Interfaces graphiques

responsable : Wiesław Zielonka
zielonka@liafa.univ-paris-diderot.fr
http://liafa.univ-paris-diderot.fr/~zielonka

January 19, 2016

Première application Swing

```
public class ActionTester
{
  public static void main(String[] args)
  {
    SwingUtilities.invokeLater( new Runnable(){
      ActionTester tester=new ActionTester();
      public void run(){
        tester.createGUI();
      }
     });
  }
  public void createGUI(){
    JFrame frame = new JFrame();
    frame.setSize(400,200);
    frame.setTitle("Hello");
//A faire
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
```

JFrame

– la fenêtre principale

```
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

D'autres possibilités :

- `WindowConstants.DISPOSE_ON_CLOSE`,
- `WindowConstants.DO_NOTHING_ON_CLOSE`
- `WindowConstants.HIDE_ON_CLOSE`.

Ajouter des widgets dans JFrame

```
final JTextField textField = new JTextField(20);
textField.setText("Click a button!");
final JButton helloButton = new JButton("Hello");
JButton goodbyeButton = new JButton("Say Goodbye");

/* On cree un JPanel pour les boutons */

JPanel butonPanel = new JPanel();
//on utilise GridLayout
butonPanel.setLayout(new GridLayout(1,2));
butonPanel.add( helloButton );
butonPanel.add( goodbyeButton );

JPanel panel = new JPanel();
/*FlowLayout Layout par defaut de JPanel*/
panel.add( butonPanel );
panel.add( textField );
/* mettre le panel a la place de ContentPane */
frame.setContentPane( panel );
```

Layout - gestionnaire de position.

- `GridLayout(int nbLignes, int nbCollones)`
- `FlowLayout()` – les widget un après l'autre

Ajouter les listener

```
ActionListener boutonListener = new ActionListener() {
  public void actionPerformed(ActionEvent event)
   {
        Object source = event.getSource();
        if( source == helloButton )
            textField.setText("Hello, World!");
        else
            textField.setText("Goodbye, World!");
   }
  };
/* Eregistrer le listener*/
helloButton.addActionListener( boutonListener );
goodbyeButton.addActionListener( boutonListener );
```

D'habitude les listeners différents pour chaque bouton. Evite `if` est plus conforme avec la programmation objet.

```
helloButton.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent event)
    {
        textField.setText("Hello, World!");
    }
});
```

BorderLayout

```
JPanel panel = new JPanel();
panel.setLayout(new BorderLayout());
panel.add(buttonA, BorderLayout.PAGE_START);
panel.add(buttonB, BorderLayout.LINE_START);
```

```
                PAGE_START
-----------------------------------------
LINE_START |      CENTER      | LINE_END
-----------------------------------------
                PAGE_END
```

Pour separer les éléments le constructeur avec 2 paramètres :

```
BorderLayout(int horizontalGap, int verticalGap)
```

Box - vertical ou horizontal

```
Box box = Box.createVerticalBox();
box.add(new JButton("Button 1"));
box.add(Box.createVerticalStrut(20));
box.add(new JButton("Button 2"));
box.add(new JButton("Button 3"));
box.add(new JButton("Long-Named Button 4"));
box.add(Box.createVerticalGlue());
box.add(new JButton("5"));
```

Le bord d'un composant

Utiliser les méthodes statiques de la class `BorderFactory` et la méthode `setBorder` :

```
composant.setBorder(
        BorderFactory.createTitledBorder("Titre de bord"));
```

Model Vue Controller

```
class Model {
    // Les donnees sont constituees d'un seul entier.
    private int value;              // Donnees du modele
    Model(int value) { this.value = value; }
    Model() { this(0); }
    void setValue(int value) { this.value = value; }
    int getValue() { return value; }
}
```

Controller

Implementé comme classe interne de Vue

```
class Controller {
  void notify(int value) {
    // Mise en place de la nouvelle valeur
    model.setValue(value);
    // Force la vue a se mettre a jour
    view.update();
  }
}// fin Controller
```

SliderView

```
class SliderView extends JPanel
      implements ChangeListener {
    // Ce booleen bloque l'envoie des evenements lorsque
    // ces evenements sont dus a une mise a jour des cur
    private boolean eventEnable = false;
    private JSlider lowByteSlider; // Poids faible
    private JSlider higByteSlider; // Poids fort
    // Construction d'un curseur pour un octet
    private JSlider getSlider() {
      // Construction du curseur
      JSlider slider = new JSlider(0, 255);
      // Affichage du 0 et du 255
      slider.setMajorTickSpacing(255);
      slider.setPaintTicks(true);
      slider.setPaintLabels(true);
      // Ecoute du curseur par la vue partielle
      slider.addChangeListener(this);
      return slider;
    }
```

SliderView - constructeur

```
SliderView() {
  setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
  // Curseur de l'octet de poids faible
  lowByteSlider = getSlider();
  // Bord avec un titre
  lowByteSlider.setBorder(
    BorderFactory.createTitledBorder("Octet de poids fa
  // Mise en place du curseur dans le panneau
  add(lowByteSlider);
  // Curseur de l'octet de poids fort
  higByteSlider = getSlider();
  // Bord avec un titre
  higByteSlider.setBorder(
    BorderFactory.createTitledBorder("Octet de poids fo
  // Mise en place du curseur dans le panneau
  add(higByteSlider);
  // Mise a jour de l'affichage
  update();
}
```

SliderView

```
// Listener des deux curseurs
public void stateChanged(ChangeEvent event) {
  if (eventEnable) {
    // La valeur calculee a partir des deux curseurs
    // est envoyee au controleur
    controller.notify(lowByteSlider.getValue() +
                      higByteSlider.getValue()*256
     );
  }
}
```

SliderView

```
// Mise a jour des deux curseurs
void update() {
  // Valeur a visualiser
  int value = model.getValue();
  eventEnable = false; // Bloquage des evenements
  lowByteSlider.setValue(value%256); // Poids faible
  higByteSlider.setValue(value/256); // Poids fort
  eventEnable = true;  // Debloquage des evenements
 }
}//fin SlideViewer
```