

Java Swing

Swing - introduction

- Toolkit Java pour la construction d'interfaces graphiques
 - GUI : Graphic User Interface
 - IHM : Interface Homme Machine
- Permet de construire
 - la partie présentation d'une application autonome
 - le client lourd d'une application distribuée (Java EE)
 - communication par RMI (Remote Method Invocation) ou Webservice

Swing - introduction

- Swing constitue une riche bibliothèque de composants graphiques (widgets)
 - widgets de base : boutons, labels, ascenseurs, ...
 - widgets avancée : arbres, tables, ...
- Swing fait partie des JFC (Java Foundation Classes)
 - collection de packages permettant la création d'applications bureautiques
 - JFC est constitué de AWT, Swing, Accessibility, Java 2D et Drag and Drop

Swing et Eclipse

- Par défaut Eclipse n'est pas livré avec un éditeur visuel
 - il y a toujours la possibilité de construire les interfaces graphiques "à la main"
 - un éditeur permet de s'affranchir d'un bon nombres de tâches, et permet un placement plus précis des widgets
 - outils d'alignement, de redimensionnement, ...
 - présentation des propriétés
- Plugins
 - VE (Visual Editor) - Eclipse
 - Window Builder - Google

Quelques tests

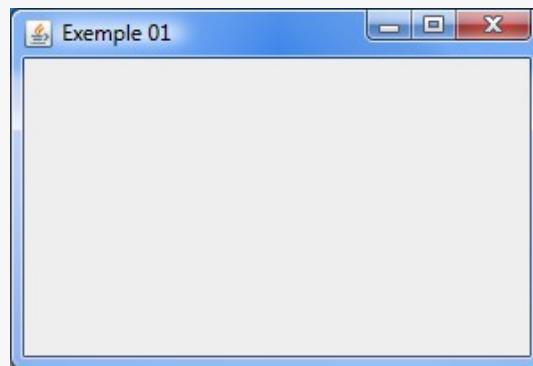
- Ce premier exemple affiche une fenêtre

Introduction_01

```
public class Fenetre01 extends JFrame {  
    public Fenetre01() {  
        setTitle("Exemple 01");  
        setSize(300, 200);  
        setLocationRelativeTo(null);  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
    }  
    public static void main(String[] args) {  
        Fenetre01 fen = new Fenetre01();  
        fen.setVisible(true);  
    }  
}
```

centre la fenêtre

provoque l'arrêt
de l'application



Quelques tests

- Ajout d'un bouton de fermeture

Introduction_02

```
public class Fenetre02 extends JFrame {  
    public Fenetre02() {  
        init();  
    }  
    public static void main(String[] args) {  
        Fenetre02 fen = new Fenetre02();  
        fen.setVisible(true);  
    }  
    private void init() {  
        JPanel panel = new JPanel();  
        getContentPane().add(panel);  
        panel.setLayout(null);  
        JButton quitButton = new JButton("Quit");  
        quitButton.setBounds(50, 60, 80, 30);  
        quitButton.addActionListener(new ActionListener() {  
            public void actionPerformed(ActionEvent event) {  
                System.exit(0);  
            }  
        });  
        panel.add(quitButton);  
        setTitle("Exemple 02");  
        setSize(300, 200);  
        setLocationRelativeTo(null);  
        setDefaultCloseOperation(EXIT_ON_CLOSE);  
    }  
}
```

ajout d'un conteneur de widgets

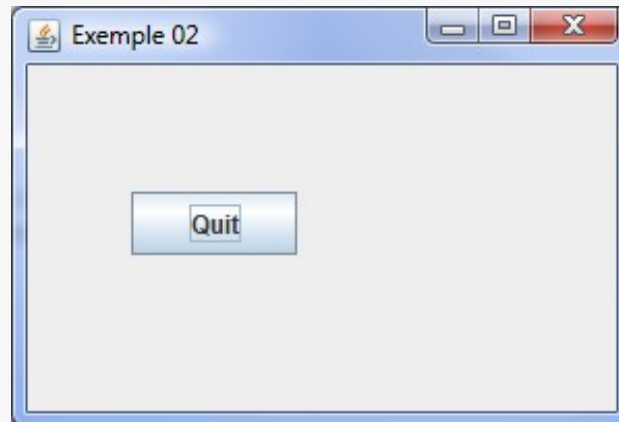
création d'un bouton avec taille et position

ajout d'un listener sur le bouton

ajout du bouton au conteneur

Quelques tests

- C'est une bonne pratique de créer une méthode `init()` qui initialise le composant graphique
 - par la suite seuls des extraits de code seront présentés
- Affichage de la fenêtre avec un bouton de fermeture



Quelques tests

- Ajout d'un tooltip au bouton
 - le survol du code affiche le tooltip
 - extrait de code

Introduction_03

```
...
JButton quitButton = new JButton("Quit");
quitButton.setBounds(50, 60, 80, 30);
quitButton.setToolTipText("Bouton de fermeture");
quitButton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent event) {
        System.exit(0);
    }
});
panel.add(quitButton);
...
```

ajout du tooltip

Widgets de base

- Les exemples survolent les possibilités d'utilisation
 - il est impératif de parcourir la documentation Java pour connaître les modes d'utilisation des composants
 - les constructeurs
 - les méthodes
 - les méthodes héritées
 - les exemples présentent aussi d'autres concepts
 - les gestionnaires de positionnement (layout)
 - utilisation des couleurs
 - utilisation des polices de caractères
 - etc.

JLabel

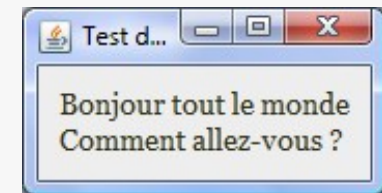
- Le widget `JLabel` permet l'affichage de texte
 - ne réagit pas aux événements en entrée
 - normalement utilisé pour l'affichage d'un message court
 - permet aussi l'affichage de d'images
 - accepte les tags HTML

JLabel

JLabelTest

création du JLabel

```
private void init(){
    setTitle("Test de JLabel");
    String contenuLabel = "<html>Bonjour tout le monde<br />"
        + "Comment allez-vous ?</html>";
    JPanel panel = new JPanel();
    panel.setLayout(new BorderLayout(10, 10));
    JLabel label = new JLabel(contenuLabel);
    label.setFont(new Font("Georgia", Font.PLAIN, 14));
    label.setForeground(new Color(50, 50, 25));
    panel.add(label, BorderLayout.CENTER);
    panel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));
    add(panel);
    pack();
    toolkit = getToolkit();
    Dimension screensize = toolkit.getScreenSize();
    setLocation((screensize.width - getWidth()) / 2,
        (screensize.height - getHeight()) / 2);
    setDefaultCloseOperation(EXIT_ON_CLOSE);
}
```

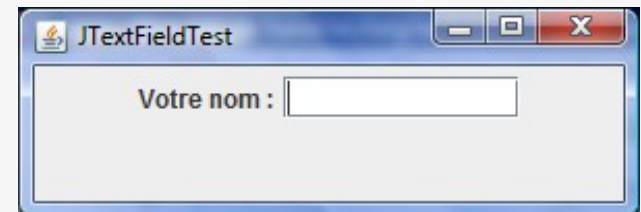


JTextField

- `JTextField` permet la saisie d'une ligne de texte
 - voir aussi :
 - `JFormattedTextField`
 - `JPasswordField`

JTextFieldTest

```
private void init(){  
    JPanel panel = new JPanel();  
    getContentPane().add(panel);  
    JLabel label = new JLabel("Votre nom :");  
    JTextField textField = new JTextField(10);  
    panel.add(label);  
    panel.add(textField);  
    setTitle("JTextFieldTest");  
    setSize(300, 100);  
    setDefaultCloseOperation(EXIT_ON_CLOSE);  
}
```

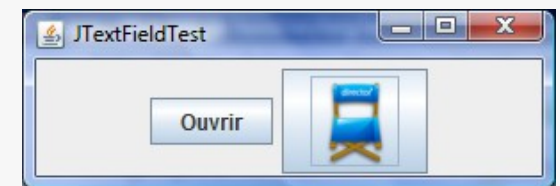


JButton

- JButton peut être construit avec du texte et/ou une image
 - est systématiquement relié à une action utilisateur

JButtonFieldTest

```
private void init() {  
    JPanel panel = new JPanel();  
    getContentPane().add(panel);  
    JButton button1 = new JButton("Ouvrir");  
    JButton button2 = new JButton();  
    button2.setIcon(createImageIcon("/images/cinema5_48.png"));  
    panel.add(button1);  
    panel.add(button2);  
    setTitle("JTextFieldTest");  
    setSize(300, 100);  
    setDefaultCloseOperation(EXIT_ON_CLOSE);  
}
```



JCheckBox

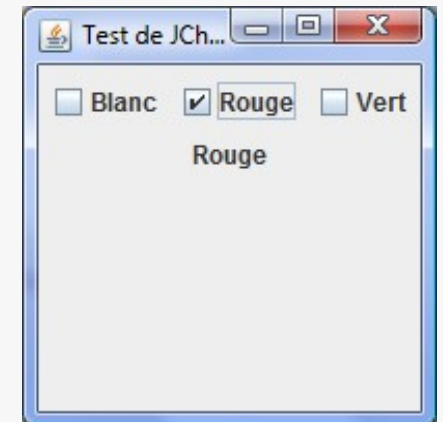
- Le JCheckBox possède deux états : vrai ou faux
 - sélectionné ou non

JCheckBoxTest

```
private void init() {  
    ...  
    JCheckBox cbBlanc = new JCheckBox("Blanc", true);  
    cbBlanc.addActionListener(this);  
    cbBlanc.setSelected(false);  
    panel.add(cbBlanc);  
  
    JCheckBox cbRouge = new JCheckBox("Rouge", true);  
    cbRouge.addActionListener(this);  
    cbRouge.setSelected(false);  
    panel.add(cbRouge);  
  
    JCheckBox cbVert = new JCheckBox("Vert", true);  
    cbVert.addActionListener(this);  
    cbVert.setSelected(false);  
    panel.add(cbVert);  
  
    label = new JLabel();  
    panel.add(label);  
    ...  
}  
public void actionPerformed(ActionEvent e) {  
    JCheckBox source = (JCheckBox) e.getSource();  
    label.setText(source.getText());  
}
```

ajout du listener, la classe implémente
ActionListener

méthode invoquée par
la sélection



JRadioButton

- Les boutons radios doivent être regroupés dans un conteneur spécifique (`ButtonGroup`) pour avoir le comportement exclusif

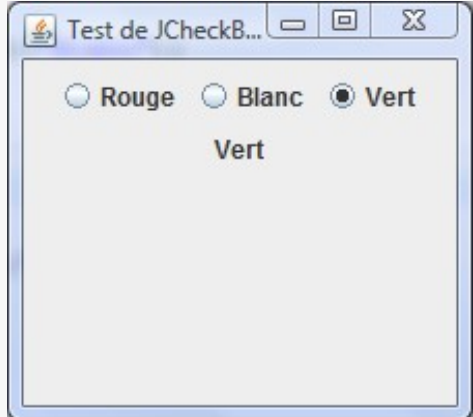
JRadioButtonTest

conteneur pour les boutons radio

```
...
JPanel panel = new JPanel();
getContentPane().add(panel);
ButtonGroup group = new ButtonGroup();

JRadioButton rdRouge = new JRadioButton("Rouge");
rdRouge.addActionListener(this);
panel.add(rdRouge);
group.add(rdRouge);
rdRouge.setSelected(true);
...
```

sélectionné par défaut

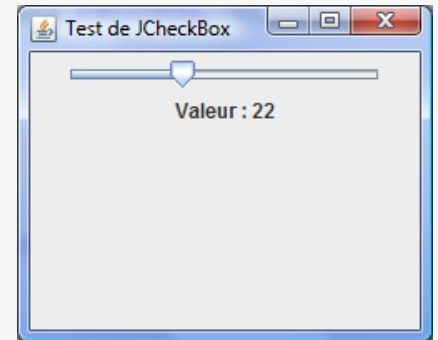


JSlider

- Ce composant permet la sélection graphique d'une valeur
 - contrôle de volume par exemple

```
private void init() {  
    setTitle("Test de JCheckBox");  
    JPanel panel = new JPanel();  
    getContentPane().add(panel);  
    JSlider slider = new JSlider(SwingConstants.HORIZONTAL, -50, +150, 0);  
    panel.add(slider);  
    slider.addChangeListener(this);  
  
    label = new JLabel();  
    panel.add(label);  
    setSize(250, 200);  
    setLocationRelativeTo(null);  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
public void stateChanged(ChangeEvent e) {  
    JSlider source = (JSlider) e.getSource();  
    label.setText("Valeur : "+source.getValue());  
}
```

JSliderTest



construction avec
orientation, valeurs
minimales, maximales et
par défaut

la classe implémente
ChangeListener

JComboBox

- Permet de combiner une zone d'édition avec une liste déroulante
- le comportement éditable est paramétrable

```
private void init() {  
    setTitle("Test de JCheckBox");  
    JPanel panel = new JPanel();  
    getContentPane().add(panel);  
    JComboBox combo = new JComboBox(couleurs);  
    combo.addItemListener(this);  
    panel.add(combo);  
  
    label = new JLabel();  
    panel.add(label);  
    setSize(100, 200);  
    setLocationRelativeTo(null);  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
public void itemStateChanged(ItemEvent e) {  
    JComboBox source = (JComboBox) e.getSource();  
    label.setText(source.getSelectedIndex() + " " + source.getSelectedItem());  
}
```

JComboBoxTest

constructeur acceptant
un String[]

la classe implémente
ItemListener



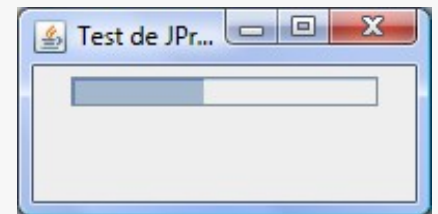
JProgressBar

- Permet de visualiser un traitement long
 - la valeur de ce composant évolue de 0 à 100

JProgressBarTest

```
public static void main(String[] args) {  
    JProgressBarTest test = new JProgressBarTest();  
    test.setVisible(true);  
    for(int i=1 ; i<101 ; i++)  
    {  
        try {  
            Thread.sleep(100);  
            bar.setValue(i);  
        } catch (InterruptedException e) {  
            e.printStackTrace();  
        }  
    }  
}  
  
private void init() {  
    setTitle("Test de JProgressBar");  
    JPanel panel = new JPanel();  
    getContentPane().add(panel);  
    bar = new JProgressBar();  
    panel.add(bar);  
    setSize(200, 100);  
    setLocationRelativeTo(null);  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}
```

mise à jour de la barre
de progression



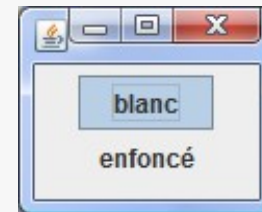
JToggleButton

- Bouton possédant deux états : enfoncé ou non

JToggleButtonTest

```
private void init() {  
    setTitle("Test de JToggleButton");  
    JPanel panel = new JPanel();  
    getContentPane().add(panel);  
    ButtonGroup group = new ButtonGroup();  
  
    JToggleButton tbBlanc = new JToggleButton("blanc");  
    tbBlanc.addActionListener(this);  
    panel.add(tbBlanc);  
  
    label = new JLabel();  
    panel.add(label);  
    setSize(100, 200);  
    setLocationRelativeTo(null);  
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
}  
  
public void actionPerformed(ActionEvent e) {  
    JToggleButton source = (JToggleButton) e.getSource();  
    if(source.isSelected())  
        label.setText("enfoncé");  
    else  
        label.setText("apparent");  
}
```

la classe implémente
ActionListener



Swing

- Ce chapitre n'est qu'une présentation de Swing
- Pour aller plus loin, il est nécessaire de voir
 - le modèle événementiel
 - les gestionnaires de positionnement (`LayoutManager`)
 - les boîtes de dialogue
 - les menus et barres d'outils
 - le modèle MVC
 - l'intégration au système d'exploitation ...