Java Avancé - Cours 3

Plan

1 Rappel sur AWT

AWT (Abstract Window Toolkit) est l'ensemble des classes Java originales qui permet de faire des interfaces graphiques. Ces classes sont situées dans le package `java.awt`. AWT repose sur les widgets systèmes. Par widget on entend les boutons, les menus, etc...

Pour être portable au niveau également de l'apparence, et bien dégager la notion de modèle (par exemple une structure d'arbre) par rapport à son rendu (comment l'arbre va être affiché à l'écran), les classes Swing ont été créées. Ces classes sont dans le package `javax.swing`.

Le passage de AWT à Swing se fait facilement. Pour beaucoup de classes il suffit de rajouter un `J` devant le nom. Par exemple en AWT on dispose de `Applet`, `Panel`, `Frame`. Les classes équivalentes sont disponibles en Swing : `JApplet`, `JPanel`, `JFrame`. De plus on trouve des nouveaux composants très pratiques, comme par exemple : `JTree` pour afficher un arbre, `JTabbedPane` pour afficher des onglets.

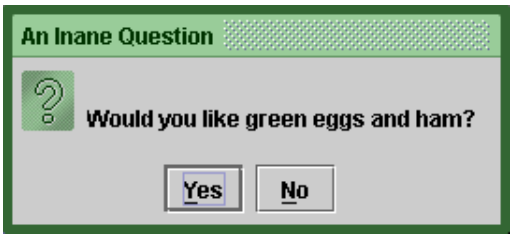
2 Présentation de Swing

Dans cette section plusieurs images proviennent de l'URL:

```
http://java.sun.com/docs/books/tutorial/uiswing/components/components.html
```

2.1 Containers de haut niveau

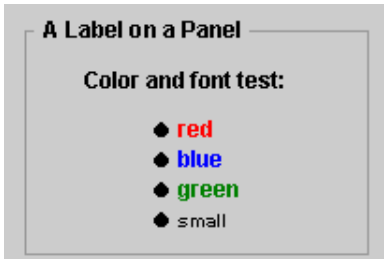
JApplet : Cette classe est une sous classe de `java.applet.Applet`. Elle permet de faire des applets où les composants sont des widgets Swing et non AWT. Son usage est similaire à celui de `java.applet.Applet`.
JDialog : C'est la classe swing qui permet de créer des dialogues sur mesure. Pour des dialogues simples on préfèrera utiliser la classe `JOptionPane`. Typiquement l'image ci-contre est obtenue à l'aire de `JOptionPane`. Il existe une classe toute faite pour obtenir une boite de dialogue permettant de choisir un fichier `JFileChooser`.



JFrame : C'est la classe qui remplace Frame dans l'AWT. Il s'agit d'une fenêtre avec les boutons standards.

2.2 Containers génériques

JPanel : Il s'agit d'une classe destinée à contenir d'autres classes. Les attributs généralement modifiés sont la couleur du fond, les bords, ainsi que l'appel à la méthode `void setLayout(LayoutManager mgr)` pour gérer la manière dont les composants sont placés.

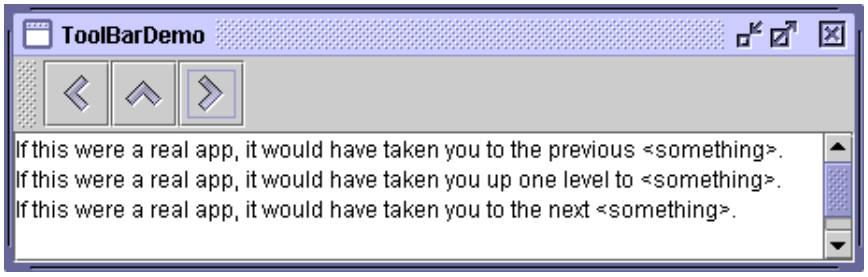


JScrollPane : Permet de visualiser un objet trop grand. Au lieu d'écrire directement dans un `JPanel` :

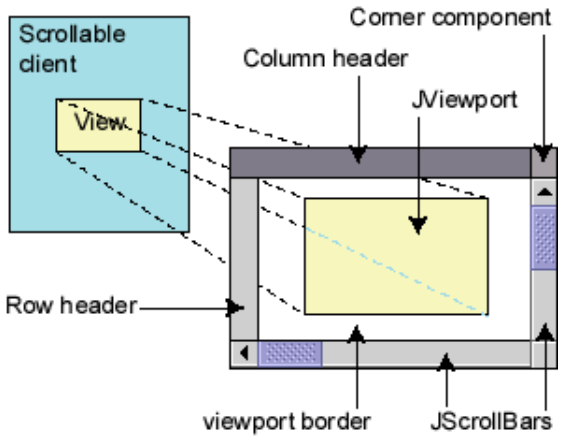
```
textArea = new JTextArea(5, 30);
add(textArea, BorderLayout.CENTER);
```

On peut le mettre préalablement dans un `JScrollPane` :

```
textArea = new JTextArea(5, 30);
JScrollPane scrollPane = new JScrollPane(textArea);
add(scrollPane, BorderLayout.CENTER);
```

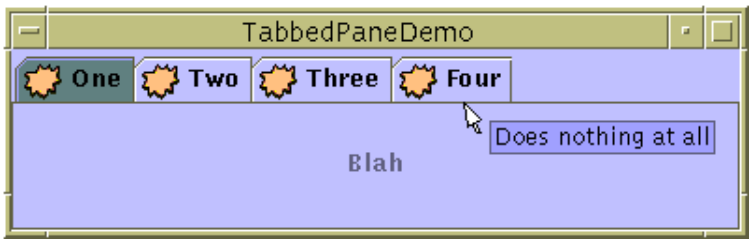


La portion visualisée à l'écran est contenue dans un objet nommé viewport de type `JViewPort` :



JTabbedPane : Permet de mettre plusieurs `JPanel` dans des onglets. La méthode pour ajouter des onglets est `add` :

```
JTabbedPane tabbedPane = new JTabbedPane();
ImageIcon icon = createImageIcon("images/middle.gif");
JComponent panel1 = makeTextPanel("Panel #1");
tabbedPane.addTab("One", icon, panel1, "Does nothing");
...
```



JToolBar : Permet de créer une barre de boutons. Les boutons sont des instances de `JButton`. Un ActionListener leur est attribué. Il suffit de les ajouter dans la toolbar à l'aide de la commande `add`.

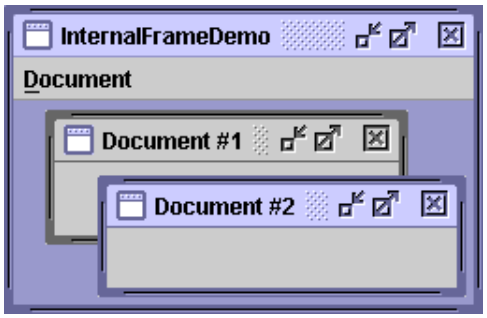
```
public class RecupereLesActions implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        ...
    }
}

JToolBar toolBar = new JToolBar();
JButton button = new JButton();
RecupereLesActions recupereLesActions = new RecupereLesActions();
button.addActionListener(recupereLesActions);
toolBar.add(button);
```

JSplitPane : Permet d'affichier deux JPanels séparés verticalement ou horizontalement.

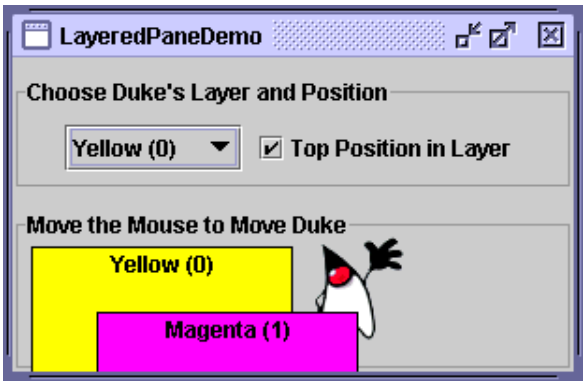
```
JSplitPane hSplitPane = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT,
                                       panel1,
                                       panel2);
```

2.3 Containers spécifiques

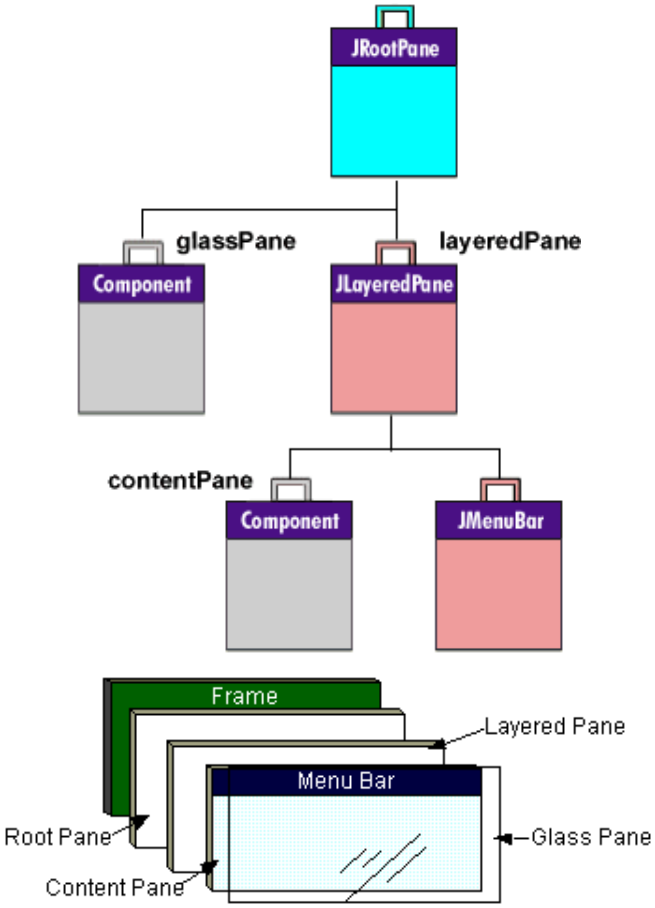
JInternalFrame : Permet d'affichier une fenêtre dans un JFrame.



JLayeredPane : Permet d'afficher des JPanels à différents niveaux.



JRootPane : C'est la structure à la base des composants Swing pouvant contenir des menus. Le JRootPane n'est pas l'objet dans lequel on ajoute des composants `rootPane. add(child);` est impossible, à la place on fera `rootPane. getContentPane(). add(child);`. Les JRootPane ne sont pas crées tout seuls, mais viennent en général avec des composants existants.



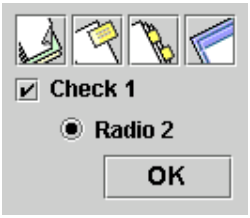
Ainsi par exemple pour spécifier la mise en page sur un rootPane :

```
rootPane.getContentPane().setLayout(new BoxLayout());
```

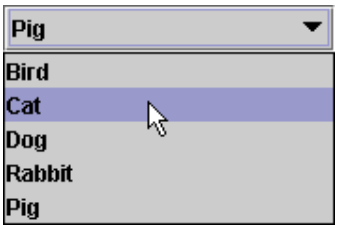
2.4 Contrôles de base

JButton : représente les boutons. Cette classe partage la classe abstraite AbstractButton avec entre autres : JCheckBox, JRadioButton, JMenuButton, JCheckBoxMenuItem, JRadioButtonMenuItem.

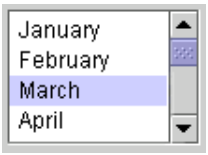
```
public class ButtonDemo extends JPanel
                        implements ActionListener {
    protected JButton b1, b2, b3;

    public ButtonDemo() {
        // ...

        b1 = new JButton("MonBouton");
        b1.setActionCommand("disable");
        b1.setActionListener(this);
        add(b1);
    }

    public void actionPerformed(ActionEvent e) {
        if ("disable".equals(e.getActionCommand())) {
            ...
        }
    }
}
```



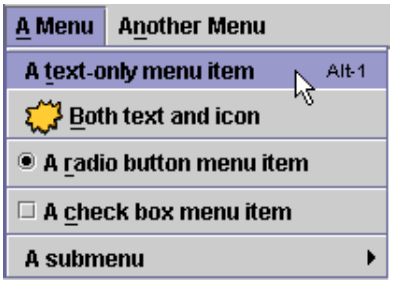
JComboBox : Permet de faire une liste dans laquelle l'utilisateur va pouvoir faire un seul choix. Il existe une forme éditable pour rajouter des items à la volée dans la liste.



JList : Permet d'afficher une liste d'objets, où l'utilisateur peut choisir plusieurs items.



JMenu : Permet de faire des menus. Les menus sont généralement ajoutés dans une barre de menus (`JMenuBar`) et comportent de items de type `JMenuItem`. L'exercice 1 détaille comment relier les menus aux actions à effectuer.



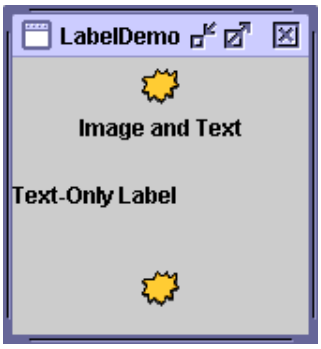
JSlider : Permet de rentrer une valeur numérique comprise entre deux bornes.

JSpinner : ressemble à une combo box mais sans partie dépliable qui peux recouvrir d'autres composants.

JTextField : permet de rentrer un champ texte.

2.5 Information non éditable

JLabel : permet d'afficher une chaîne de caractères.

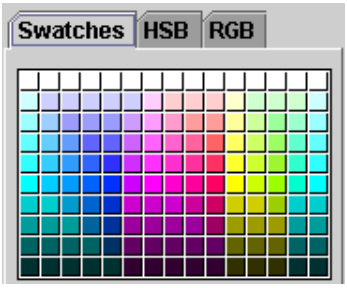


JProgressBar : Affiche une barre de progression.

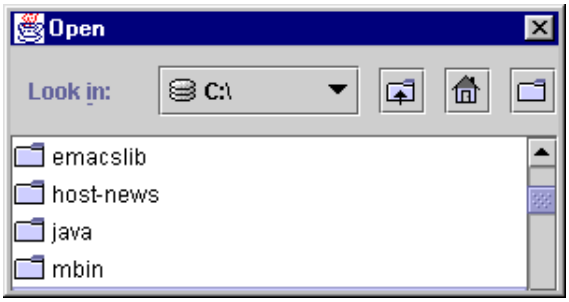
JToolTip : Affiche des indications sur un composant.

2.6 Représentations interactives

JColorChooser : Dialogue permettant de choisir une couleur.



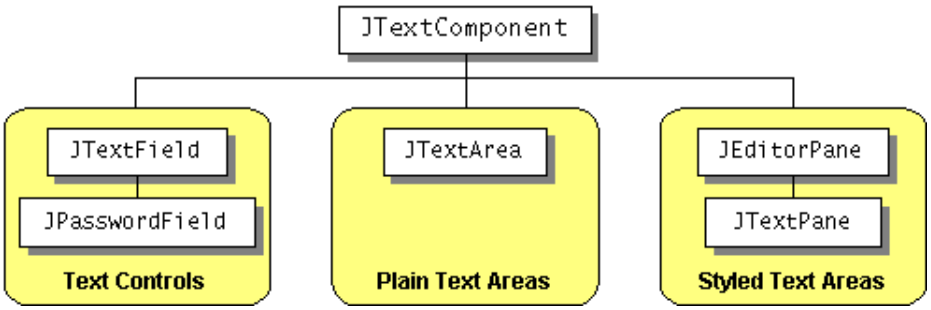
JFileChooser : Dialogue permettant de choisir un fichier. Une utilisation de cette classe est demandée dans l'exercice 1.



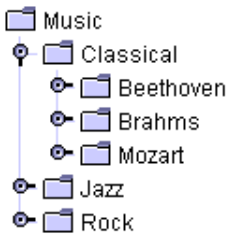
JTable : Permet d'afficher un tableau

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

JText :



JTree :



2.7 Présentation MVC

Plusieurs composants swing sont implémentés suivant l'architecture Model View Controler (MVC), c'est à dire que les fonctionnalités suivantes sont implémentées de manière séparées :

- le modèle : c'est à dire quelle est la logique de fonctionnement du composant (combien d'états, est-il activé ...),
- la vue : comment se composant est-il affiché, quel est son aspect,
- le contrôleur : en charge des actions utilisateur.

Les rôles view et controler sont souvent joués par les mêmes entités.
Le modèle s'obtient en général par la méthode `getModel` (par exemple dans `JTree`, `JSlider`, `JTabbedPane` ...).
Pour plus de détails :

```
http://java.sun.com/products/jfc/tsc/articles/architecture/
```

L'exercice 2 donne une illustration d'un usage fréquent de l'architecture MVC.

3 Evènements

3.1 Notion d'event listener

Pour pouvoir gérer les évènements qui se produisent il faut implémenter l'interface correspondant à l'évènement.

3.2 Exemple d'event listener

Ainsi par exemple pour être notifié des changements d'un `JTextField`, la classe `Toto` doit avoir le squelette suivant:

```
class Toto implements java.awt.event.ActionListener {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        // ...
    }
}
```

et pour enregistrer une instance de `Toto` auprès d'une instance de `JTextField` :

```
  JTextField text = new JTextField();
  Toto toto = new Toto();
  text.addActionListener(toto);
```

3.3 Autre exemple d'event listener

Ainsi par exemple pour être notifié des changements d'un `JSlider`, la classe `Toto` doit avoir le squelette suivant:

```
class Toto implements javax.swing.event.ChangeListener {
    public void stateChanged(javax.swing.event.ChangeEvent e) {
        // ...
    }
}
```

et pour enregistrer une instance de `Toto` auprès d'une instance de `JSlider` :

```
  JSlider slider = new JSlider(javax.swing.JSlider.HORIZONTAL,0,50,100);
  Toto toto = new Toto();
  slider.addChangeListener(toto);
```

3.4 Event Listener possibles

Voici une liste non exhaustive des évènements possibles:

- `ActionListener` se déclenche pour un bouton pressé, pour un item de menu choisi, pour un champ textuel modifié.
- `CaretListener` se déclenche quand dans un éditeur le curseur est bougé ou du texte est sélectionné.
- `ChangeListener` permet d'être notifié des changements dans un `JSlider`, un `JColorChooser` ou bien un `JSpinner`.
- `ItemListener` permet de savoir quand un item est sélectionné ou non, en particulier dans les `JCheckBox`, les `JComboBox` et les `JCheckBoxMenuItem`.
- `WindowListener` permet de récupérer auprès d'une fenêtre les évènements de d'ouverture, de fermeture, d'activation, de désactivation, d'iconification, et de desiconification.

Un tableau récapitulatif est présent à l'URL :

`http://java.sun.com/docs/books/tutorial/uiswing/events/eventsandcomponents.html`

4 Exercices

Exercice 3.1

Récupérer la classe `ex1.Drawing` à l'url :

`http://www.derepas.com/java/cours3_exercice_1.jar`

Il s'agit une petite classe (79 lignes) permettant de charger des images et de pouvoir dessiner dessus comme le programme `paint`. La classe vient sans menus ou outils pour choisir les couleurs, sauver les fichiers.

Q1 Le but de cette question est de rajouter un menu qui permet de sortir. Il faut rajouter un menu `Fichier` avec un item `Quitter` qui permet de fermer.

On va pour cela créer une classe nommée `DrawingPanel` qui "enrobe" la classe `Drawing`. On utilisera le squelette suivant :

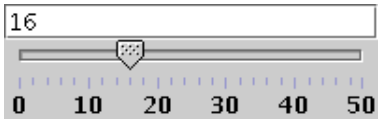
```
package ex1;

public class DrawingPanel extends javax.swing.JPanel
{
    Drawing drawing = null;

    class QuitAction extends javax.swing.AbstractAction {
        QuitAction() { super("quit"); }
        public void actionPerformed(java.awt.event.ActionEvent e) {
            quit();
        }
    }

    DrawingPanel() {
        // ...
    }

    public void quit() {
        System.exit(0);
    }

    public javax.swing.JMenuBar createMenuBar() {
        // ...
    }

    public static void main(String [] argv) {
        javax.swing.JFrame frame = new javax.swing.JFrame("Drawing");
        frame.setDefaultCloseOperation(javax.swing.JFrame.EXIT_ON_CLOSE);
        DrawingPanel d = new DrawingPanel();
        frame.setContentPane((javax.swing.JComponent)d);
        frame.setJMenuBar(d.createMenuBar());
        frame.pack();
        frame.setVisible(true);
    }
}
```

Indications : dans la méthode `createMenuBar` on créé une instance de `javax.swing.JMenuBar` dans laquelle on a ajouté une instance de `javax.swing.JMenu` dans laquelle on a ajouté une instance de `javax.swing.JMenuItem`. On

aura utilisé la méthde `addActionListener` sur l'instance de `javax.swing.JMenuItem` pour associer l'action correspondante.
Q2 Ajouter un item `ouvrir` dans le menu pour ouvrir un fichier. Indication : utiliser la classe `javax.swing.JFileChooser` pour choisir le fichier à ouvrir, appeler la méthode `loadImage` de la classe `ex1.Drawing` pour charger l'image.
Q3 Si cela n'a pas déjà été fait à la question 1, mettre l'instance de `ex1.Drawing` dans un `javax.swing.JScrollPane` pour pouvoir visualiser les images de taille importantes.

Exercice 3.2

L'idée de l'exercice est de montrer qu'il est facile de remplacer un modèle existant par un modèle ayant de nouvelles fonctionnalités.
Q1 Créer une classe `ex2.CustomSlider` qui permet de choisir un entier à l'aide d'une classe `JSlider` et affiche la valeur dans un `JTextField`, comme illustré ci-dessous.



Q2 On souhaite qu'une modification du `JTextField` permette de modifier la position du slider. Indication : utiliser la méthode `addActionListener` sur le `JTextField` pour récupérer les changements.
Q3 On souhaite maintenant que la partie modèle stocke un `double` et non un `int`.
Téléchargez la classe `CustomRangeModel` à l'url :

`http://www.derepas.com/java/CustomRangeModel.java`

On remarquera que `CustomRangeModel` implémente `BoundedRangeModel`. On peut donc s'en servir comme modèle pour n'importe quel `JSlider`.

Replacer le modèle du `JSlider` grâce au constructeur `JSlider(BoundedRangeModel brm)`. Adapter les méthodes précédemment écrites pour pouvoir rentrer des données de type `double` dans le `JTextField`.