Interface graphique

PO2 (2014-2015)

Chapitre 4 Evénements

Evénements

1. Evénements
2. Traitement des événements
3. Evénements et écouteurs/Ecouteurs

Evénements

- Au sens large, il s'agit d'un changement d'état d'un objet dans une intervalle de temps.

Evénements

- Au sens propre pour la programmation graphique, lorsque l'utilisateur interagit avec une interface graphique (en appuyant sur une touche du clavier ou un bouton de souris par exemple), un (ou plusieurs) événement est généré .

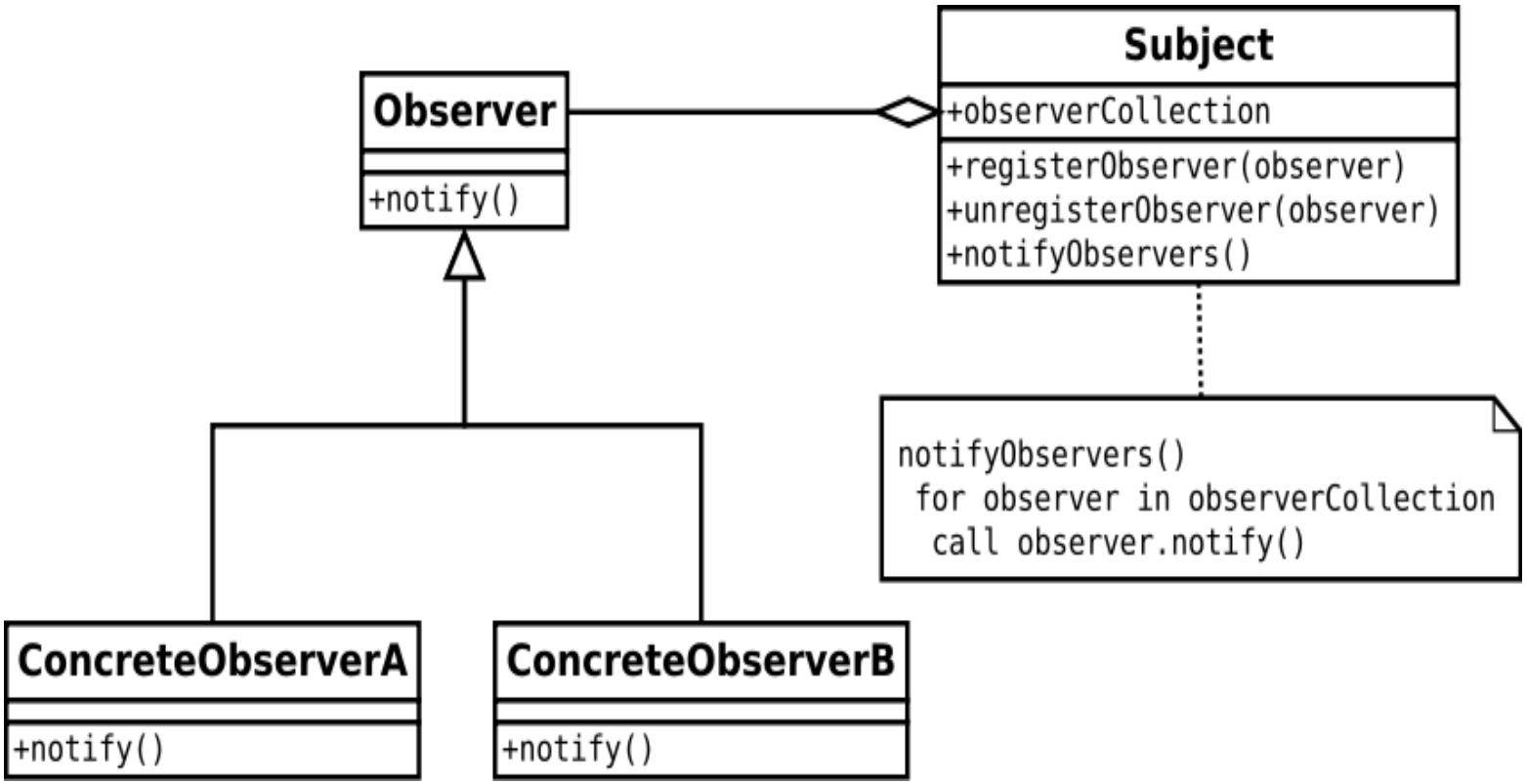
Traitement des événements

- La programmation événementielle
  - répond aux différents événements.
  - utilise un design pattern Observateur/Observé pour le traitement des événements.

Observateur/Observé

- Le design pattern Observateur/Observé définit une relation entre un objet et plusieurs objets, de façon que, si un objet change d'état, tous ceux qui en dépendent en soient informés et mis à jour automatiquement.
- Ce design pattern contient trois éléments :
  - Observé (Observable)
    - Associé à une liste des observateurs
    - Produit des évènements
    - Notifie les observateurs lorsque un évènement se produit
  - Evénement (Event)
    - Un changement d'état de l'observé
  - Observateur (Observer)
    - Traite des événements venant de l'observé

Observateur/Observé



Observateur/Observé

- Avec GUI :
  - Composant (Observé)
    - Produit l'événement (généralement en conséquence d'une action de l'utilisateur).
  - Evénement
    - ComponentEvent
    - ContainerEvent
    - MouseEvent
    - WindowEvent
    - keyEvent
  - Ecouteur (Observateur)
    - Traite l'événement en exécutant une méthode en rapport avec la nature précise de celui-ci.

Evénements et écouteurs

- Types d'événements et d'écouteurs
  - ActionEvent /ActionListener
  - MouseEvent /MouseListener, MouseMotionListener
  - WindowEvent /WindowListener

ActionEvent et ActionListener

- ActionEvent correspond au déclanchement de l'action sur un composant graphique:
  - Le clic sur un bouton
  - La frappe de la touche « enter » sur un TextField
  - Etc.
- http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/event/ActionEvent.html

Classe interne : JetDeEvent1

```
JetDePanel() {
    // creer deux De
    deGauche  = new DePanel();
    deDroit = new DePanel();
    // creer deux buttons
    jetButton1 = new JButton("Joueur1");
    jetButton2 = new JButton("Joueur2");
    // créer une instance de JetListener
    JetListener jetListener = new JetListener();
    jetButton1.addActionListener(jetListener);
    jetButton2.addActionListener(jetListener);
    //... placer components
    this.setLayout(new BorderLayout());
    this.add(jetButton1, BorderLayout.NORTH);
    this.add(deGauche , BorderLayout.WEST);
    this.add(deDroit, BorderLayout.EAST);
    this.add(jetButton2, BorderLayout.SOUTH);
  }
```

- JetListener sous forme de classe interne
- Avantage :
  - accède facilement aux variables d'instance de la classe JetPanel
  - permet plusieurs instances

Classe interne : JetDeEvent1

```
// classe interne : JetListener
   private class JetListener implements
      ActionListener {
       public void actionPerformed(ActionEvent
      e) {
        Object source = e.getSource();
        if (source == jetButton1)
                deGauche.jet();
        else
                deDroit.jet();
      }
   }
}
```

- Les événements écoutés sont de la classe "ActionEvent" qui hérite de la superclasse Event
- Méthodes permettant d'obtenir des informations sur ce qui est survenu
  - getSource() donne le composant source de l'événement
  - getActionCommand() donne le nom d'action, par défaut le texte du bouton, du menu, ...

Classe interne anonyme : JetDeEvent2

```
// creer deux buttons
jetButton1 = new JButton("Joueur1");
jetButton2 = new JButton("Joueur2");
// classes interne anonyme
jetButton1.addActionListener(new
 ActionListener() {
 public void actionPerformed(ActionEvent e) {
            deGauche.jet();
 }
 }
    );
    jetButton2.addActionListener(new
 ActionListener() {
 public void actionPerformed(ActionEvent e) {
            deDroit.jet();
 }
 }
    );
```

- JetListener sous forme de classe interne anonyme
- Avantage :
  - accède facilement aux variables d'instance de la classe JetDePanel
- Inconvénient :
  - n'a qu'1 instance
  - code moins lisible

MouseEvent et MouseListener

- La souris (le curseur) passant sur la surface graphique du composant, içi un JPanel, provoque MouseEvent :
- Le programme MouseEvent.java affiche ces événements en console :

```
mouseMoved - MouseEvent :
java.awt.event.MouseEvent[MOUSE_MOVED(197,0),button=0,click
Count=0] on javax.swing.JPanel[,
0,0,200x100,layout=java.awt.FlowLayout,
alignmentX=0.0,alignmentY=0.0,border=,flags=9,maximumSize=,
minimumSize=,
preferredSize=java.awt.Dimension[width=200,height=100]]
mouseExited - MouseEvent :
java.awt.event.MouseEvent[MOUSE_EXITED(197,-
1),button=0,clickCount=0] on javax.swing.JPanel[,
0,0,200x100,layout=java.awt.FlowLayout,
alignmentX=0.0,alignmentY=0.0,border=,flags=9,maximumSize=,
minimumSize=,
preferredSize=java.awt.Dimension[width=200,height=100]]
```

MouseEvent et MouseListener

```
panel.addMouseListener(new MouseListener() {
     public void mouseClicked(MouseEvent me) {
          System.out.println("mouseClicked - MouseEvent : "+me.toString());
     }
     public void mouseEntered(MouseEvent me) { … }
     public void mouseExited(MouseEvent me) { …}
     public void mousePressed(MouseEvent me) { … }
     public void mouseReleased(MouseEvent me) { …}
});
```

Les méthodes des MouseListener :

- mouseEntered() : entrée de souris
- mouseExited() : sortie de souris
- mousePressed() : bouton pressé
- mouseReleased() : bouton relaché
- mouseClicked() : bouton pressé puis relaché dans sa zone graphique

Ces méthodes permettent de recevoir et d'écouter les événements qui se sont produits

MouseEvent et MouseMotionListener

```
panel.addMouseMotionListener(new MouseMotionListener() {
        public void mouseMoved(MouseEvent me) { …   }
        public void mouseDragged(MouseEvent me) { …      }
});
```

Les méthodes desMouseMotionListener :

- mouseMoved() : mouvement de souris
- mouseDragged() : mouvement de souris avec bouton enfoncé

L'événement en paramètre donné des méthodes est celui qui s'est produit :
il contient des informations de ce qui s'est passé.