INFO-H-200
Programmation orientée objet

Séance d'exercices 8
GUI Libraire javax.Swing

Université libre de Bruxelles
École polytechnique de Bruxelles

Professeur : Hugues Bersini

2015-2016

Swing

Swing est la libraire standard Java pour créer des interfaces graphiques (GUI).

Fait partie des Java Foundation Classes (JFC) qui contiennent d'autres outils utiles aux applications graphiques :

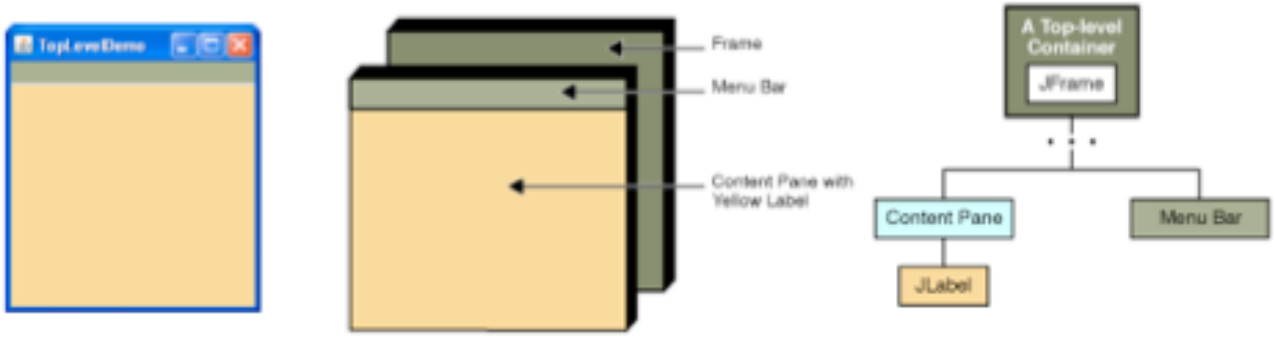
- Swing GUI Components : boutons, textes, menus, checkboxes, tableaux, boites de dialogues,...
- Java 2D API : Libraire d'affichage 2D sur laquelle se base Swing
- Synth : Librairie permettant de créer son propre "look and feel"
- Data transfert : Gestion du copier/coller, du drag and drop
- Internationalization : Gestion des différentes langues et caractères
- Accessibilité API, Undo Framework API,...

Conteneurs Swing

Un composant Swing (label, bouton,...) ne peut se trouver que dans un et un seul conteneur.

Un conteneur est un composant qui peut contenir d'autres composants. Si le conteneur contient des conteneurs, on parle alors de "containment hierarchy".

Une fenêtre (un Top-Level container) est un conteneur et est implémentée dans la classe JFrame. Une JFrame est composée d'un titre, d'un "Content Pane" et éventuellement d'un "Menu Bar".



Composants Swing

A l'exception des "Top-level" containers, toutes les classes Swing qui commencent par "J" descendent de la classe JComponent. (JPanel, JButton, JLabel,...)

JComponent descend de "Container" qui descend lui-même de "Component". Un conteneur est donc un composant.

Tous les "JComponent" ont des méthodes communes pour :

- Gérer l'apparence
- Gérer l'état
- Gérer des événements
- Dessiner les composants
- Placer les composants dans des conteneurs, spécifier leur position
- ...

Exemple : HelloWorld

```
package Example;

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

public class GUI {
   public static void main(String[] args){
      JFrame frame = new JFrame("HelloWorld");
            // Nouvelle fenêtre (top level container)
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      JLabel label = new JLabel("HelloWorld");
            // Nouveau label jaune contenant le texte "HelloWorld"
      label.setOpaque(true);
      label.setBackground(Color.YELLOW);

      frame.getContentPane().add(label);
                  // Ajout du label dans le ContentPane

      frame.pack();
                                                // Dimensionne la frame

      frame.setVisible(true);
                              // Affiche la fenêtre
   }
}
```

Quelques composants

- **JButton :** un bouton
- **JCheckBox :** un bouton à cocher
- **JRadioButton :** un bouton radio (doit être dans un ButtonGroup)
- **JLabel :** un label non éditable
- **JTextField :** un champ de texte
- **JTextArea :** une zone de texte (plusieurs lignes)
- **JDialog :** boîtes de dialoge (voir aussi JOptionPane)
- **JList :** liste
- **JComboBox :** liste déroulante
- **JProgressBar :** barre de progression
- **JTable, JTree :** tableaux et arbres
- ...

Panels

On peut grouper des composants dans des JPanel (un conteneur)

Un JPanel peut avoir un layout qui définira comment seront disposés les composants à l'intérieur du conteneur

Par défaut le layout est un FlowLayout, c'est à dire que les composants seront placés les uns à la suite des autres en fonction de la taille de la fenêtre

BorderLayout permet d'ajouter des composants dans cinq zones différentes :


```
JPanel p = new JPanel(new BorderLayout());
p.add(quelqueChose, BorderLayout.LINE_END);
```

Gestion des événements

Quand on intéragit avec un élément (p.e : un bouton), un événement est déclenché.

Un gestionnaire d'événement est donc nécessaire pour y réagir.

Solution simple : implémenter l'interface ActionListener et sa méthode actionPerformed(ActionEvent e)

Exemple : Programme avec un bouton déclenchant un son lorsqu'il est cliqué

```
public class GUI{
   JButton button;

   public GUI(){
   ...
      button.addActionListener(new BeepListener());
   ...
   }
}

public class BeepListener implements ActionListener{
   public void actionPerformed(ActionEvent e){
      ...
      // Make a beep sound...
   }
}
```

Gestion des événements (suite)

Ce système est simple et flexible

Exemple : Un programme pourrait créer un listener par source d'événement ou créer un seul listener pour toutes les sources ou encore avoir plus d'un listener pour un même événement d'une même source.

Chaque événement est représenté par un objet qui donne de l'information sur l'événement et sur sa source. Les sources sont souvent des composants mais ce n'est pas toujours le cas (ex : un message reçu sur un réseau)

Types d'événements

Les composants peuvent déclencher des événements différents. Le plus simple à utiliser et le plus courant est l'événement Action. Par contre, JColorChooser peut déclencher un événement de type Change qui est un peu différent (pour lequel il faudra implémenter ChangeListener. Idem pour la souris qui peut déclencher des événements Mouse ou MouseMotion.

Lors de l'utilisation d'un composant particulier, il est donc utile de consulter sa documentation (1) et celle de chaque événement (2) qu'il peut déclencher :

1. `http://docs.oracle.com/javase/tutorial/uiswing/components/`
2. `http://download.oracle.com/javase/tutorial/uiswing/events/handling.html`

Remarques

Un listener doit pouvoir s'exécuter très rapidement car la gestion des événements et du dessin d'une application s'exécutent dans le même thread. Un listener lent pourrait donc figer l'application le temps de son exécution.

Il est plus sûr d'implémenter des listeners dans des classes package-private

Un événement est un objet qui hérite de la classe EventObject, par exemple : MouseEvent

Object getSource() d'EventObject retourne l'objet qui a déclenché l'événement. Il existe des méthodes plus spécifiques comme getComponent(), implémentées dans les événements de type ComponentEvent.

Certaines interfaces listener contiennent plus d'une méthode. Par exemple, MouseListener contient 5 méthodes : mousePressed, mouseClicked,... Ce n'est pas une mauvaise pratique de laisser certaines méthodes vides.