IFT1025 été 2010

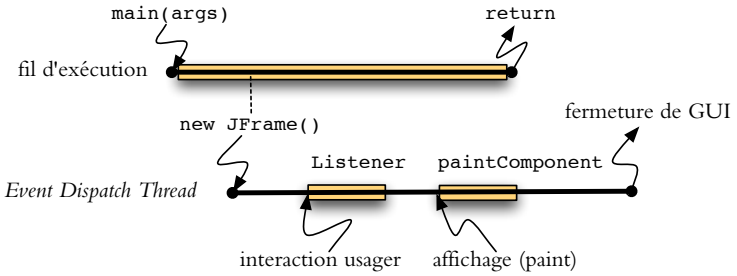
Miklós Csűrös

8 juin 2010

5 Événements en Swing

5.1 Event Dispatch Thread

L'interface graphique fonctionne à l'aide d'un procéssus légér (thread) dédié qui s'appelle l'*Event Dispatch Thread* (**EDT**).



L'interaction usager génère des **événements**. Chaque composant graphique vient avec son répertoire d'événements associés. Pour réagir aux événements, il faut y attacher des écouteurs (*listeners*) appropriés. Le code d'un écouteur s'exécute sur l'EDT. Swing gère l'affichage des composants graphiques en appellant la méthode `paintComponent` dans l'EDT.

Pour vérifier si on est dans l'EDT, utiliser `javax.swing.SwingUtiliities.isEventDispatchThread()`.

5.2 Événéments et écouteurs

Un événement en Java est representé par un objet de type `EventObject`. Des sous-classes prédéfinies existent dans les packages `java.awt.event` et `javax.swing.event`. Chaque événement vient d'une source (`getSource()`). Le code du GUI est composé de

⋆ initialisation de composants graphiques, et de leurs relations

⋆ pièces de code pour les écouteurs qui réagissent aux événements intéressants

Il est très important d'assurer qu'on répond toujours à l'interaction usager — exécution du code dans l'écouteur doit prendre très peu de temps !

source	type d'événement	écouteur	méthode(s)
`JButton`, `JMenuItem`, `Timer`	`ActionEvent` (AWT)	`ActionListener`	`actionPerformed`
souris	`MouseEvent` (AWT)	`MouseListener`	`mouseClicked`, `mousePressed`, `mouseReleased`, `mouseEntered`, `mouseExited`
souris	`MouseMotionEvent` (AWT)	`MouseMotionListener`	`mouseDragged`, `mouseMoved`

Exemple (écouteur pour bouton)

```
    JButton bouton = new JButton("OK");
    bouton.addActionListener(
        new ActionListener() // classe anonyme
            {
                public void actionPerformed(ActionEvent e)
                {
                     // ce qu'on doit faire quand le bouton est pressé
                }
            }));
```

5.3 Adaptateurs

Il existe un nombre de classes **adaptateures** prédéfinies pour simplifier le code de nos écouteurs. Par exemple, la classe `MouseAdapter` implante `MouseListener` et `MouseMotionListener` (code ne fait rien) — il faut juste redéfinir les méthodes qui sont importantes.

```
JPanel panneau = new JPanel();
...
MouseAdapter souris = new MouseAdapter() // classe locale
    {
        public void mouseClicked(MouseEvent e)
        {
            if (SwingUilities.isLeftMouseButton(e))
            {
                // clique avec le bouton gauche
            }
            if (e.isPopupTrigger())
            {
                // menu popup
            }
        }
        public void mouseDragged(MouseEvent e)
        {
            System.out.println("Ne glisse pas mon souris!");
        }
    };
panneau.addMouseListener(souris);
panneau.addMouseMotionListener(souris);
```

5.4 Timer

Un objet de type `javax.swing.Timer` est un «réveil» répétitif.

```
import javax.swing.Timer;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
...
    ActionListener tacheRepetitive = new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                // ce qu'on doit faire quand le Timer nous signale
                // executé dans l'EDT
            }
        };
    int delai = 1000; // millisecondes
    Timer timer = new Timer(delai, tacheRepetitive);
    timer.start(); // lancer
```

Ce mécanisme peut être exploité pour des animations.