INTERFACES GRAPHIQUES : SWING
(d'après "Swing, la synthèse", de V.Berthié et J-B Briaud)
==========

*** Introduction à la programmation événementielle en Java**
- IHM (Interface Homme-Machine) : la partie visible à l'écran d'une application
- programmation événementielle : basé sur des évenements (+ IG)
- nouveaux concepts :
  - avant : un programme commençait, faisait des calculs, ensuite il finissait (gcc, apt-get)
  - maintenant : le programme est toujours à l'attente d'un événement, et il fait quelque chose en fonction de l'événement (emacs, firefox, jeux, mathematica, openoffice.org)
  - en général c'est mixte (avec beaucoup plus de temps pour événements que pour calcul), ex. firefox : partie événementielle pour taper l'URL, partie calcul pour l'affichage
  - avant : affichage par du texte (System.out.println)
  - maintenant : affichage de composants graphiques
  - avant : exécuté dans un terminal, c'est lui qui gérait l'affichage, donc pas besoin de sauvegarder l'état de l'affichage
  - maintenant : nécessité de pouvoir peindre la fenêtre à tout moment (par ex. minimize->maximize), quand des parties masquées deviennent de nouveau visible
  - avant : chaque fonction était appelée dans le programme lui-même (appel explicite)
  - après : callback functions (appel implicite)

**** Exemple : affichage d'un texte**

```
import javax.swing.*;
public class Swing extends JFrame{
    public Swing (){
        setTitle ("Premier programme Swing");  // met le titre de la fenetre
        JLabel l = new JLabel ("Bonjour le monde !");  // cree texte
        add (l);  // ajoute le texte sur la fenetre
        setDefaultCloseOperation (EXIT_ON_CLOSE);  // ferme l'appli si X
        pack ();  // la taille de la fenetre est automatique
        setVisible (true);  // affiche la fenetre
    }
    public static void main (String[] args){
        new Swing ();  // cree fenetre
    }
}
```

**** Interfaces graphiques**
- beaucoup de bibliothèques d'IG : Java (AWT, Swing, SWT), gtk, qt, windows (MFC de Microsoft, VCL de Borland)
  - elles ont des "binding" dans plusieurs langages de programmation
- AWT - interface native
- Swing - interface gérée par Java, écrite en Java, créée par Sun
- nous allons utiliser que Swing (classes commençant par J pour composants)
  - Button (AWT) -> JButton (Swing)

*** Composants de base**

**** JLabel**
- affiche texte
- JLabel (String texte), setIcon (new ImageIcon ("icone.jpg"))
- setText (String texte), getText ()

**** JTextField**
- champ de saisie de texte (String !)
- JTextField (String texte), JTextField (int colonnes)
- setText (String texte), getText ()
- setEditable (boolean)

**** JButton**
- bouton simple, avec un texte et/ou une image
- JButton (String), JButton (String, Icon)
- setRolloverIcon (Icon), setPressedIcon (Icon)

- setMnemonic ('n') - ALT-n appuie sur le bouton

**** JToggleButton**
- bouton à deux états (enfoncé ou non)
- setIcon (non enfoncé), setSelectedIcon (enfoncé)
- setSelected (boolean), isSelected ()

**** JCheckBox, JRadioButton**
- cases à cocher, boutons radio
- JCheckBox cb1 = new JCheckBox ("Entree");

```
JCheckBox cb2 = new JCheckBox ("Plat");
JCheckBox cb3 = new JCheckBox ("Fromage");
JCheckBox cb4 = new JCheckBox ("Dessert");
```

- JRadioButton rb1 = new JRadioButton ("Allemagne");

```
JRadioButton rb2 = new JRadioButton ("France");
JRadioButton rb3 = new JRadioButton ("Espagne");
rb2.setSelected (true);
ButtonGroup g = new ButtonGroup ();
g.add (rb1);
g.add (rb2);
g.add (rb3);
```

**** JComboBox**
- liste déroulante pour choisir un item
- deux états : fermé ou ouvert
- new JComboBox (), .addItem (String)
- setEditable (boolean)

**** Autres**

http://download.oracle.com/javase/tutorial/uiswing/components/componentlist.html
- JProgressBar
  - une barre montrant la progression d'une tâche
  - JProgressBar (int orientation, int min, int max)
  - getValue (), setValue (int)
  - http://download.oracle.com/javase/tutorial/uiswing/components/progress.html
- JSlider
  - barre permettant de déplacer un curseur pour choisir une valeur
  - plus facile d'utilisation qu'un JTextField
  - http://download.oracle.com/javase/tutorial/uiswing/components/slider.html
  - JSlider (int orientation, int min, int max, int initialvalue)
  - setValue (int), getValue ()
  - espacement des "tics" de l'axe
- JSpinner
  - deux petites flèches permettant d'augmenter ou réduire une valeur
  - trois types : liste, nombres et dates
  - http://download.oracle.com/javase/tutorial/uiswing/components/spinner.html
- JTextArea
- etc.

*** Caractéristiques communes des composants de base**

**** Activation / désactivation**
- setEnabled (boolean), isEnabled ()

**** Visibilité**
- setVisible (boolean), isVisible ()
- exemple : version démo = options invisibles

**** Curseurs**
- l'apparence du curseur sur un composant
- setCursor (new Cursor(Cursor.WAIT_CURSOR));

**** Tooltips (bulles d'aide)**
- setToolTipText (String)

**** Couleurs**
- setBackground (Color.red)
- setForeground (Color.blue)

**** Menus**
- classes :
  - JMenuBar - top menu line
  - JMenu - top menu item
  - JMenuItem - menu item

```
    - JMenuItem.addSeparator () ou classe Separator - line menu item
  - JCheckBoxMenuItem - check box menu item
    - JCheckBoxMenuItem.setState (boolean)
- méthodes :
  - JFrame.setJMenuBar (JMenuBar)
  - JMenuBar.add (JMenu)
  - JMenu.add (JMenuItem)
  - JMenu.addSeparator ()
- exemple :
  JMenuBar barre = new JMenuBar ();
  setJMenuBar (barre);
  JMenu f = new JMenu ("Fichier");
  barre.add (f);
  JMenuItem i = new JMenuItem ("Nouveau");
  f.add (i);
  i.addActionListener (new Ecouteur());  // Ecouteur spécifie l'action
  i.setAccelerator (KeyStroke.getKeyStroke ('c', InputEvent.CTRL_MASK));
** Taille préférée
- setPreferredSize (new Dimension (largeur, hauteur));

* Containers et fenêtres
** JFrame, JDialog
- JFrame - fenêtre principale, avec décorations (barre de titre)
  - setDefaultCloseOperation (int flag) spécifie quoi faire si le
    JFrame est fermé
    - pour quitter l'application : setDefaultCloseOperation (EXIT_ON_CLOSE)
  - new JFrame ("Titre de la fenetre");
- JDialog - fenêtre secondaire, sur laquelle on ne travaille pas, mais
  qui affiche une information
  - peut être modale : empêche le travail sur les autres fenêtres tant
    qu'elle n'est pas fermée
  - JDialog jd = new JDialog ();
  - JDialog jd = new JDialog (JFrame, true);  // avec parent, true = modale
  - JOptionPane.showXYZDialog (...);
- setVisible (true/false)
- getRootPane().setDefaultButton (jb);
- spécifier la taille :
  - rigide : setSize (int largeur, int hauteur)
  - flexible : pack () - la taille de la fenêtre en fonction de la
    taille préférée de ses composants
- dispose () - fait disparaître et la mémoire occupée peut être
  libérée (ramasse-miettes)
** Containers
- objets contenant des composants
  - tout objet Swing est container, mais seuls quelques-uns sont utilisés
- JFrame, JDialog (fenêtres)
  - ajouter des composants : .add (...)
- JPanel - container minimal (vide)
  - .add (...)
- comment placer les composants à l'intérieur : layout managers (voir
  plus bas)
** JScrollPane
- normalement, si un composant déborde le container, il est invisible
  - il nous faut des scroll bars pour le container
- au lieu d'ajouter le container à la fenêtre, on ajoute un ScrollPane
  qui le contient :
  - JPanel container = new JPanel ();
  - add (new JScrollPane (container));

* Dessiner : classe Graphics
- classe abstraite fournissant des méthodes pour dessiner
  - lignes, rectangles, cercles, définir la couleur du crayon, la
    police (drawLine, drawOval, drawString etc.)
- origine d'un objet Graphics : haut-gauche
```



```
- en général, on dessine dans un JPanel
- pour dessiner, il faut redéfinir la méthode paintComponent d'une
  classe qui hérite par ex. JPanel, comme suit :
class MonJPanel extends JPanel {
  protected void paintComponent (Graphics g) {  // cet en-tete est obligatoire
    super.paintComponent(g);  // demande a la classe mere de se reafficher
    g.drawOval (0, 0, 50, 50);  // dessine un cercle
  }
}
...
MonJPanel jp = new MonJPanel ();
jp.setPreferredSize (new Dimension (300, 300));  // il faut preciser la taille
...

- paintComponent est une "callback function"
- au cas où on a besoin de forcer le réaffichage d'un composant :
  - jp.repaint ();

* Événements
- événements : tout ce qui est interaction avec l'utilisateur :
  - souris (clic...), clavier (appui touche...), focus, sélection d'un item...
- écouteur : méthode qui gère (est appelée pour) un ou plusieurs événements
- import java.awt.event.*;
** Exemple
  JButton jb = new JButton ("Bouton non appuye");
  jb.addActionListener (new Ecouteur ());
  ...
  class Ecouteur implements ActionListener{
    public void actionPerformed (ActionEvent e){
      jb.setText ("Bouton appuyé");
      // ou ((JButton)e.getSource()).setText ("Bouton appuyé");
    }
  }
** Exemples d'événements
- clic gauche sur un bouton - ActionEvent
- clic sur un JPanel - MouseEvent
- frappe d'une touche sur JPanel - KeyEvent
- passage du focus à un composant - FocusEvent
- iconification d'une fenêtre - WindowEvent
- sélection d'un item dans une JList - ListSelectionEvent
- redimensionnement, déplacement d'un composant - ComponentEvent
- d'autres
- pour trouver les événements qu'un composant peut générer, on regarde
  dans la documentation ses méthodes addXYZListener
- tous ces événements encapsulent le composant graphique (la source) et
  les coordonnées du clic, la touche frappée etc.
  - la source peut être obtenue avec XYZEvent.getSource ()
** Utilisation
- on écrit une classe (Ecouteur) qui implémente l'interface qui nous
  intéresse, e.g. ActionListener, WindowListener...
  - on écrit les méthodes demandées par l'interface
- on lie un objet de cette classe (l'écouteur) au composant qui est la
  source de l'événement :
  jb.addActionListener (new Ecouteur ());
- (un même composant peut avoir plusieurs écouteurs, qui seront
  appelés l'un après l'autre)
*** Classes internes
- peuvent accéder aux champs privés de la classe mère
- appropriées pour les classes écouteur, si elles ne sont pas réutilisées
- ExtClass.IntClass i = new ...()
** Adaptateurs
- exemple : pour fermer la fenêtre, on devrait implémenter
  WindowListener, mais elle a 7 méthodes :
  - windowOpened (WindowEvent e)
```

```
  - windowClosing
  - windowClosed
  - windowIconified
  - windowDeiconified
  - windowActivated
  - windowDeactivated
- ce qui nous intéresse est seule windowClosing
- pour éviter d'écrire toutes les 7 méthodes, alors qu'une seule nous
  intéresse, on utilise un "adaptateur", qui est une classe non
  abstraite avec des méthodes vides => on peut écrire que les méthodes
  qui nous intéressent
- WindowAdapter dans ce cas
public class Fermeture extends WindowAdapter{
  public void windowClosing (WindowEvent e){
    ...
  }
}
```

**** Terminologie**

- pour le "type" XYZ :
  - XYZEvent – la classe événement
  - addXYZListener – méthode du composant pour y ajouter un écouteur
  - XYZListener – interface avec les méthodes nécessaires à l'événement
    - les méthodes ont comme paramètre un XYZEvent
  - XYZAdapter (si elle existe) – classe non abstraite fournissant un corps vide à toutes les méthodes du XYZListener

*** Gestionnaires de placement (layout managers)**

- gère le placement des composants à l'intérieur d'un container
- non précis (et difficile à comprendre), car il doit traiter le placement pour toute taille du container
  - exemples de choses qui peuvent changer la taille des composants/fenêtres : thèmes, traductions, fontes, actions de l'utilisateur
- méthode setLayout (LayoutManager) du container
- import java.awt.*;

**** Exemple**

- mettre dans un container 6 boutons en 3 lignes et 2 colonnes :

```
setLayout (new GridLayout (3, 2));
add (new JButton ("1"));
add (new JButton ("2"));
add (new JButton ("3"));
add (new JButton ("4"));
add (new JButton ("5"));
add (new JButton ("6"));
```

**** BorderLayout**

- c'est le gestionnaire par défaut d'un JFrame
- divise le container en 5 parties : S, N, E, W, C

```
add (bouton, BorderLayout.SOUTH);
```

**** GridLayout**

- divise le container en rectangles de taille égale
- GridLayout (int nblignes, int nbcol)
- méthodes setColumns, setRows etc.

**** FlowLayout**

- le gestionnaire par défaut de JPanel
- place les composants du container l'un après l'autre, de gauche à droite et de haut en bas

**** Beaucoup d'autres**

- CardLayout : plusieurs composants au même endroit, un seul à la fois est affiché
- GridBagLayout : on précise l'endroit précis ou chaque composant se trouve à l'intérieur d'une grille
- SpringLayout : chaque composant précise la distance par rapport aux composants voisins
- etc.

**** Combinaison des gestionnaires**

- créer plusieurs JPanel, chacun avec le gestionnaire approprié

*** Applets**

- programmes téléchargeables sur Internet et exécutés par le navigateur
- téléchargés à partir d'une page Web (HTML)
- exécution :
  - on crée une page Web qui contient un lien vers l'applet
  - on charge la page dans un navigateur ou dans un "visualiseur d'applets" (appletviewer, fourni par Java SDK)

**** Page basique HTML**

- fichier a.html exécutant l'applet de MonApplet.java :

```
<HTML>
<BODY>
<APPLET code=MonApplet width=207 height=170></APPLET>
</BODY>
</HTML>
```

**** Particularités**

- différences par rapport à un programme classique basé sur un JFrame :
  - JFrame -> JApplet
  - main -> init
  - setTitle, setSize – interdites (car la taille est définie par la page Web)
  - sécurité (SecurityException) :
    - pas d'accès ou d'exécution des fichiers locaux
    - pas d'obtention d'informations système sensibles (nombre d'utilisateurs, la charge du système etc.)
- méthodes automatiquement appelées :
  - en-tête de ces méthodes : public void XYZ ()
  - init : après la création de l'objet applet
    - mettre ici le code d'initialisation de l'applet
  - destroy : fermeture du navigateur ou de l'appletviewer
  - stop : chaque fois que l'applet devient invisible, puis avant destroy
    - faire arrêter ici les gros calculs (s'il y en a)
  - start : après init, puis chaque fois que l'applet redevient visible
    - faire redémarrer ici les gros calculs (s'il y en a)

*** Divers**

- JFileChooser – classe pour choisir un fichier
- "look and feel" paramétrable

```
UIManager.setLookAndFeel(new MetalLookAndFeel());
UIManager.setLookAndFeel(new SynthLookAndFeel());
```

- Java beans
  - convention d'écriture d'une classe : set/getX, isX
  - tous les composants sont des beans
- plate-formes Java :
  - ME (Micro Edition)
  - SE (Standard Edition)
  - EE (Enterprise Edition)