Interface graphique

Pour commencer ¶ §

Packages

- `java.awt`: première bibliothèque. Fournit des interfaces et des composants limités en code natif.
- `javax.swing`: seconde bibliothèque. Ne remplace pas `java.awt`, mais la complète avec des composants originaux et/ou performants, écrits en Java.
- On va utiliser `javax.swing` en priorité (composants commençant par `J`), mais aussi `java.awt` (*layout managers*, événements).

Hello World

```
import javax.swing.*;
public class HelloWorldSwing implements Runnable {
    @Override
    public void run() {
        //Create the window
        JFrame f = new JFrame("HelloWorldSwing");
        //Set the behavior for when the window is closed
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        //Add a label
        f.getContentPane().add(new JLabel("Hello world!"));
        //Set the window size from its components size
        f.pack();
        //By default, the window is not visible; make it visible
        f.setVisible(true);
    }
    public static void main(String[] args) {
        //Run the application at the correct time in the event queue
        SwingUtilities.invokeLater( new HelloWorldSwing() );
    }
}
```

Event-Dispatching Thread

- Au lancement d'une application graphique, en plus du thread principal, est lancé un second thread appelé *Event-Dispatching Thread* (EDT).
- Une application graphique est réactive aux évenements (clic sur un bouton par exemple). Or les réactions (affichage d'un commentaire par exemple) sont placées dans une file et exécutées les unes après les autres dans l'EDT.
- Les composants graphiques sont *thread unsafe*. C'est une erreur fréquente de les manipuler depuis plusieurs threads. Les autres threads doivent soumettre leur code à exécuter à l'EDT pour éviter les incohérences.

Structure d'une fenêtre

Hiérarchie de conteneurs

Le nom des conteneurs de `javax.swing` commencent par `J`.

Chaque application graphique possède au moins un conteneur de haut-niveau:

- `JFrame` (fenêtre)
- `JDialog` (boîte de dialogue)
- `JApplet` (fenêtre à intégrer dans une page web)

Ces conteneurs possèdent un *panneau d'affichage* (accessible par `getContentPane()`, `setContentPane()`) sur lequel vous pouvez disposer des composants (qui dérivent de `JComponent`).

Composants graphiques

Il existe de nombreux composants graphiques:

- `JPanel` (conteneur générique léger)
- `JLabel` (étiquette de texte)
- `JButton` (bouton)
- `JTextField` (champs texte pouvant être éditable)
- `JMenuBar` (barre des menus)
- ...

Consultez l'API standard pour la liste des composants disponibles dans `javax.swing`.

Personnaliser le panneau d'affichage

Habituellement, on crée un nouveau panneau d'affichage de zéro en utilisant `JPanel`, puis on l'ajoute au conteneur de haut niveau (typiquement de type `JFrame`) avec `setContentPane()`.

```
//Create a panel and add components to it.
JPanel contentPane = new JPanel();
contentPane.add(someComponent);
contentPane.add(anotherComponent);

topLevelContainer.setContentPane(contentPane);
```

Layout Manager

Il existe des modèles pour disposer les composants graphiques sur un panneau d'affichage:

- `java.awt.BorderLayout` (répartition en 5 zones: haut, bas, gauche, droite, centre)

- `java.awt.FlowLayout` (place les composants de gauche à droite, ligne par ligne)
- `java.awt.GridLayout` (répartition en un tableau 2d régulier)
- `javax.swing.BoxLayout` (place les composants les uns au-dessous des autres)
- ...

Consultez la page des layout managers pour avoir un aperçu de ces modèles.

Utiliser un Layout Manager

Le modèle par défaut est `FlowLayout`. Mais il est très simple d'en utiliser un autre en le passant en paramètre, soit au constructeur de `JPanel`, soit à la méthode `setLayout()`.

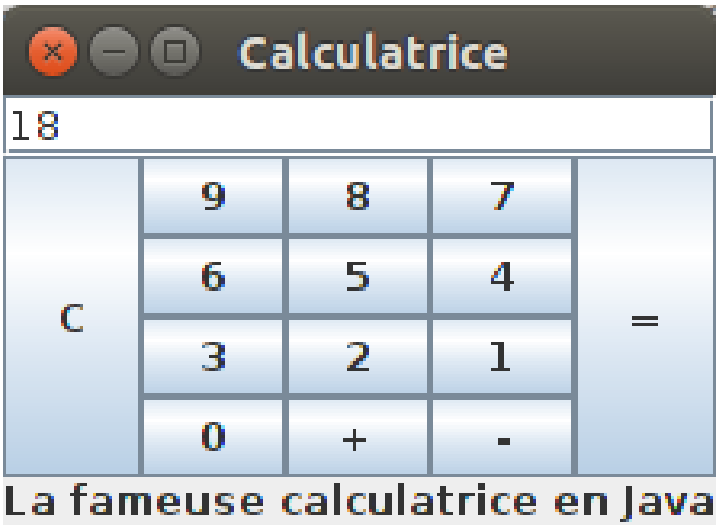
```
JPanel contentPane = new JPanel(new BorderLayout());
//or:
//JPanel contentPane = new JPanel();
//contentPane.setLayout( new BorderLayout() );

contentPane.add(someComponent, BorderLayout.CENTER);
contentPane.add(anotherComponent, BorderLayout.PAGE_END);

topLevelContainer.setContentPane(contentPane);
```

Ex.1. Calculatrice/Structure (20 min)

Reproduisez la fenêtre suivante:



Ex.1. Détails

- Le panneau d'affichage principal utilise un `BorderLayout`.
- En haut (PAGE_START), se trouve un `JTextField`.
- En bas (PAGE_END), se trouve un `JLabel`.
- A gauche (LINE_START), se trouve un `JButton` pour remettre la calculatrice à zéro.
- A droite (LINE_END), se trouve un `JButton` pour afficher le résultat.
- Au centre (CENTER), se trouve un panneau secondaire utilisant un `GridLayout` contenant

des `JButton` pour les chiffres et les opérations.
- Pensez à garder des références vers tous les composants.

Gestion des evénements

Fonctionnement général

- A un composant graphique (`JButton`), on attache un objet capable d'écouter les événements (`ActionListener`).
- Quand l'utilisateur réalise une action sur le composant (clic), un événement est généré (`ActionEvent`).
- La machine virtuelle reçoit tous les événements, mais seul les événements écoutés déclenchent une réaction (dont le code se trouve dans la méthode `actionPerformed()` de l'interface `ActionListener`).

Exemple de Listener

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

/*
 * Button listener
 */
public class ButtonBeeper implements ActionListener {

    @Override
    public void actionPerformed(ActionEvent e) {
        //print 'beep' to the standard output
        //when an action event is received
        System.out.println("beep");
    }

}
```

Exemple d'application

```
import javax.swing.*;
public class BeeperApp implements Runnable {
    @Override
    public void run() {
        //Create the window.
        JFrame f = new JFrame("Beeper");
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        // Add a button with an action listener
        JButton b = new JButton("Click me");
        b.addActionListener( new ButtonBeeper() );
        f.getContentPane().add(b);
        //Display the window.
        f.pack();
        f.setVisible(true);
```

```
    }
    public static void main(String[] args) {
        //Run the application at the correct time in the event queue.
        SwingUtilities.invokeLater( new BeeperApp() );
    }
}
```

Quelques types d'évenement

- `ActionEvent` / `ActionListener` (déclenchés par `JButton`)
- `ItemEvent` / `ItemListener` (déclenchés par `JCheckBox`)
- `MouseEvent` / `MouseListener` (déclenchés par la souris sur un composant)
- `KeyEvent` / `KeyListener` (touche clavier)
- ...

Consultez l'API standard pour la liste des événements disponibles dans `java.awt.event`, le tutoriel Evénements, ainsi que les How To consacrés aux composants pour savoir quel type d'événement ils déclenchent.

ActionEvent / ActionListener

Le couple `ActionEvent` / `ActionListener` est le plus commun.

- `ActionEvent` est la classe des événements déclenchés par une action bien définie sur un composant. Elle possède notamment la méthode `getSource()` pour obtenir le composant ayant déclenché l'événement.
- Un objet issu d'une classe qui implémente l'interface `ActionListener`, doté de la méthode `actionPerformed()`, est attaché à un tel composant par la méthode `addActionListener()`.

Ex.2. Calculatrice/Evenements (30 min)

- Faites en sorte que votre calculatrice affiche le résultat d'une suite d'additions ou de soustractions données par des actions sur les différents boutons.
- Téléchargez la classe **`Processeur`** qui se charge des calculs.
- Ecrivez un *listener* par type de bouton (un pour les chiffres, le plus, le moins, le `=` et le `C`); chacun connaissant une instance de `Processeur` (calcul) et une instance de `JTextField` (affichage).
- Astuce: les composants `JButton` et `JTextField` possèdent tous les deux les méthodes `getText()` et `setText()` pour respectivement lire et écrire le texte attaché au composant.

Ce qu'il faut retenir

- Une fenêtre (`JFrame`) est composée d'une hiérarchie de conteneurs (`JPanel`), contenant eux-même des composants graphiques élémentaires (`JLabel`, `JButton`, etc.).
- Des modèles (*layout managers*) permettent de disposer les composants sur un panneau d'affichage.

- Les composants graphiques peuvent s'abonner à des *listeners* (`ActionListener`), écoutant des événements particuliers. Lorsqu'un énénement écouté (`ActionEvent`) est déclenché, le code associé au *listeners* est exécutée par le *Event-Dispatching Thread*.