSouris ds = new DetecteurSouris();
       panneau.addMouseListener(ds);
       getContentPane().add(panneau);
       setSize(250, 150);
       setVisible(true);}
  public static void main(String[] args) {
       new FenetreSouris();}
}
public class DetecteurSouris extends MouseAdapter {
  public void mousePressed(MouseEvent e) {
       System.out.println("Clic en (" + e.getX() + ", " + e.getY() + ")");
  }
}
```

Quelques gestionnaires d'événements et méthodes d'enregistrements

Gestionnaire	Exemple de Composant	Méthodes d'enregistrement	Événement
ActionListener	**JButton** **JCheckbox** **JRadioButton** **JMenuItem** **JTextField**	**addActionListener**	- clic sur le bouton, la case à cocher, le bouton radio, l'élément de menu -l'utilisateur a tapé [Entrée] dans la zone de saisie
ItemListener	**JComboBox** **JList**	**addItemListener**	L'élément sélectionné a changé
InputMethodListener	**JTextField** **JTextArea**	**addMethodInputListener**	le texte de la zone de saisie a changé ou le curseur de saisie a changé de position
CaretListener	**JTextField** **JTextArea**	**addcaretListener**	Le curseur de saisie a changé de position
AdjustementListener	**JScrollBar**	**addAjustementListener**	La valeur du variateur a changé

Quelques gestionnaires d'événements et méthodes d'enregistrements

Gestionnaire	Exemple de Composant	Méthodes d'enregistrement	Événement
WindowsListener	**JFrame**	**addWindowsListener**	Evénement fenêtre
MouseMotionListener		**addMouseMotionListener**	La souris a bougé
MouseListener		**addMouseListener**	événements souris (clic, entrée/sortie du domaine d'un composant, bouton pressé, relâche)
FocusListener		**addFocusListener**	Evénement focus (obtenu, perdu)
KeyListener		**addKeyListener**	événement clavier (touche tapée, pressée, relâchée)

Quelques méthodes des gestionnaires d'événements

Interface	Méthodes
ActionListener	public void actionPerformed(ActionEvent)
AdjustmentListener	public void adjustmentValueChanged(AdjustmentEvent)
ComponentListener	public void componentHidden(ComponentEvent) public void componentMoved(ComponentEvent) public void componentResized(ComponentEvent) public void componentShown(ComponentEvent)
ContainerListener	public void componentAdded(ContainerEvent) public void componentRemoved(ContainerEvent)
FocusListener	public void focusGained(FocusEvent) public void focusLost(FocusEvent)
ItemListener	public void itemStateChanged(ItemEvent)
KeyListener	public void keyPressed(KeyEvent) public void keyReleased(KeyEvent) public void keyTyped(KeyEvent)

Quelques méthodes des gestionnaires d'événements

Interface	Méthodes
MouseListener	public void mouseClicked(MouseEvent) public void mouseEntered(MouseEvent) public void mouseExited(MouseEvent) public void mousePressed(MouseEvent) public void mouseReleased(MouseEvent)
MouseMotionListener	public void mouseDragged(MouseEvent) public void mouseMoved(MouseEvent)
TextListener	public void textValueChanged(TextEvent)
InputmethodListener	public void InputMethodTextChanged(InputMethodEvent) public void caretPositionChanged(InputMethodEvent)
CaretLisetner	public void caretUpdate(CaretEvent)
WindowListener	public void windowActivated(WindowEvent) public void windowClosed(WindowEvent) public void windowClosing(WindowEvent) public void windowDeactivated(WindowEvent) public void windowDeiconified(WindowEvent) public void windowIconified(WindowEvent) public void windowOpened(WindowEvent)

Gestionnaires de disposition

Ils se chargent :

- du placement initial des composants, lors des appels de la méthode **add**
- de donner une taille à chaque composant, en fonction de sa « taille préférée », de son contenu et de sa disposition relativement aux autres composants dans le même conteneur,
- du repositionnement des composants lorsque la taille ou la forme du conteneur change.

On change le placement des composants grâce à la méthode **setLayout()**.

Parmi ces gestionnaires on peut citer:

- FlowLayout
- BoxLayout
- BorderLayout
- GridLayout
- …

FlowLayout

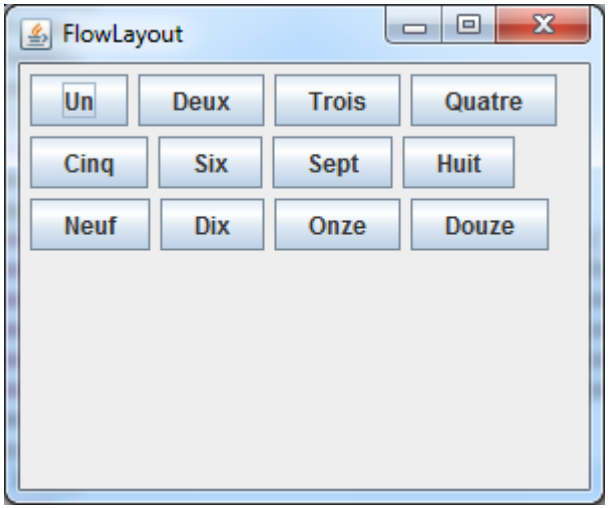
Un gestionnaire **FlowLayout** dispose les composants par lignes, de la gauche vers la droite et du haut vers le bas.

Exemple:

```
…
JPanel panneau = new JPanel();
panneau.setLayout(new FlowLayout(FlowLayout.LEFT));
panneau.add(new JButton("Un"));
panneau.add(new JButton("Deux"));
…
```

Remarque:

Le gestionnaire de disposition par défaut d'un JPanel est FlowLayout.

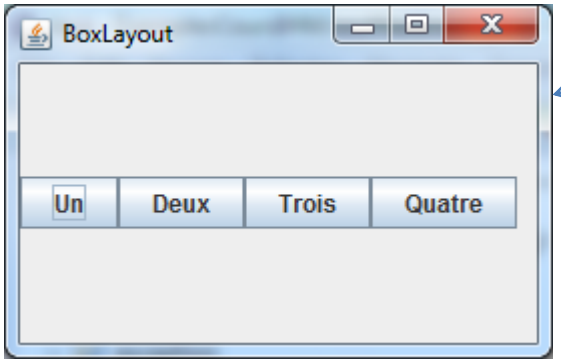


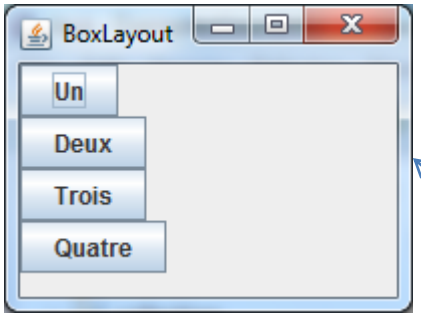
BoxLayout

Ce gestionnaire permet de disposer de multiples composants soit horizontalement (selon l'axe des **X**) ou verticalement (selon l'axe des **Y**).

Exemple:

```
…
JPanel panneau = new JPanel();
panneau.setLayout(new BoxLayout(panneau, BoxLayout.X_AXIS));
…
```

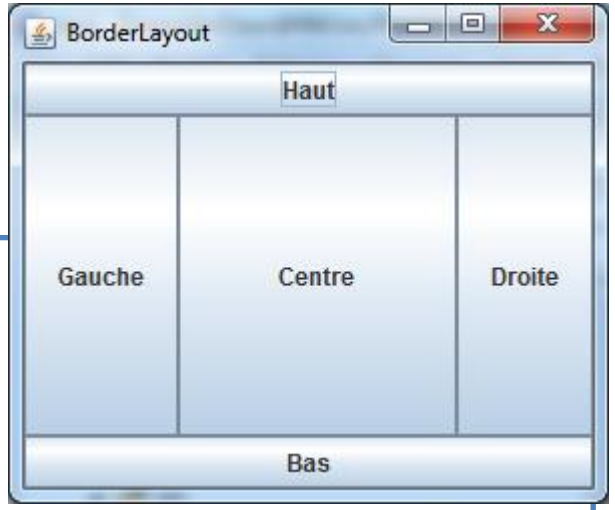





```
panneau.setLayout(new BoxLayout(panneau, BoxLayout.Y_AXIS));
```

BorderLayout

BorderLayout distingue cinq zones dans le conteneur auquel il est attaché : le nord, le sud, l'est, l'ouest et le centre.



Exemple:

```
…
JPanel panneau = new JPanel();
panneau.setLayout(new BorderLayout());
panneau.add(new JButton("Haut"), BorderLayout.NORTH);
panneau.add(new JButton("Droite"), BorderLayout.EAST);
panneau.add(new JButton("Bas"), BorderLayout.SOUTH);
panneau.add(new JButton("Gauche"), BorderLayout.WEST);
panneau.add(new JButton("Centre"), BorderLayout.CENTER);
…
```

Remarque:

Le gestionnaire de disposition **par défaut** du panneau de contenu d'un JFrame ou d'un JDialog est BorderLayout.

GridLayout

Un gestionnaire **GridLayout** organise les composants selon une grille rectangulaire ayant un nombre de lignes et de colonnes convenus lors de la création du gestionnaire.
Toutes les cases de cette grille ont les mêmes dimensions.

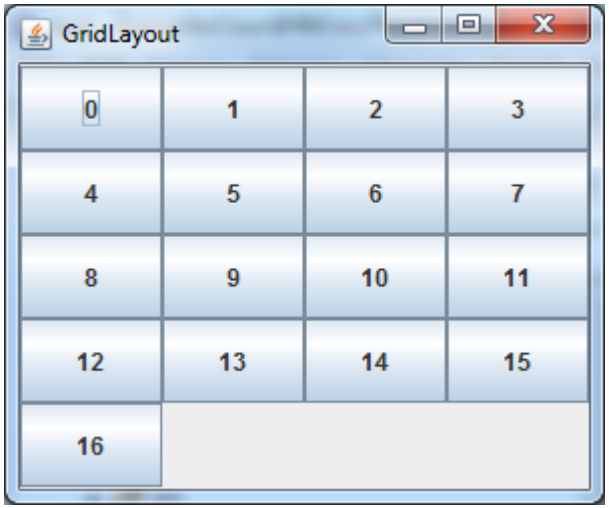
Exemple:

```
…
JPanel panneau = new JPanel();
panneau.setLayout(new GridLayout(5, 4));
for (int i = 0; i <= 16; i++)
panneau.add(new JButton("" + i));
…
```

Exemple:

On peut aussi fixer un espace, horizontal ou vertical, entre les cellules, par exemple dix pixels :

```
GridLayout disposition = new GridLayout(5,4,10,10);
```



Absence de gestionnaire de disposition

- On peut imposer à un objet « container » de n'avoir pas de gestionnaire en fixant son **LayoutManager** à la valeur **null**

```
Frame f = new Frame(); f.setLayout(null);
```

  - A la charge alors du programmeur de positionner chacun des composants « manuellement » en indiquant leur position absolue dans le repère de la fenêtre.
  - C'est à éviter, sauf dans des cas particuliers.
- Il est possible d'écrire ses propres LayoutManager…
- Les méthodes pour cela sont:

```
void setLocation(int x, int y), void setLocation(Point p)
void setSize(int width, int height), void setSize(Dimension d)
void setBounds(int x,int y,int width,int height), void setBounds(Rectangle r)
```

- Lorsque les composants ont été placés manuellement on a intérêt à interdire les changements de taille du conteneur. Cela s'obtient par : `setResizable(false);`

Les boîtes de dialogue: JOptionPane

Une boîte de dialogue est une petite fenêtre pouvant servir à plusieurs choses :

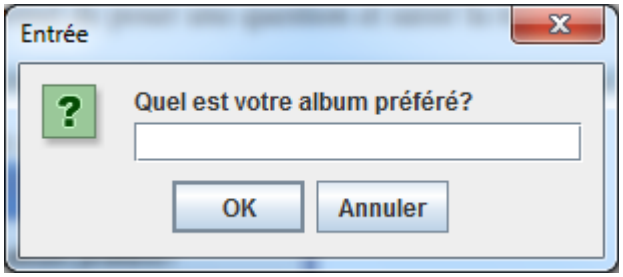
- ✓ afficher une information (message d'erreur, d'avertissement…),
- ✓ demander une validation, une réfutation ou une annulation,
- ✓ demander à l'utilisateur de saisir une information dont le système a besoin,
- ✓ etc.

La classe **JOptionPane** est utilisée via les méthodes statiques suivant:

Méthodes	Description	Retourne
showConfirmDialog	Demande une question qui se répond par oui/non/annulé.	Option
showInputDialog	Demande de taper une réponse.	String (Objet)
showMessageDialog	Transcrit un message à l'usager.	Rien
showOptionDialog	Une méthode générale réunissant les 3 précédentes.	Option

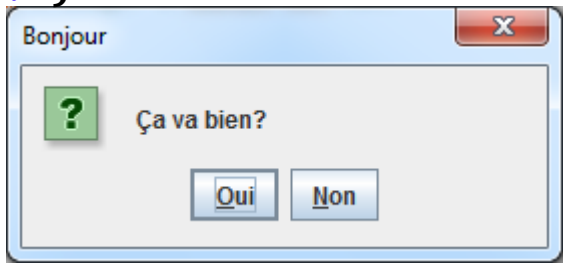
JOptionPane: Exemple

```
String rep = JOptionPane.showInputDialog("Quel est votre album préféré?");
```




```
JOptionPane.showMessageDialog(null,"Bonjour le monde", "Afficher un message",
JOptionPane.INFORMATION_MESSAGE);
```

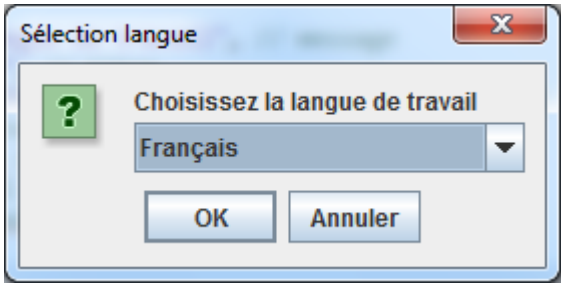

```
int rep = JOptionPane.showConfirmDialog(null,"Ça va bien?",
"Bonjour", JOptionPane.YES_NO_OPTION);
if(rep == JOptionPane.YES_OPTION)
  //traitement si l'usager a appuyé sur oui.
else
  //traitement si l'usager a appuyé sur non.
```



JOptionPane: Exemple

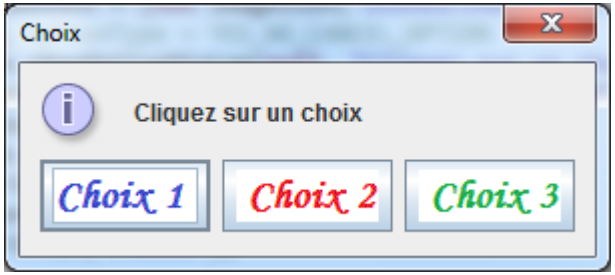
```
String[] langues = {"Français", "Anglais", "Espagnol"};
String s = (String)JOptionPane.showInputDialog(
 null, // parent
 "Choisissez la langue de travail", // message
 "Sélection langue", // titre
 JOptionPane.QUESTION_MESSAGE, // type de message
(icone)
 null, // icône, (null conserve icône par défaut)
 langues, /* choix (un tableau d'objets)
                 ou null si modifiable */
 langues[0]); // valeur par défaut

if(s!=null)
        //Traitement si pas annulé
```



JOptionPane: Exemple

```
Object[] optionsBoutons = {new ImageIcon("icones/choix1.gif"),new
ImageIcon("icones/choix2.gif"),new ImageIcon("icones/choix3.gif") };
// 3 boutons donc optionType = YES_NO_CANCEL_OPTION
int r=JOptionPane.showOptionDialog(null, "Cliquez sur un choix", "Choix" ,
JOptionPane.YES_NO_CANCEL_OPTION, JOptionPane.INFORMATION_MESSAGE, null,
optionsBoutons, optionsBoutons[0]) ;
if(r==JOptionPane.YES_OPTION )
  System.out.println("Choix1");
else if(r==JOptionPane.NO_OPTION )
  System.out.println("Choix2");
else
  System.out.println("Choix3");
```



Dessiner avec JAVA

Pour qu'un composant ait un dessin personnalisé il faut et il suffit de redéfinir la méthode **paint** dans la classe du composant.
La méthode **paint** n'est jamais explicitement appelée par le programme ; au lieu de cela, c'est la machine Java qui se charge de l'appeler, chaque fois que l'apparence du composant doit être refaite. Cela arrive principalement dans trois sortes de situations:

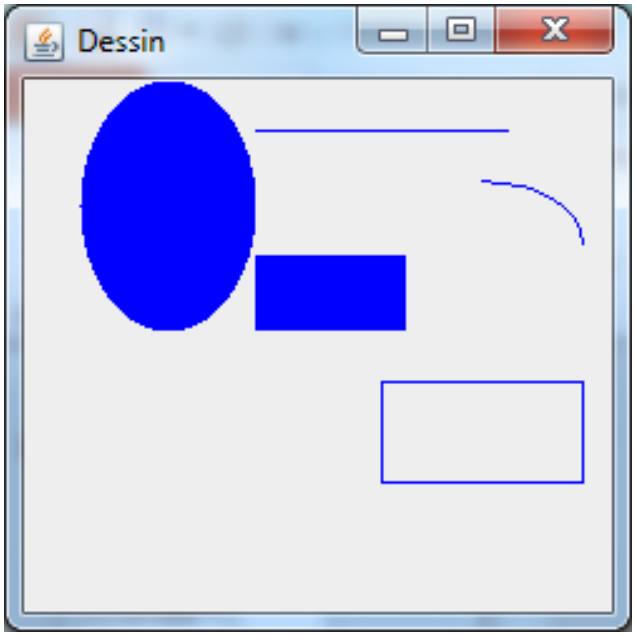
- ✓ Le composant doit être repeint suite à un événement extérieur à notre application.
- ✓ Le composant doit être repeint à cause d'un événement qui s'adresse bien à l'interface graphique de notre application (exemple : l'utilisateur a changé la taille de la fenêtre),
- ✓ Le composant doit être repeint car les données dont il exhibe une représentation ont changé (la machine Java ne peut pas deviner cela, le programmeur le signal par la méthode **repaint**).

La classe Graphics

Un contexte graphique est un objet qui encapsule l'ensemble des informations et des outils nécessaires pour effectuer des opérations graphiques. Les contextes graphiques sont instances de la classe **Graphics**, ou d'une de ses sous-classes comme Graphics2D.

Exemple:

```
public class Dessin extends JFrame {
  public Dessin() {
       setTitle("Dessin");
       setSize(250, 250);
       setDefaultCloseOperation(EXIT_ON_CLOSE);
       setVisible(true);}
  public void paint(Graphics g) {
       g.setColor(Color.BLUE);
       g.drawLine(100, 50, 200, 50);
       g.drawRect(150, 150, 80, 40);
       g.drawArc(150, 70, 80, 50, 0, 90);
       g.fillOval(30, 30, 70, 100);
       g.fillRect(100, 100, 60, 30);}
  public static void main(String[] args) {
       new Dessin();}
}
```

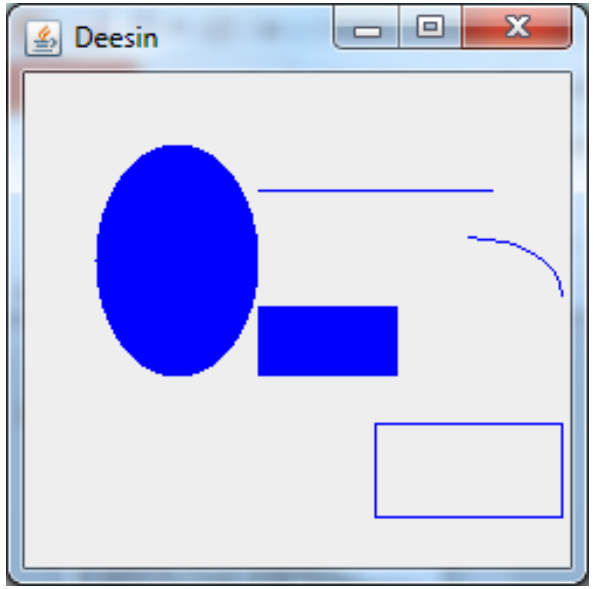


La classe Graphics

Pour les composants de Swing, il vaut souvent mieux redéfinir la méthode **paintComponent**

Exemple:

```
public class Dessin2 extends JPanel {
  public Dessin2() {
       JFrame fen=new JFrame("Deesin");
       fen.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       fen.setSize(250,250);
       fen.setVisible(true);
       fen.add(this);}
  public void paintComponent(Graphics g) {
       g.setColor(Color.BLUE);
       g.drawLine(100, 50, 200, 50);
       g.drawRect(150, 150, 80, 40);
       g.drawArc(150, 70, 80, 50, 0, 90);
       g.fillOval(30, 30, 70, 100);
       g.fillRect(100, 100, 60, 30);
  }
  public static void main(String[] args) {
       new Dessin2();
  }
}
```



Les images

Utilisation des icônes :

Les icônes sont des images de taille fixe, souvent petites, généralement utilisées pour décorer certains composants comme les étiquettes et les boutons.

Exemple:

```
JLabel label=new JLabel() ;
ImageIcon ic=new ImageIcon("nomImage.jpg");
label.setIcon(ic);
```

Chargement une image depuis un fichier

Le chargement de l'image est réalisé par la méthode **getImage** de la classe **Toolkit**.

uneImage = Toolkit.getDefaultToolkit().getImage(nom de fichier);

Les images: Exemple

```
public class AfficheurdImages extends JPanel {
  private Image uneImage;
  public AfficheurdImages() {
      uneImage=Toolkit.getDefaultToolkit().getImage("image.jpg");
      setPreferredSize(new Dimension(500,400));
  }
  public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawImage(uneImage,5,5,this);
  }
  public static void main(String[] args) {
      JFrame cadre = new JFrame("Afficher une image");
      cadre.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      cadre.getContentPane().add(new AfficheurdImages());
      cadre.pack();
      cadre.setVisible(true);
  }
}
```

TP11

Les interfaces graphiques en JAVA