

Licence Informatique

Librairie Java Swing & Programmation événementielle

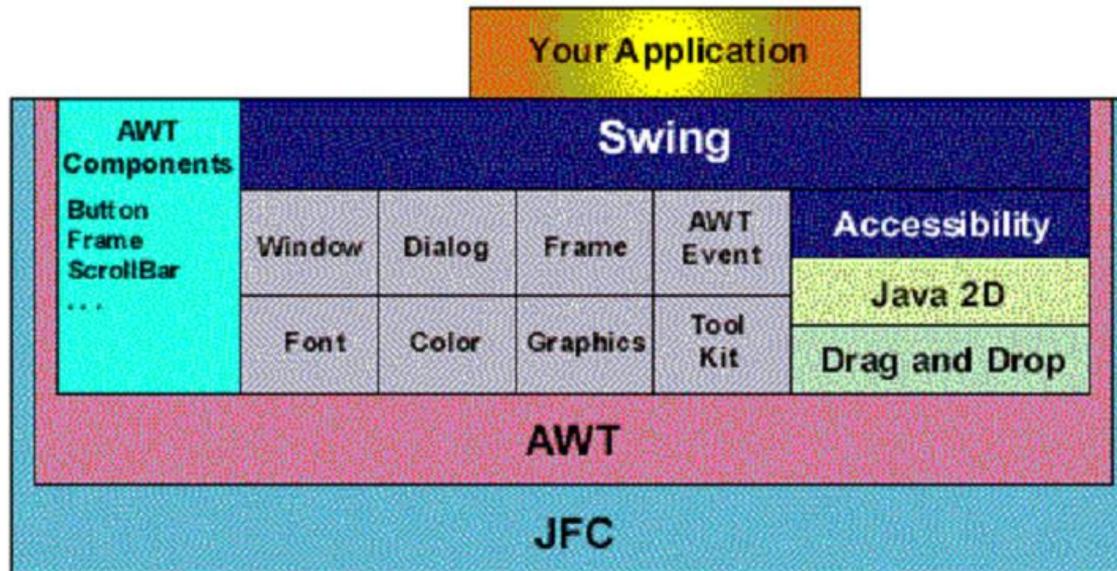
Mathieu RAYNAL

mathieu.raynal@irit.fr

<http://www.irit.fr/~Mathieu.Raynal>

AWT et Swing

- AWT – Abstract Window Toolkit: depuis Java 1.1
- SWING depuis Java 1.2



- Abstract Window Toolkit
- Conçue pour la portabilité des applications
- Ressources systèmes encapsulées par des abstractions
- Affichage délégué au système
 - Utilise le système de gestion des fenêtre du système
- Éléments graphiques ont l'apparence fournie par l'OS

SWING

- Extension d'AWT
 - Architecture plus souple
 - Plus de possibilités graphiques
- Affichage non délégué au système.
- Un élément Swing peut
 - avoir la même apparence d'un système à l'autre
 - être paramétré: Pluggable Look and Feel
 - n'avoir aucun équivalent dans le système d'exploitation sur lequel il s'exécute

Le système de fenêtrage

Fenêtre principale - **JFrame**

- Toute interface graphique a une fenêtre principale

- Classe **JFrame**

- Principaux constructeurs

```
JFrame(String titre)
```

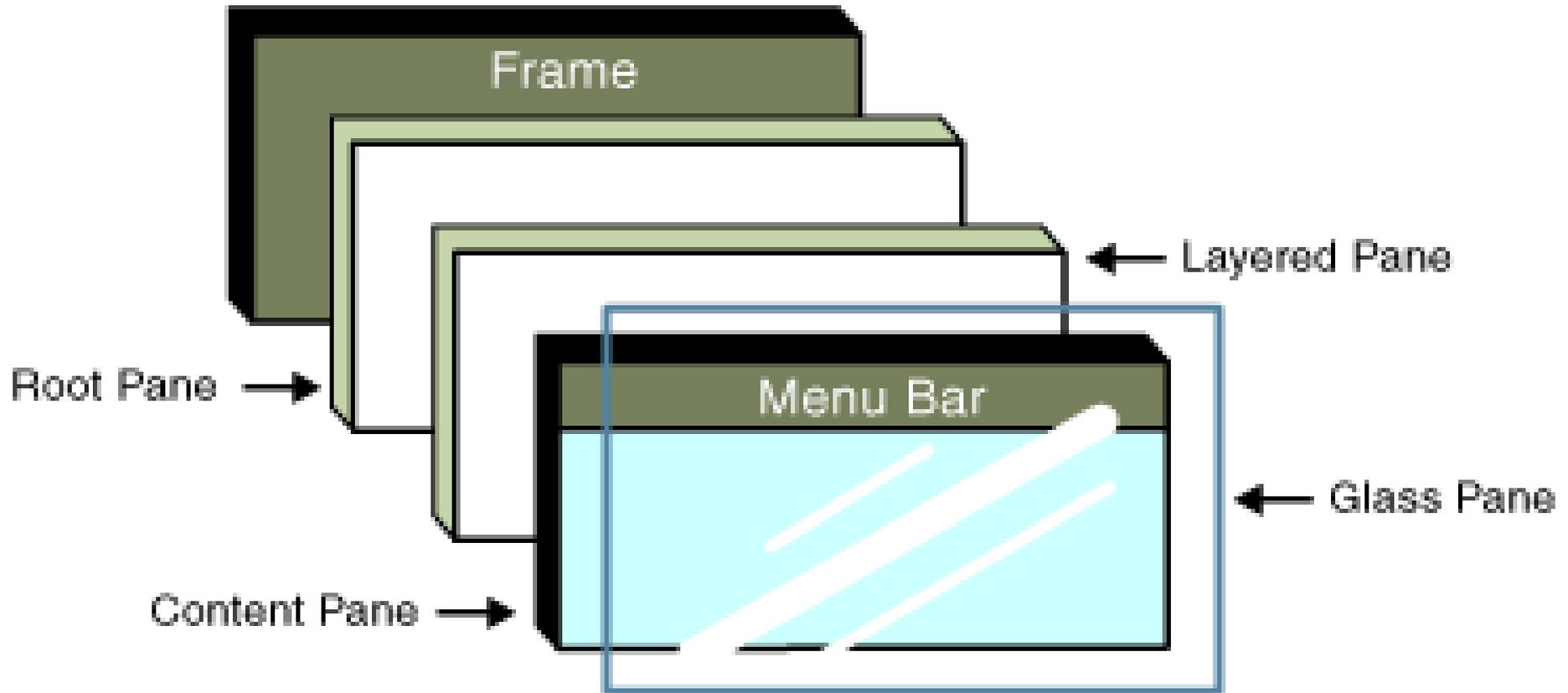
```
JFrame()
```

- Mettre un titre

```
void setTitle(String titre)
```



JRootPane et ses différentes couches



Accéder aux éléments de la fenêtre

- Accéder au Menu

```
JMenuBar getJMenuBar()
```

```
void setJMenuBar(JMenuBar)
```

- Accéder au conteneur principal

```
Container getContentPane()
```

```
void setContentPane(Container)
```

Positionner la fenêtre à l'écran

- Positionner la fenêtre à l'écran

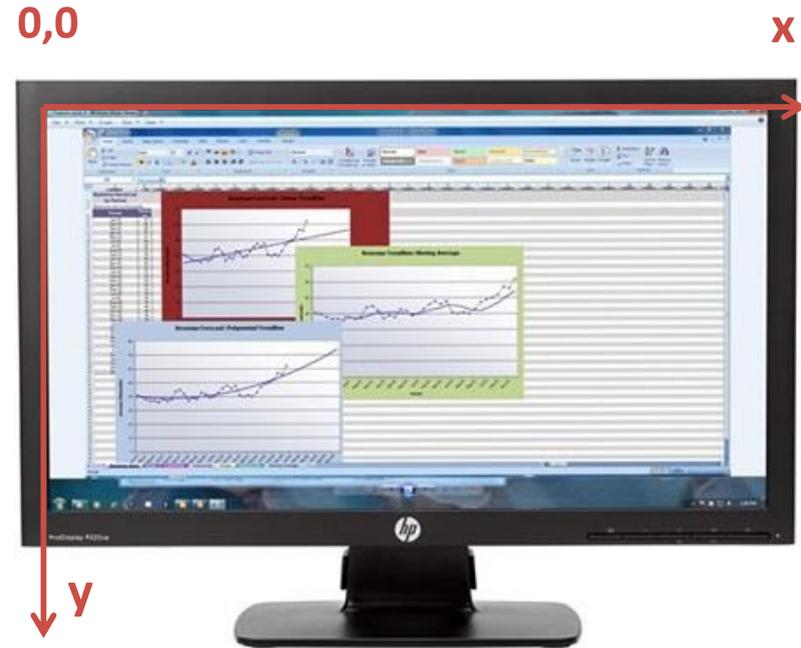
```
void setLocation(int x, int y)
```

- Centrer la fenêtre à l'écran

```
void setLocationRelativeTo(null)
```

- Connaitre la taille de l'écran

```
Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
```



Positionnement et dimension de la fenêtre

- Dimensionner la fenêtre

```
void setSize(int longueur, int hauteur)
```

- Donner la dimension optimale à la fenêtre

```
void pack()
```

- Positionner et dimensionner la fenêtre

```
void setBounds(int x, int y, int longueur, int hauteur)
```

Fermer correctement la fenêtre ... et l'application

- Opération à effectuer au moment de l'appui du bouton rouge

```
void setDefaultCloseOperation(int operation)
```

- DO_NOTHING_ON_CLOSE
- HIDE_ON_CLOSE
- DISPOSE_ON_CLOSE
- EXIT_ON_CLOSE

- Fermer l'application

```
void dispose()
```

- Libère la mémoire

Modifier les paramètres de la fenêtre

- Changer l'icône de l'application

```
void setIconImage(Image icone)
```

- Ne pas permettre le redimensionnement de la fenêtre

```
void setResizable(boolean choix)
```

- Mettre la fenêtre en premier plan

```
void setAlwaysOnTop(boolean choix)
```

- Supprimer les bordures de la fenêtre

```
void setUndecorated(boolean choix)
```

Principales étapes de construction

- Créer de la fenêtre
 - La paramétrer
 - Paramétrer la fermeture de la fenêtre
- Créer les composants
 - Les paramétrer
 - Les positionner
- Positionner la fenêtre et définir sa taille
- Afficher la fenêtre

```
void setVisible(boolean visibilité)
```

Exemple

```
private static void createAndShowGUI() {
    JFrame frame = new JFrame("Exemple de création de fenêtre");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 600);
    // Création, paramétrage et positionnement des composants
    frame.setLocationRelativeTo(null);
    frame.setVisible(true);
}

public static void main(String[] args) {
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
        public void run() {
            createAndShowGUI();
        }
    });
}
```

Boîte de dialogue - **JDialog**

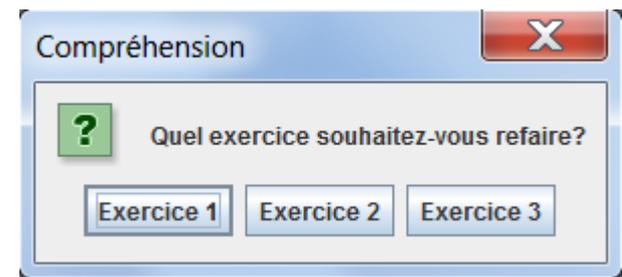
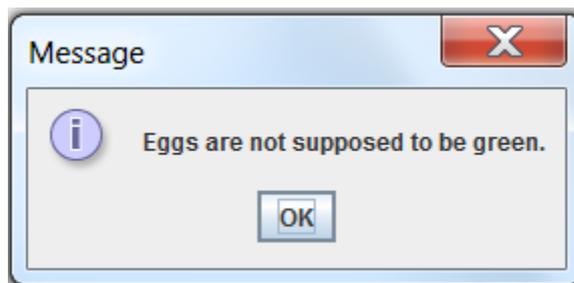
- Exemple d'utilisation
 - Login,
 - Enregistrement d'un document,
 - ...
- Vient en complément d'une fenêtre principale
 - Même fonctionnement que la **JFrame**
 - Principale différence : possibilité de rendre l'interface **modale**

- Principal constructeur

```
JDialog(Window owner, String title, boolean modal)
```

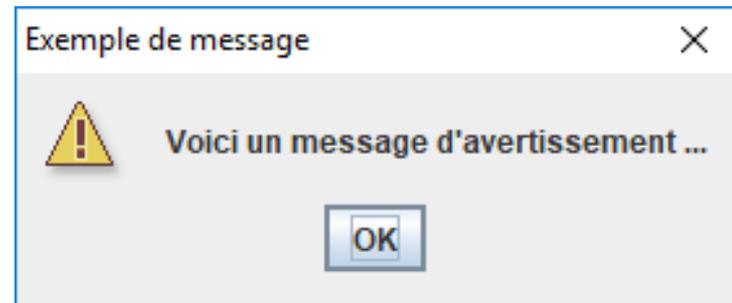
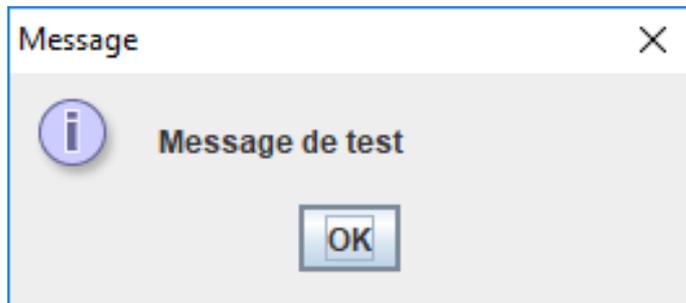
Boîtes de dialogue prédéfinies - **JOptionPane**

- Création de boîtes de dialogue standardisées modales
- Classe **JOptionPane**
 - Méthodes statiques qui créent directement une boîte de dialogue
- Plusieurs types
 - Messages d'erreur, d'information
 - Demande d'information
 - Choix



JOptionPane - showMessageDialog

- public static void **showMessageDialog**(
Component parentComponent,
Object message,
[String title,
int messageType],
[Icon icon])



showMessageDialog - attributs

- **parentComponent** : fenêtre mère (JFrame, JDesktopPane, ...)
- **message** : contenu du message
- **title** : titre de la fenêtre de dialogue
- **messageType** : choix de l'icône
 - Les différentes valeurs possibles sont enregistrées sous forme de *static final int* dans **JOptionPane**
 - ERROR_MESSAGE
 - INFORMATION_MESSAGE
 - WARNING_MESSAGE
 - QUESTION_MESSAGE
 - PLAIN_MESSAGE
- **icon** : personnaliser l'icône avec une image personnelle

Icons used by JOptionPane

Icon description	Java look and feel	Windows look and feel
question		
information		
warning		
error		

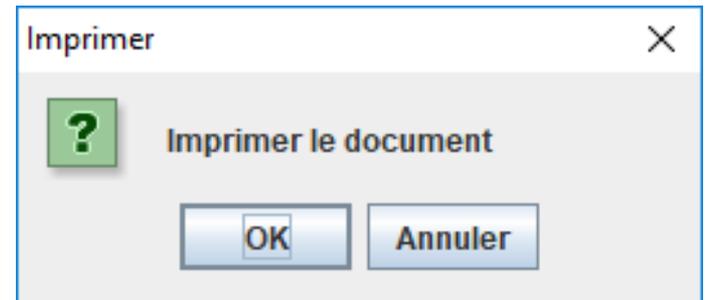
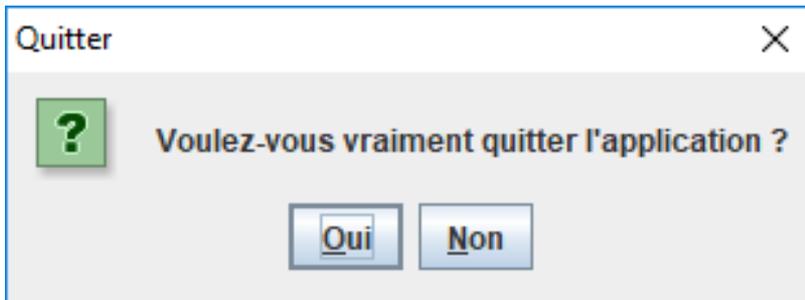
Exemple

```
ImageIcon ii = new ImageIcon("index.jpg");  
Image i = ii.getImage().getScaledInstance(80, -1, java.awt.Image.SCALE_SMOOTH);  
JOptionPane.showMessageDialog(ExempleOptionPane.this,  
    "J'aime le JAVA",  
    "Mon langage favori",  
    JOptionPane.PLAIN_MESSAGE,  
    new ImageIcon(i));
```



JOptionPane - showConfirmDialog

- public static int **showConfirmDialog**(
Component parentComponent,
Object message,
[String title,
int optionType],
[int messageType],
[Icon icon])



showConfirmDialog – Attributs

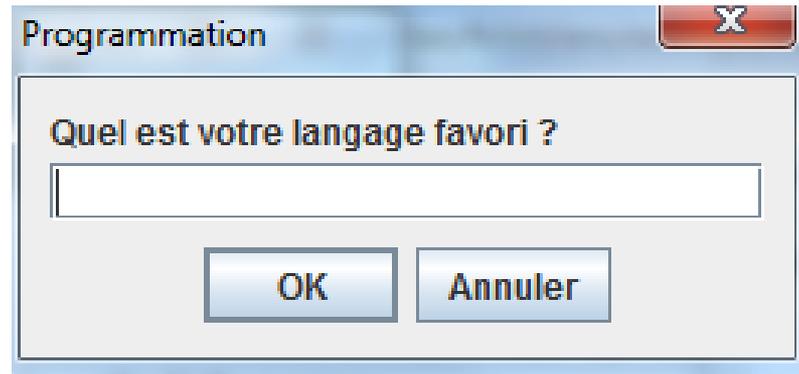
- **optionType** : Pour configurer le nombre et le type de boutons
 - Les différentes valeurs possibles sont enregistrées sous forme de *static final int* dans **JOptionPane**
 - **DEFAULT_OPTION**
 - 1 bouton : « ok »
 - **YES_NO_OPTION**
 - 2 boutons : « oui » , « non »
 - **YES_NO_CANCEL_OPTION**
 - 3 boutons : « oui », « non », « annuler »
 - **OK_CANCEL_OPTION**
 - 2 boutons : « ok », « annuler »

showConfirmDialog– Valeur de retour

- Lors de l'appui sur un des boutons de la boîte de dialogue
 - La boîte de dialogue se ferme
 - un entier est renvoyé
 - sa valeur dépend du bouton sur lequel on a appuyé
 - Les différentes valeurs possibles sont enregistrées sous forme de *static final int* dans **JOptionPane**
 - YES_OPTION
 - NO_OPTION
 - CANCEL_OPTION
 - OK_OPTION
 - CLOSED_OPTION

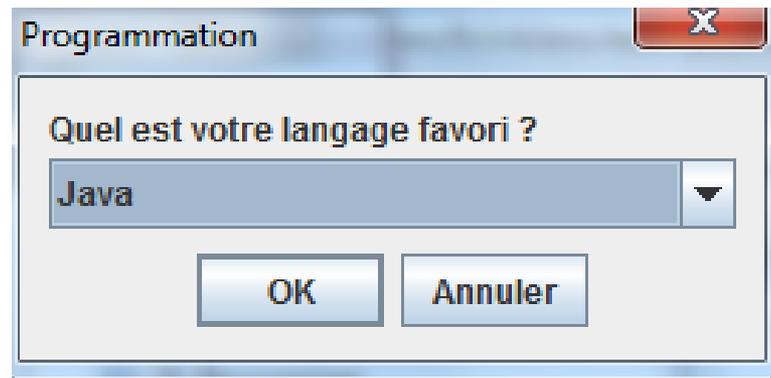
JOptionPane - showInputDialog

- public static String **showInputDialog**(
Component parentComponent,
Object message,
String title, int messageType,
Object initialValue)



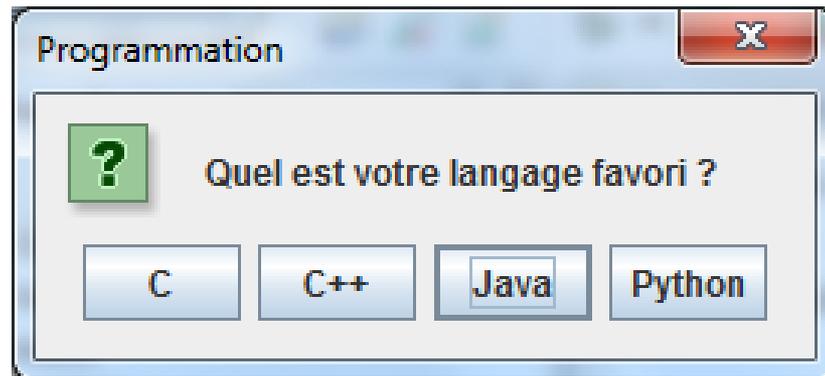
JOptionPane - showInputDialog

- public static Object **showInputDialog**(
Component parentComponent,
Object message,
String title, int messageType,
Icon icon, Object[] selectionValues,
Object initialSelectionValue)

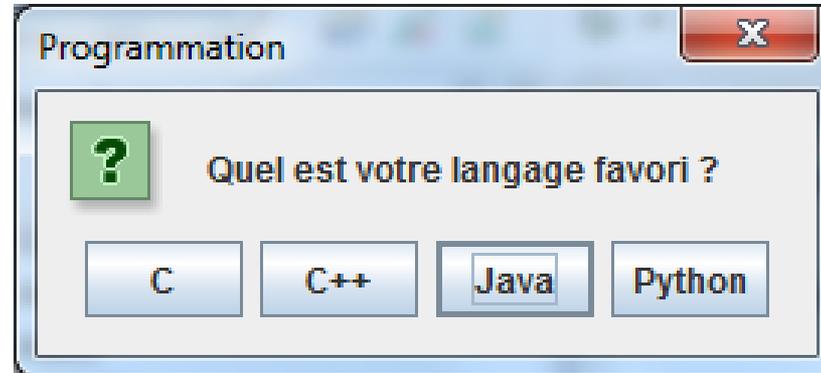


JOptionPane - showOptionDialog

- `public static int showOptionDialog(`
Component parentComponent,
Object message,
String title,
int optionType,
int messageType,
Icon icon, Object[] options,
Object initialValue)



Exemple



```
String[] liste = {"C","C++","Java","Python"};
int n = JOptionPane.showOptionDialog(ExempleOptionPane.this,
                                     "Quel est votre langage favori ?",
                                     "Programmation",
                                     JOptionPane.DEFAULT_OPTION,
                                     JOptionPane.QUESTION_MESSAGE,
                                     null, liste, "Java");
```

Interface de sélection de fichiers - **JFileChooser**

- Principaux constructeurs

JFileChooser()

JFileChooser(File currentDir)

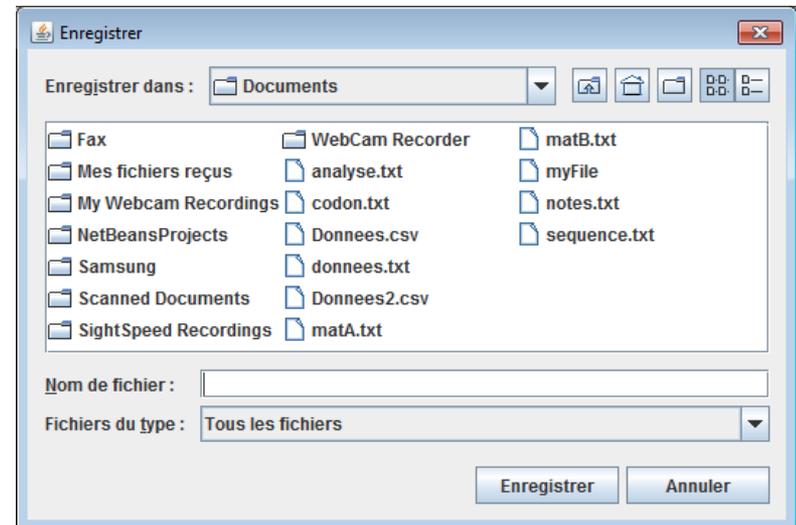
JFileChooser(String currentDirPath)

- Afficher la boîte de dialogue

int showOpenDialog(Component c)

int showSaveDialog(Component c)

int showDialog(Component c, String s)



JFileChooser

- La valeur de retour
 - Les différentes valeurs possibles sont enregistrées sous forme de *static final int* dans **JFileChooser**
 - **APPROVE_OPTION**
 - **CANCEL_OPTION**

- Récupérer la sélection

- un fichier

```
File getSelectedFile()
```

- Un ensemble de fichiers

```
File[] getSelectedFiles()
```

Filtrer le sélecteur en fonction d'un mode

- Filtrer en fonction du type : fichier ou répertoire

```
void setFileSelectionMode(int mode)
```

- Les différentes modes possibles sont enregistrées sous forme de *static final int* dans **JFileChooser**
 - FILES_AND_DIRECTORIES
 - FILES_ONLY
 - DIRECTORIES_ONLY

Filtrer le sélecteur en fonction d'un type de fichier

- Filtrer en fonction d'un filtre particulier

```
void addChoosableFileFilter(FileFilter mode)
```

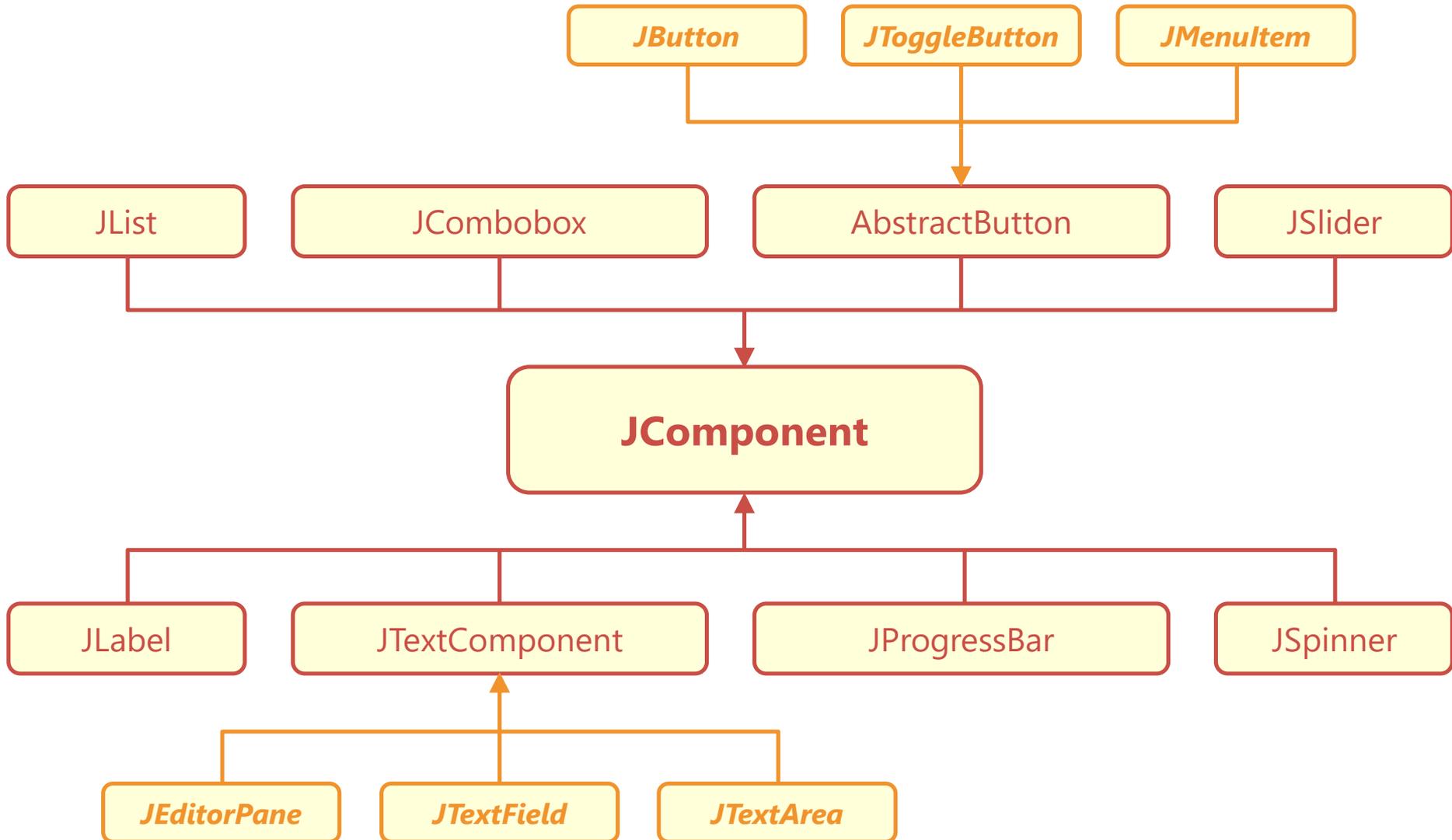
- Créer le filtre : créer une classe qui étend de **FileFilter**
 - Masquer la méthode

```
boolean accept(File fichier)
```

- Renvoie **true** si le fichier doit être affiché
- **false** sinon

Principaux composants interactifs

Hiérarchie des composants



Caractéristiques communes à tous les composants

- Apparence
 - Couleur
 - Fonte de caractères
 - Bordure
- Etat
 - Visible / Invisible
 - Actif / Inactif
- Taille
- Bulle d'aide
- Curseur

Utiliser des couleurs

- Couleur d'arrière plan

```
Color getBackground()
```

```
void setBackground(Color c)
```

- Couleur d'écriture

```
Color getForeground()
```

```
void setForeground(Color c)
```

- Gestion de la transparence

```
void setOpaque(boolean)
```

```
boolean isOpaque()
```

Créer une couleur - **Color**

- Elles sont gérées par la classe `Color`

- Objet construit

- à partir de ses composantes

- rouge/vert/bleu
- alpha en option

- Soit avec des float (0.0-1.0) ou des int (0-255)

`Color(float r, float g, float b)`

`Color(float r, float g, float b, float a)`

`Color(int r, int g, int b)`

`Color(int r, int g, int b, int a)`

- Ensemble de couleurs prédéfinies sous forme de `static`

- `Color.BLACK`

- `Color.GRAY`

- `Color.ORANGE`

- `Color.BLUE`

- `Color.GREEN`

- `Color.PINK`

- `Color.CYAN`

- `Color.LIGHT_GRAY`

- `Color.RED`

- `Color.DARK_GRAY`

- `Color.MAGENTA`

- `Color.WHITE`

- `Color.YELLOW`

Utiliser des fontes

- Modifier la fonte d'un composant

```
void setFont(Font)
```

- Connaitre la fonte utilisée par un composant

```
Font getFont()
```

- Connaitre les fontes disponibles dans l'environnement

```
GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();  
Font [] allFonts = ge.getAllFonts();
```

Créer une fonte - **Font**

- Classe **Font** pour gérer les polices de caractères

```
Font(String name, int style, int size)
```

- Différentes familles de fontes sont enregistrées sous forme de *static final String* dans **Font**
 - SANS_SERIF
 - SERIF
 - DIALOG_INPUT
 - DIALOG
 - MONOSPACED
- Différentes styles sont enregistrées sous forme de *static final int* dans **Font**
 - BOLD
 - PLAIN
 - ITALIC

- Pour connaître l'ensemble des fontes installées

```
Font [] allFonts = GraphicsEnvironment.getLocalGraphicsEnvironment().getAllFonts();
```

Utiliser une bordure

- Modifier la bordure d'un composant

```
void setBorder(Border bordure)
```

- Connaitre la bordure utilisée par un composant

```
Border getBorder()
```

Créer une bordure - **BorderFactory**

- Méthodes statiques dans **BorderFactory**
 - createBevelBorder
 - createCompoundBorder
 - createDashedBorder
 - createEmptyBorder
 - createEtchedBorder
 - createLineBorder
 - createLoweredSoftBevelBorder
 - createMatteBorder
 - createRaisedBevelBorder
 - createSoftBevelBorder
 - createStrokeBorder
 - createTitledBorder

Etat d'un composant

- Visibilité

```
void setVisible(boolean b)
```

- Actif

```
void setEnabled(boolean b)
```

Bulle d'aide- Tooltip

- Apparaît au dessus du composant
 - lorsque le pointeur de la souris reste immobile 1 seconde au-dessus du composant

```
public void setToolTipText(String text)
```



Modifier le curseur

- Modifier le curseur

```
void setCursor(Cursor c)
```

- Différents types de curseur sont enregistrés sous forme de *static final int* dans **Cursor**
 - CROSSHAIR_CURSOR
 - CUSTOM_CURSOR
 - DEFAULT_CURSOR
 - XX_RESIZE_CURSOR
 - HAND_CURSOR
 - MOVE_CURSOR
 - TEXT_CURSOR
 - WAIT_CURSOR

Utiliser des images

- Les composants qui affichent de l'information peuvent le faire sous la forme
 - d'une chaîne de caractères (String)
 - d'une image (interface Icon)

- La classe `ImageIcon` implémente l'interface `Icon`

```
ImageIcon(String nomFichier);  
ImageIcon(URL location);
```

- Possibilité de redimensionner une image via la classe `Image`

```
getScaledInstance(int longueur, int hauteur, Image.SCALE_DEFAULT)
```

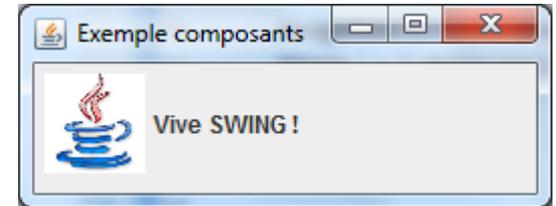
- Mettre un nombre négatif à longueur ou hauteur pour conserver les proportions

Principaux composants interactifs

Afficher de l'information

Afficher une information : **JLabel**

- La classe **JLabel** permet d'afficher une information
 - sous la forme d'une chaîne de caractères
 - et/ou d'une image
- Principaux constructeurs



JLabel(String text)

JLabel(String text, int horizontalAlignment)

JLabel(Icon image)

JLabel(Icon image, int horizontalAlignment)

JLabel(String text, Icon icon, int horizontalAlignment)

- Les différents alignements possibles sont enregistrées sous forme de *static final int* dans **JLabel**
 - LEFT
 - CENTER
 - RIGHT

JEditorPane

- Composant utilisé pour visualiser des textes

```
JEditorPane(String type, String text)
```

- Trois types de texte possibles pour la mise en page

- text/plain
- text/rtf
- text/html

- Méthodes pour modifier le texte affiché

```
setText(String s)
```

```
read(Reader r)
```

```
setPage(URL url)
```

JTextPane

- Hérite de **JEditorPane**
- Permet d'ajouter des styles de mise en page via un **StyleDocument**

```
JTextPane jTextPane = new JTextPane();

Style default = jTextPane.getStyle("default");
Style monStyle = jTextPane.addStyle("monStyle", default);
StyleConstants.setForeground(monStyle, Color.RED);
StyleConstants.setFontSize(monStyle, 25);

String texte = "Bonjour";
StyledDocument sDoc = (StyledDocument)jTextPane.getDocument();
try {sDoc.insertString(0, texte, monStyle); } catch (BadLocationException e) {...}
```

Barre de progression - **JProgressBar**

- La classe **JProgressBar**

```
JProgressBar(int orient)
```

```
JProgressBar(int min, int max)
```

```
JProgressBar(int orient, int min, int max)
```



- Modifier la valeur

```
void setValue(int val)
```

- Modifier le texte

```
void setStringPainted(boolean b)
```

```
void setString(String s)
```

Principaux composants interactifs

Saisir du texte

JTextField

- Champ de saisie sur une seule ligne
- Principaux constructeurs

```
JTextField()  
JTextField(int col)  
JTextField(String text)  
JTextField(String text, int col)
```



Un texte ...

- Récupérer le texte saisi

```
String getText()
```

- Modifier le texte affiché

```
void setText(String text)
```

JFormattedTextField

- Permet de formater un saisie
 - Nombre, Date, Pourcentage
- Principaux constructeurs

```
JFormattedTextField()
```

```
JFormattedTextField(Format f)
```

```
JFormattedTextField(Object o)
```

- Récupérer une valeur

```
Object getValue()
```

JPasswordField

- Même fonctionnement que le **JTextField**
 - Les caractères sont masqués par un caractère unique



- Mêmes constructeurs que le **JTextField**
- Récupérer le texte saisi
- Modifier le caractère utilisé pour masquer

```
public char[] getPassword()
```

```
public setEchoChar(char c)
```

JTextArea

- Champ de saisie sur plusieurs lignes

```
JTextArea(int nbLignes, int nbColonnes)
```

```
JTextArea(String texte)
```

```
JTextArea(String texte, int nbLignes, int nbColonnes)
```

- Ajouter du texte

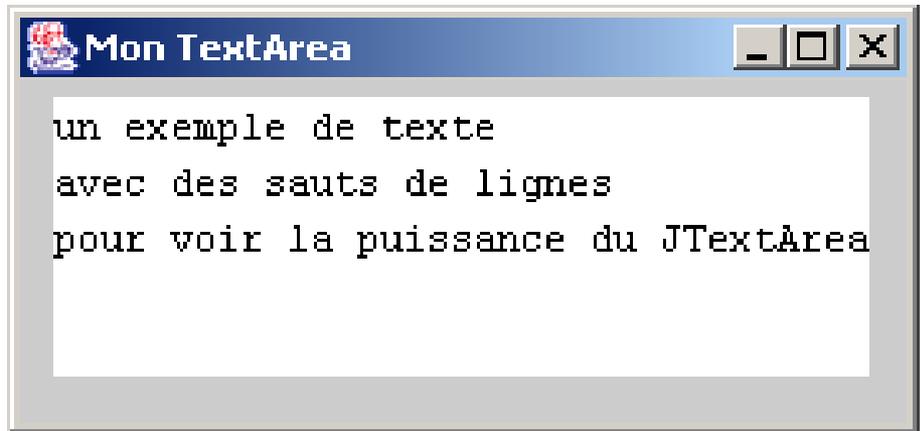
```
void append(String text)
```

- Modifier le texte

```
void setText(String text)
```

- Récupérer le texte

```
String getText()
```



Principaux composants interactifs

Boutons

AbstractButton

- Classe abstraite
- Activation du bouton
 - via la souris
 - par combinaison de touche : alt + mnemonic (autre touche du clavier)

```
void setMnemonic(int mnemonic)
```

- L'entier est géré par une constant de `java.awt.event.KeyEvent`

- Gestion de l'état du bouton : sélectionné ou non

```
public void setSelected(boolean b)
```

JButton

- Bouton simple à un seul état
- Principaux constructeurs

```
JButton()  
JButton(Icon icon)  
JButton(String text)  
JButton(String text, Icon icon)
```

- Possibilité de changer l'icône du bouton en fonction de son état

```
void setIcon(Icon i)  
void setSelectedIcon(Icon i)  
void setPressedIcon(Icon i)  
void setDisabledIcon(Icon i)
```

JToggleButton

- Boutons à deux états
 - L'aspect visuel du bouton reflète l'état d'une fonctionnalité
 - Exemple : barres d'outils d'un éditeur de texte (gras, italique, surligné)
 - Deux états / aspects
 - Sélectionné
 - Non sélectionné
- Principaux constructeurs

```
JToggleButton(String text)
```

```
JToggleButton(String text, boolean selected)
```

```
JToggleButton(Icon icon)
```

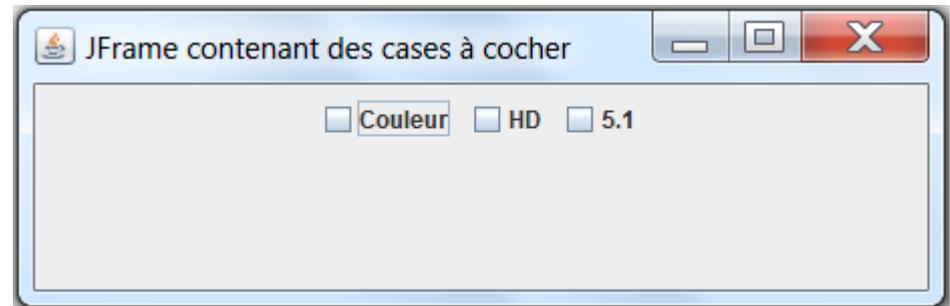
```
JToggleButton(Icon icon , boolean selected)
```

```
JToggleButton(String text, Icon icon)
```

```
JToggleButton(String text, Icon icon , boolean selected)
```

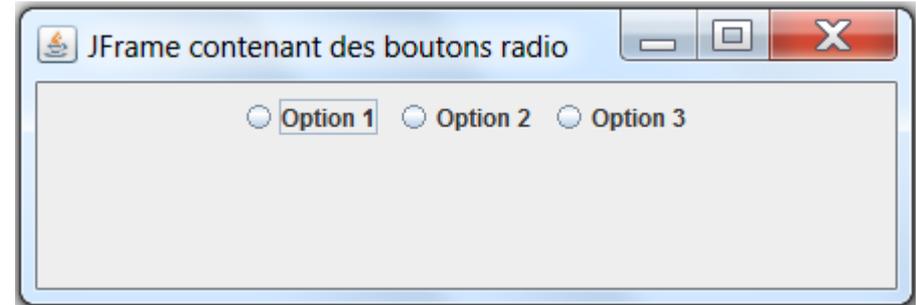
Les cases à cocher - JCheckBox

- Plusieurs cases peuvent être cochées simultanément
- Hérite de **JToggleButton**
→ Constructeurs identiques



Case à cocher à choix unique - **JRadioButton**

- Hérite de **JToggleButton**
→ Constructeurs identiques



- Le choix unique est géré par un **ButtonGroup**
 - ButtonGroup : composant « logique »
 - Pas besoin de l'ajouter à l'interface graphique
 - Les JRadioButton doivent être ajouté au ButtonGroup

```
void add(AbstractButton b)
```

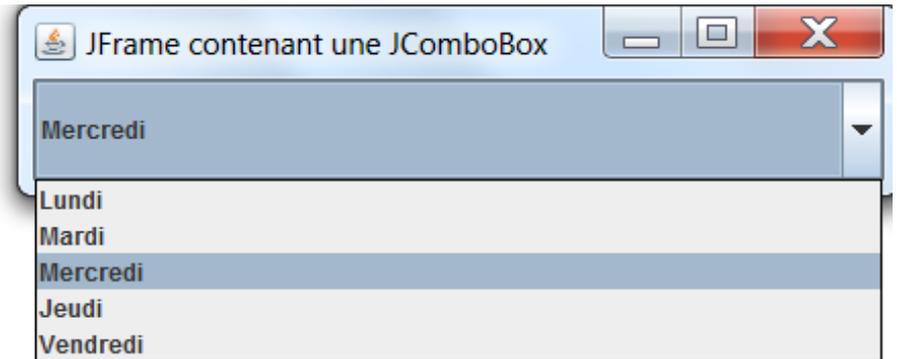
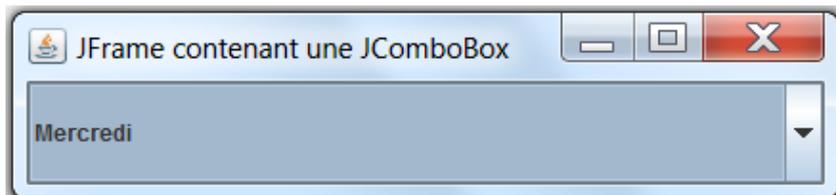
Principaux composants interactifs

Sélection d'une valeur

Liste déroulante - **JComboBox**

- Liste déroulante : Un seul item est affiché
- Principaux constructeurs

```
public JComboBox(E[] items)  
public JComboBox(Vector<E> items)
```



JComboBox

- Pré-sélection

```
void setSelectedIndex(int anIndex)
```

- Nombre d'items à afficher quand la liste est déroulée

```
void setMaximumRowCount(int count)
```

- Connaitre l'élément sélectionné

```
Object getSelectedItem()
```

JList

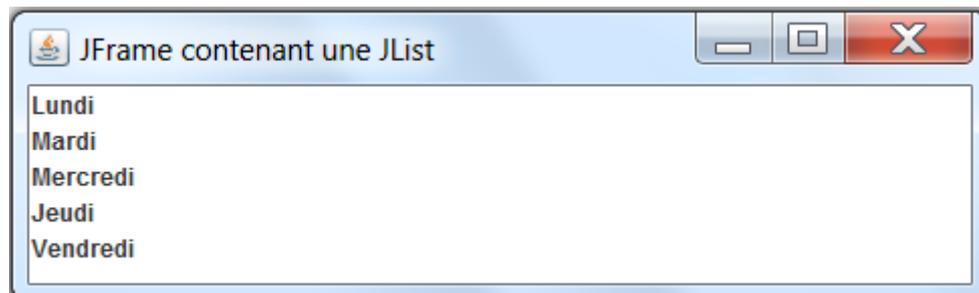
- Liste de valeurs

```
public JList(E[] items)  
public JList(Vector<E> items)
```

- Pré-sélection

```
void setSelectedIndex(int anIndex)
```

- La liste doit être ajoutée à un container de type **JScrollPane** pour faire défiler les items



JSlider

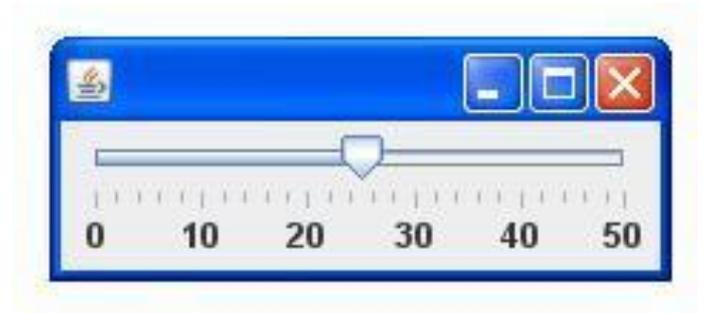
- Sélecteur de valeurs sur une barre horizontale ou verticale
- Principaux constructeurs

JSlider(int orient)

JSlider(int min, int max)

JSlider(int min, int max, int val)

JSlider(int orient, int min, int max, int val)



- Méthodes pour l'affichage de la barre

void **setMajorTickSpacing**(int max)

void **setMinorTickSpacing**(int min)

void **setPaintTicks**(boolean b)

void **setPaintLabels**(boolean b)

JSpinner

- Sélecteur de valeurs

- Constructeurs

```
JSpinner()  
JSpinner(SpinnerModel model)
```

- Sélection d'une valeur

```
Object getValue()  
Object getNextValue()  
Object getPreviousValue()
```

