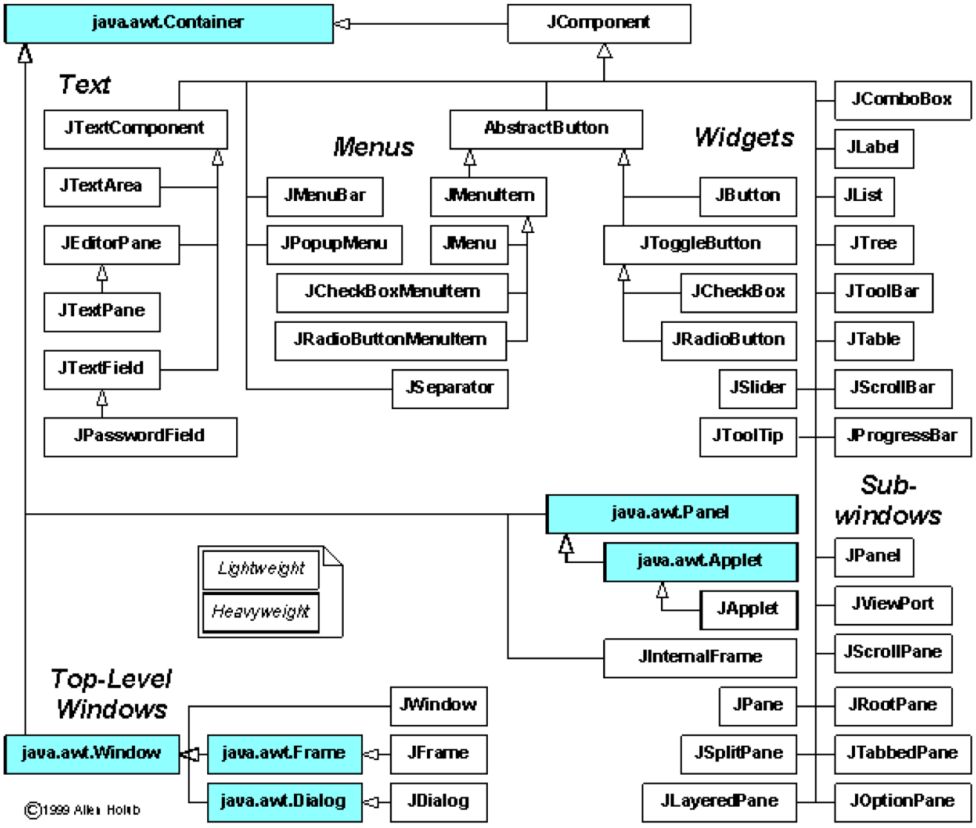
IFT1025 été 2010

Miklós Csűrös

25 juin 2010

10 Interface graphique

Deux packages : AWT (Java $\leq$ 1.1) et Swing.



10.1 Composants

Haut niveau. Affichage de fenêtres «indépendants» : `JFrame`, `JApplet`, `JDialog`

Niveau intermédiaire. Composition de fenêtres : `JPanel`, `JDesktopPane`, `JSplitPane`, `JTabbedPane`, `JScrollPane`.

Éléments. Widgets : `JLabel`,`JButton`, `JCheckBox`, `JTextArea`,....

10.2 JFrame

Règles pour un `JFrame` : (1) il faut spécifier sa position et taille, (2) composants sont ajoutés à son *content pane*, (3) il faut le rendre visible explicitement.

```
public void lancerGUI()
{
    JFrame frame = new JFrame(titre);
    // ajout d'éléments
    frame.setSize(longeur, hauteur); // ou frame.pack();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true); // affichage
}

public void static main(String[] args)
{
    ...
    javax.swing.SwingUtilities.invokeLater(new Runnable() {
            public void run() {lancerGUI();}
        }); // affichage dans Event Dispatch Thread
}
```

10.3 Panneaux intermédiaires.

JPanel. Le panneau le plus simple.

JTabbedPane. Onglets : **addTab**(`Component` composant, `String` titre).

JSplitPane. Panneau divisé en deux composants : **JSplitPane**(`int` orientation, `Component` composant1, `Component` composant2), avec orientation=`JSplitPane.HORIZONTAL_SPLIT`, `JSplitPane.VERTICAL_SPLIT`.

JDesktopPane. Un «petit» environnement de bureau en soi, avec sa propre gestion de fenêtres. Un fenêtre ici est de la classe `JInternalFrame`.

JScrollPane. Scrolling sur un composant qui est peut être trop grand pour afficher complètement : **JScrollPane**(`Component` vue).

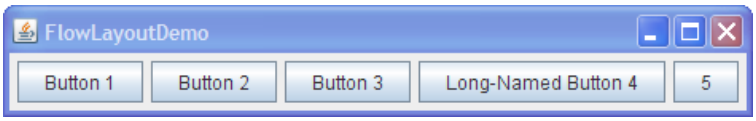
10.4 Gestion de placement

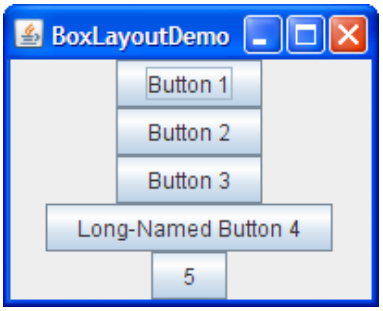
Interface `java.awt.LayoutManager` : calcule le placement optimal de composants dans un conteneur. Le `LayoutManager` utilise les méthodes **getPreferredSize**(), **getMinimumSize**() et **getMaximumSize**() des composants pour optimiser l'affichage. Lors de la création d'un élément intermédiaire, on y associe un `LayoutManager`. Typiquement, les éléments sont placés à l'aide de constantes du `LayoutManager` pour spécifier leur position.

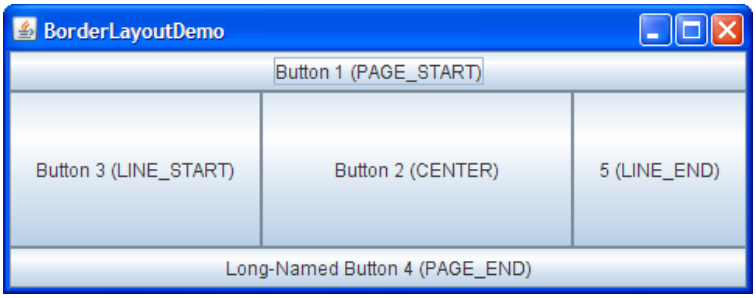
```
    JPanel panneau = new JPanel();
    panneau.setLayout(new BorderLayout());
    panneau.add(new JButton("Button 1"), BorderLayout.PAGE_START); // BorderLayout.NORTH
    ...
```

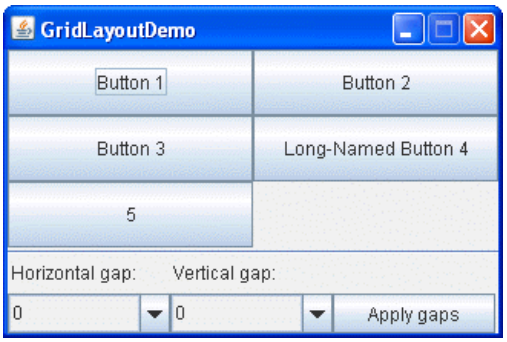
Un conteneur avec le placement calculé est *valide*. Pour valider un conteneur, Java fait la validation de tous ses composants, et utilise la taille et placement suggerés par les composants pour définir leur position dans le conteneur. On peut valider un conteneur par **validate**(). Si la taille d'un composant change, on peut le revalider par **revalidate()**. Tous les composants sont invalides lors de leur création : `pack()` de haut niveau fait la validation récursive de tous les membres de la fenêtre.

`JFrame` : content pane est avec `BorderLayout` par défaut ; `JPanel` : avec `FlowLayout` par défaut.

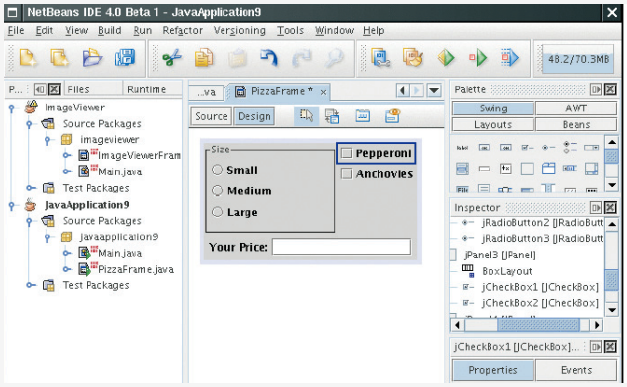






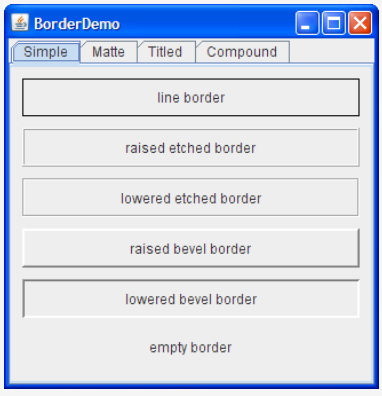


Pour un placement avancé, on utilise souvent des conteneurs imbriqués.



Ou bien, one peut utiliser un GUI builder.

10.5 Bordures



Les composants peuvent avoir des bordures : utiliser **setBorder**(`Border` border). Pour efficacité, il est recommandé d'utiliser le `BorderFactory` avec ses méthodes statiques : on peut réutiliser le même style de bordure sur plusieurs composants en même temps.

```
JPanel P = new JPanel();
P.setBorder(BorderFactory.createEtchedBorder());
Jpanel P2 = new JPanel();
P2.setBorder(BorderFactory.createEtchedBorder()); // même bordure
```

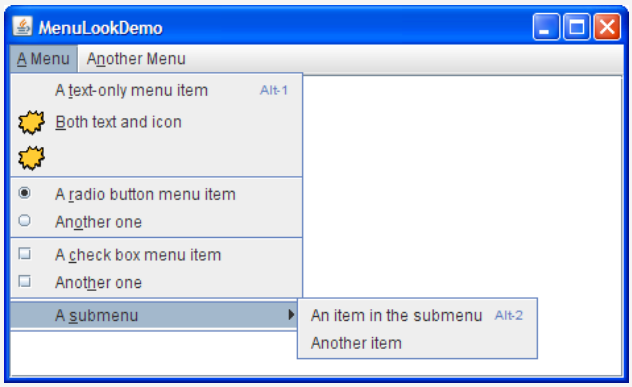
10.6 Choix

JCheckBox. Deux états : sélectionné ou non (`boolean` **getSelected**()). Événements : `ItemEvent` et `ItemListener`.

JRadioButton. Choix d'exclusion mutuelle : `JRadioButton` et `ButtonGroup`. États : séléctioné ou non (`boolean` **getSelected**() du bouton). Événements : `ItemEvent` et `ItemListener`.

```
JRadioButton bouton_oui = new JRadioButton("Oui");
JRadioButton bouton_non = new JRadioButton("Non");
JRadioButton bouton_neutre = new JRadioButton("Peut-etre");
ButtonGroup choix = new ButtonGroup();
choix.add(bouton_oui);
choix.add(bouton_non);
choix.add(bouton_neutre);
...
if (bouton_oui.isSelected()){ ... }
```

10.7 Menus



Le frame de haut niveau vient avec une barre de menu : classe `JMenuBar`. Un `JMenuBar` contient plusieurs objets de `JMenu`. Un `JMenu` contient d'autres objets `JMenu` et `JMenuItem`. Séléction d'un `JMenuItem` : `ActionEvent`.

```
JFrame frame;
...
JMenuBar menu_bar = new JMenuBar();
JMenu menu1 = new JMenu("A menu");
menu_bar.add(menu1);
JMenuItem item1 = new JMenuItem("A text-only menu item");
item1.setMnemonic(KeyEvent.VK_T);
item1.setAccelerator(
    KeyStroke.getKeyStroke(KeyEvent.VK_1,  // java.swing.KeyStroke
        Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()); // java.awt.Toolkit
item1.addActionListener(new ActionListener()
    { public void actionPerformed(ActionEvent e){ // menu choisi
    });
menu1.add(item1);
frame.setMenuBar(menu_bar)_;
```