Interfaces graphiques

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

February 13, 2016

Travailler avec des images

Pour connaître les types d'image gérés par swing on utilise les méthodes statiques de la classe

`ImageIO`

`String[] getReaderFileSuffixes()`

retourne les suffixes de fichiers d'images que swing gère en lecture.

`String[] getWriterFileSuffixes()`

retourne les suffixes de fichiers d'images que swing gère en écriture.

L'image dans un fichier $\leftrightarrow$ BufferedImage

Pour transférer une image stockée dans un fichier dans BufferedImage] utilisez une méthode static read de la classe ImageIO :

```
BufferedImage BufferedImage = ImageIO.read(file);
```

où file un objet de type File ou InputStream ou URL.
Pour transférer une image d'un BufferedImage] vers un fichier utilisez la méthode static write de ImageIO :

```
ImageIO.write(BufferedImage,ext,file);
```

où

- file un objet de type File,
- ext – un String qui spécifie le format de l'image.

Exemple de lecture/écriture d'image

```
BufferedImage bufferedImage;
try{
    bufferedImage = ImageIO.read(file);
}catch(IOException e){
    JOptionPane.showMessageDialog(this,
                "Impossible to open the file " +
                 file.toString());
}
```

```
File file = new File("toto.png");
try{
    ImageIO.write(bufferedImage,"png", file);
}catch(IOException e){
     JOptionPane.showMessageDialog(this,
                 "Impossible to save to the file " +
                  file.toString());
```

JFileChooser avec un filtre

Nous voulons lire un fichier d'image en limitant le choix dans JFileChooser aux fichiers d'image gérés par swing (et tous les répertoires).

```
String[] rs = ImageIO.getReaderFileSuffixes();
String s = "image files:";
for(String e : rs){  s += " " + e; }
FileFilter rFilter = new FileExtensionFilter(s,rs);
JFileChooser chooser = new JFileChooser();
chooser.setFileSelectionMode(
            JFileChooser.FILES_AND_DIRECTORIES);
chooser.setFileFilter(rFilter);
int res = chooser.showOpenDialog(frame);
if( res == JFileChooser.APPROVE_OPTION ){
    File file = chooser.getSelectedFile();
    if( ! rFilter.accept(file) ){
        JOptionPane.showMessageDialog(frame,"not valid image")
        return;
    }
    try{
      bufferedImage = Image.read(file);
    }catch(IOException e){  //traiter l'exception
```

Un peu d'animation — `javax.swing.Timer`

(à ne pas confondre avec `java.util.Timer`)

`Timer` permet d'effectuer périodiquement une tâche.

```
int delay = 1000; //milliseconds
ActionListener taskPerformer = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        //...effectuer une tache...
    }
};
new Timer(delay, taskPerformer).start();
```

- créer un `Timer` et enregistrer un ou plusieurs `ActionListeners` (la méthode `addActionListener(ActionListener listener)`)
- spécifier le periode – `setDelay(int delay)`,
- faire marcher/arrêter/redemarrer le `Timer` : `start()`, `stop()`, `restart()` .

Enregistrer les actions associées au clavier

`JComponent` (et chaque widget qui hérite de `JComponet`) possède les méthodes

```
InputMap getInputMap(int condition)
ActionMap getActionMap()
```

`InputMap` permet d'associer `KeyStroke` (suite des touches du clavier) avec une clé d'action:

```
void InputMap.put(KeyStroke keys, Object cle)
```

`ActionMap` permet d'associer une clé d'action avec une action:

```
void ActionMap.put(Object cle, Action action)
```

Exemple – utiliser `ActionMap` et `InputMap` pour arrêter/redémarrer une animation

```
// key "s" stops Timer
panel.getInputMap(JComponent
        .WHEN_IN_FOCUSED_WINDOW)
        .put(KeyStroke.getKeyStroke('s'),"stop");
panel.getActionMap().put("stop",new AbstractAction(){
        public void actionPerformed(ActionEvent e){
            timer.stop();
        }
    });
//  key "a"  starts Timer
panel.getInputMap(JComponent.WHEN_IN_FOCUSED_WINDOW)
     .put(KeyStroke.getKeyStroke('a'),"start");
panel.getActionMap().put("start",new AbstractAction(){
     public void actionPerformed(ActionEvent e){
            timer.start();
     }
   });
```

JComboBox

Exemple d'utilisation de JComboBox :

```
Unit[] unit; /* un tableau d'elements a mettre dans JComboBox
//initialiser et remplir le tableau units
//creer JComboBox preremplie avec les elements du tableau unit
JComboBox<Unit> combo = new JComboBox<Unit>(units);
combo.setSelectedIndex(0); /*valeur initiale*/
//ajouter un ActionListener
combo.addActionListener(new AnctionListener() {
    public void actionPerformed(ActionEvent e){
        int i = combo.getSelectedIndex();
        ....
    }
});
```

JComboBox<T> est une classe paramétré par uns classe T d'éléments qui sont affichés dans JComboBox. On suppose que la classe T possède la méthode `String toString()` qui est utilisée par swing pour trouver les chaînes de caractères affichées dans JComboBox.

JTextField

JTextField est un widget qui permet d'écrire une ligne de texte.
Si plusieurs lignes utilisez la classe JTextArea.

```
JTextField tf = new JTextField(20);
tf.setText("text inital");

//ActionListener est active quand l'utilisateur tape ENTER
tf.addActionListener(new ActionListener(){
    public vois actionPerformed(ActionEvent e){
        //recuperer le texte tape par l'utilisateur
        String s = tf.getText();
        ...
    }
});
```

JList

JList une liste dont un ou plusieurs éléments peuvent être sélectionnés par l'utilisateur.
JList est construit selon le principe Modèle/Vue/Controlleur.
Le modèle contient le données. Un de modèles possibles la classe DefaultListModel.

```
//instantier le modele
final DefaultListModel<String> leftModel =
                    new DefaultListModel<String>();
//ajouter des elements dans le modele
leftModel.addElement("Nathalie");
leftModel.addElement("Sylvie");
//instantier JList en lui assiciant le modele
final JList<String> leftList = new JList<String>(leftModel);
//trouver les elements de la JList selectionnes
//par utilisateur
List<String> values = leftList.getSelectedValuesList();
//supprimer des elements du modele
//(et par le meme de la liste)
for(String s : values)
  leftModel.removeElement(o);
```

Composer les éléments graphiques avec `AlphaComposite`

//obtenir une instance de AlphaCompositte

```
public static AlphaComposite.getInstance(int regle,
                                         float alpha)
```

Règles de compostion :

```
AlphaComposite.SRC
AlphaComposite.DST_IN
AlphaComposite.DST_OUT
AlphaComposite.DST_OVER
AlphaComposite.SRC_IN
AlphaComposite.SRC_OVER
AlphaComposite.SRC_OUT
AlphaComposite.CLEAR
```

`alpha` - transparence entre 0 (complètement transparent) et 1 (complètement opaque).

Exemple

```
BufferedImage buffImg = new BufferedImage(w, h,
                                BufferedImage.TYPE_INT_ARGB);
Graphics2D gbi = buffImg.createGraphics();

//dessiner un rectangle
gbi.setColor(new Color(0.0f, 0.0f, 1.0f, 1.0f));
gbi.fill(new Rectangle2D.Double(rectx, recty, 150, 100));

//definir la regle de composition
AlphaComposite ac = AlphaComposite.getInstance(regle, alpha);
gbi.setComposite(ac);

//dessiner une ellipse
gbi.setColor(new Color(1.0f, 0.0f, 0.0f, 1.0f));
gbi.fill(new Ellipse2D.Double(rectx+rectx/2,recty+recty/2,150
```