Interfaces graphiques

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

January 26, 2016

Le menu

Mettre en place la barre de menu :

```
JMenuBar menuBar = new JMenuBar();
jFrame.setMenuBar(menuBar);
```

Ajouter un nouveau menu avec un mnemonic :

```
JMenu fileMenu = new JMenu("Ficher");
fileMenu.setMnemonic(KeyEvent.VK_F);
```

Le mnemonic : il faut choisir une des lettres du nom du menu (la lettre sera soulignée).

Pour activer ce mnemonic par le clavier : ALT + f

Le mnemonic est activé si son père possède le focus.

Ajouter un JMenuItem dans JMenu

```
JMenuItem q = new JMenuItem("Quitter");
// Installer l'accelerateur CTRL-Q pour quitter
// KeyStroke.getKeyStroke(int keyCode, int modifiers)
q.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,
                                InputEvent.CTRL_DOWN_MASK));
//installer un mnemonic pour Quitter
q.setMnemonic(KeyEvent.VK_Q);
//ajouter ActionListener
q.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
          System.exit(0);
   }
});
fileMenu.add(q);
```

Noter l'affichage de la séquence de touches qui définissent l'accélérateur.

Différence entre le mnemonic et l'accélérateur : l'accélérateur peut être activé même quand JMenuItem n'est pas visible.

Touche de modification

```
JMenuItem q = new JMenuItem("Quitter");
// Installer l'accelerateur CTRL-Q pour quitter
// KeyStroke.getKeyStroke(int keyCode, int modifiers)
q.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,
                                        InputEvent.CTRL_DOWN_MASK));
```

Mais la touche de modification dépend de système ! Le code suivant permet d'obtenir la touche de modification appropriée pour le système donné (CTRL pour Linux, command pour mac).

```
JMenuItem q = new JMenuItem("Quitter");
// Installer l'accelerateur CTRL-Q pour quitter
// KeyStroke.getKeyStroke(int keyCode, int modifiers)
q.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_Q,
   Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()  )
```

Implémenter les actions avec `AbstractAction`

`AbstractAction` permet d'implémenter de façon uniforme les actions de l'utilisateur dans

- `JMenu`,
- `JPopupMenu` et
- `JToolBar`.

AbstractAction – classe abstraite

- Créer une sous classe de `AbstractAction`
- implémenter la méthode
  `public void actionPerformed(ActionEvent e)`
- ajouter l'accélérateur, mnemonic et description si besoin.

```
class MoveAction extends AbstractAction {
  private int dx, dy;
  //constructeur
  public MoveAction(int dx, int dy, String text,
                    ImageIcon icon){
    /* on passe String et icon vers AbstractAction */
    super(text, icon);
    this.dx=dx; this.dy=dy;
  }
  public void actionPerformed(ActionEvent e) {
    // Action sur le modele
    model.move(dx, dy);
  }
}
```

Créer et enregistrer `MoveAction` à la place de `JMenuItem` dans le JMenu :

```
//creer JMenu pour le mouvements
JMenu movesMenu = new JMenu("Moves");
movesMenu.setMnemonic(KeyEvent.VK_M);

MoveAction up = new MoveAction(0,-1,"Move up", iconUp);
//accelerateur
up.putValue(AbstractAction.ACCELERATOR_KEY,
    KeyStroke.getKeyStroke(KeyEvent.VK_UP,
     Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
//mnemonic
up.putValue(AbstractAction.MNEMONIC_KEY,
                KeyEvent.VK_U);
//description (tooltip)
up.putValue(SHORT_DESCRIPTION,
                "Faire bouger vers le haut");
// activer ou desactiver l'action up (valeur initiale)
up.setEnabled(false);
//ajouter MoveAction dans le JMenu
movesMenu.add(up);
```

AbstractAction dans JToolBar

```
JToolBar toolBar = new JToolBar();
toolBar.add(up);
```

Comment installer JToolBar ?

```
Container contentPane = frame.getContentPane();
contentPane.setLayout(new BorderLayout());
contentPane.add(toolBar, BorderLayout.NORTH);
```

AbstractAction dans le menu flottant

```
final JPopupMenu popupMenu = new JPopupMenu();
//ajouter MoveAction
popupMenu.add(up);
```

Comment installer JPopupMenu ?

arena un JPanel – un clic sur arena lancera le menu flottant :

```
// Menu popup dans l'arene (JPanel
arena.addMouseListener(new MouseAdapter() {
  public void mousePressed(MouseEvent e) {
        showPopup(e);
  }
  public void mouseReleased(MouseEvent e) {
        showPopup(e);
  }
  /*
   * PopupMenu est declenche soit par mousePressed soit
   * mouseReleased en fonction de la plateforme d'ou la
   * d'appeler isPopupTrigger() dans les deux cas pour v
   * faut afficher popupMenu
  */
  private void showPopup(MouseEvent e) {
     if (e.isPopupTrigger())
        popupMenu.show(e.getComponent(),
                       e.getX(), e.getY());
     }
```

Lire un fichier d'image et créer `ImageIcon`

```
ImageIcon icon =
new ImageIcon(getClass().getResource(chemin_vers_le_fiche
```

A noter que

```
new ImageIcon(chemin_vers_image)
```

ne marche pas si le programme mis en dans fichier jar.

Sélectionner un ficher pour écriture avec `JFileChooser`

```
JMenuItem save = new JMenuItem("Sauvegarder");
save.addActionListener(new ActionListener() {
   public void actionPerformed(ActionEvent e) {
   JFileChooser chooser = new JFileChooser();
   int result = chooser.showSaveDialog(frame);
   if (result == JFileChooser.APPROVE_OPTION)
         save(chooser.getSelectedFile());
   }
});
```

La méthode `getSelectedFile()` de `JFileChooser` retourne un objet de type `File`.

Sauvegarde de données si `File` connu

```
void save(File file) {
 DataOutputStream dos;
 try {
   dos = new DataOutputStream(new FileOutputStream(file));
 } catch (FileNotFoundException e) {
 System.err.println(file + " not found");
   return;
 }
 try {
   dos.writeInt(model.posx);
   dos.writeInt(model.posy);
   dos.close();
 } catch (IOException e) {
   System.err.println("IOException");
   return;
 }
 return;
}
```

Sélectionner un fichier pour la lecture avec `JFileChooser`

```
JMenuItem open = new JMenuItem("Ouvrir");
open.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    JFileChooser chooser = new JFileChooser();
    int result = chooser.showOpenDialog(frame);
    if (result == JFileChooser.APPROVE_OPTION)
        ouvrir(chooser.getSelectedFile());
    }
});
```

Noter `showOpenDialog()` contre `showSaveDialog()` précedemment.

Récupérer des données si `File` connu

```
void ouvrir(File file){
  DataInputStream dis ;
  try{
    dis = new DataInputStream(new FileInputStream(file))
  }catch(FileNotFoundException e){
    System.err.println(file + " not found");
    return;
  }
  try {
    model.move(dis.readInt(),dis.readInt());
    dis.close();
  } catch (IOException e) {
    System.err.println("IOException");
    return;
  }
  return;
}
```

Positionner l'image sur un JPanel

```
JPanel arena = new JPanel();
arena.setLayout(null);
ImageIcon icon = new ImageIcon(....);
JLabel mobile= new JLabel(icon);
mobile.setSize(icon.getIconWidth(), icon.getIconHeight());
arena.setPreferredSize(new Dimension( hauteur, largeur);
arena.add(mobile);
mobile.setPosition( positionX, positionY );
```

Communication entre le modèle (Observable) et la vue (Observer)

```
class Model extends java.util.Observable{
   ..........
   void move(int dx, int dy) {
     if (isMoveEnabled(dx, dy)) {
          posx += dx;
          posy += dy;
     }
     setChanged(); //necessaire, sinon notifyObservers()
     notifyObservers();
}
```

Et les vues comme java.util.Observer

```
class MoveAction extends AbstractAction
                         implements Observer {
 public void update(Observable o, Object arg){

  /* La methode setEnabled(boolean newValue) active
   * ou desactive l'action                          */
  setEnabled(((Model) o).isMoveEnabled(dx, dy));
 }
...
}

MoveAction up = new MoveAction( .... );
model.addObserver(up);
```