Interfaces graphiques

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

February 3, 2016

Construire votre propre composant graphique

- Implémenter une sous-classe de `JComponent` ou `JPanel`
- reimplémenter la méthode

```
protected void paintComponent(Graphics g)
```

Dans la méthode `paintComponent()` écrire le code qui dessine le composant

- affiche différente formes : lignes, rectangles, ellipses, courbes,
- affiche les textes,
- affiche les images.

Exemple

```
public MonComposant extends JComponent{
    protected void paintComponent(Graphics g){
        Graphics2D g2d = (Graphics2D) g;
        g2d.drawOval(10,10,100,150);
        g2d.drawRect(5, 5, 110, 160);
    }
}
```

dessine une ellipse et un rectangle.

Pourquoi un cast vers `Graphics2D` ?

Le paramètre déclaré de `paintComponent` est de type `Graphics` mais en réalité c'est un objet `Graphics2D`.
`Graphics2D` hérite toutes les méthodes de `Graphics` et ajoute plusieurs nouvelles méthodes permettant de construire de composants plus sophistiquées.

Quand le composant est (re)dessiné ?

Chaque fois que Swing détecte que le composant devient visible ou la fenêtre change la position sur l'écran Swing fait appel à `paintComponent()` pour redessiner le composant.

Pour faire redessiner le composant depuis java il faut faire appel à la méthode

```
public void repaint()
```

ou

```
public void repaint(int x, int y, inth largeur, int hauteur)
```

MouseListener

L'interface pour capturer les actions de la souris.
Les méthodes :

- `void mouseClicked(MouseEvent e)` – clique de la souris sur le composant,
- `void mouseEntered(MouseEvent e)` – le courser entre dans le composant,
- `void mouseExited(MouseEvent e)` – le courser sort du composant,
- `void mousePressed(MouseEvent e)` – bouton appuyé,
- `void mouseReleased(MouseEvent e)` – bouton relâché.

`MouseAdapter` – une classe qui implémente MouseListener avec les méthodes vides.

Enregistrer un listener :

```
panel.addMouseListener(new MouseAdapter(){
        // implementer les methodes pour les evenements
        // qui vous interessent
    });
```

MouseMotionListener

Les méthodes :

- `void mouseDragged(MouseEvent e)` – appuyer sur le bouton de la souris et déplacer,
- `void mouseMoved(MouseEvent e)` – courser entre dans le composant, pas de bouton appuyé.

`MouseMotionAdapter` – classe qui implémente `MouseMotionListener` avec les méthodes vides.

Enregistrer un listener :

```
panel.addMouseMotionListener(new MouseMotionAdapter(){
        // implementer les methodes pour les evenements
        // qui vous interessent
    });
```

Exemple - dessiner un cercle dynamique

```
public class Draw extends JComponent {
  private int cx, cy; /*centre*/
  private int rx, ry; /point courant*/
  private boolean draw, finished;

  public void createGUI(){
    /* ajouter les listener */
    addMouseListener(new MouseAdapter() {
      public void mousePressed(MouseEvent e) {
        cx = e.getX(); cy = e.getY(); draw = true;
    }
    public void mouseReleased(MouseEvent e) {
        draw = false; finished = true;
        rx = e.getX(); ry = e.getY();
        repaint();
    });

    addMouseMotionListener(new MouseMotionAdapter() {
        public void mouseDragged(MouseEvent e) {
         rx = e.getX(); ry = e.getY(); repaint();
        }
    });
```

Dissiner un cercle (et un rayon)

```
 float thickness = 1.0f; //largeur de la ligne
protected void paintComponent(Graphics g) {
  Graphics2D g2d = (Graphics2D) g;
  if (draw) {
    g2d.drawLine(cx, cy, rx, ry);
  }
  if (draw || finished) {
    BasicStroke stroke = new BasicStroke(thickness);
    g2d.setStroke(stroke);
    double rayon = Math.sqrt((cx - rx) * (cx - rx) +
                             (cy - ry) * (cy - ry));
    g2d.drawOval((int) (cx - rayon), (int) (cy - rayon),
                 2 * (int) rayon, 2 * (int) rayon);
    g2d.setStroke(new BasicStroke(1.0f));
  }
}
```

BasicStroke

L'objet `BasicStroke` permet de spécifier la largeur de ligne (et d'autres caractéristiques - type de jointure de lignes, ligne continue ou non).
Pour dessiner les lignes de 3 pixels :

```
Grahics2D g2d;

g2d.setStroke(new BasicStroke(3.0f));
//a partir de ce moment toutes les lignes de
//largeur 3
```

Ensuite toutes les lignes dessinées avec l'objet Graphics2D aurons la largeur 3 pixels jusqu'au changement de stroke avec un nouveau appel à `setStroke()`.

Afficher un dialogue – `JOptionPane`

Pour les dialogues simples utilisez les méthodes statiques de `JOptionPane`:

- `showConfirmDialog` – demande de confirmation comme oui/non/cancel
- `showInputDialog` – demander un input de la part de l'utilisateur
- `showMessageDialog` – pour informer l'utilisateur d'un evenement
- `showOptionDialog` – une unification de trois précédent.

Exemple de dialogue

```
Object o = JOptionPane.showInputDialog(
    c, //composant parent
    "Choisir la largeur de la ligne", //message
    "Largeur ligne", //titre
    JOptionPane.QUESTION_MESSAGE, //type de message
    null, //icon
    //table de valeurs a selectionner
    new Integer[] { new Integer(1), new Integer(2),
                    new Integer(3), new Integer(4),
                    new Integer(5), new Integer(6) },
    new Integer(i));//valeur initiale
    //recuperer la valeur selectionnee
    if (o != null)
        i = ((Integer) o).intValue();
```

AbstractAction complete

```
class Thickness extends AbstractAction {
  private int i; private Component c;

  public Thickness(Component c, int i) {
    super("largeur ligne");
    this.i = i; this.c = c;
  }
  public int getThickness() {
    return i;
  }

  @Override
  public void actionPerformed(ActionEvent e) {
    //ici mettre le code de la page precedente
  }
}
```

Et dans la classe principale `Draw` ajoutez :

```
public class Draw extends JComponent {
private Thickness thickness;

public void createGUI() {
  JFrame frame = new JFrame("draw");
  frame.setContentPane(this);
  JMenuBar menuBar = new JMenuBar();

  JMenu actions = new JMenu("Actions");
  menuBar.add(actions);

  thickness = new Thickness(this, 1);
  actions.add(thickness);
  frame.setJMenuBar(menuBar);
```

et dans `paintComponent()`:

```
BasicStroke stroke =
    new BasicStroke((float) thickness.getThickness());
```

JColorChooser

Pour choisir une couleur utliser `JColorChooser`.
La méthode static de `JColorChooser` :

```
public static Color showDialog(Component component,
    String title,
    Color initialColor)
```

Choisir couleur du cercle

Ajouter variable d'instance :

```
Color color = Color.BLACK;
```

Dans `createGUI` ajouter :

```
JMenuItem colorMenu = new JMenuItem("choisir couleur");
colorMenu.addActionListener(new ActionListener(){
  @Override
  public void actionPerformed(ActionEvent e) {
    color = JColorChooser.showDialog(Draw.this,
                "choisir couleur ligne", color);
  }
});
actions.add(colorMenu);
```

Et si nous voulons dessiner plusieurs cercles ?

On peut maintenir une liste de cercles (avec leurs couleurs). Et `paintComponet()` devrait parcourir cette liste pour dessiner tous les cercles. Fastidieux.

Mieux: Faire le dessin dans `BufferedImage` et dans `paintComponent` juste afficher BufferedImage sur l'écran.

Ajouter les variables d'instance :

```
private BufferedImage bi;
private Graphics2D big2d;
private int l=300, h=400;
```

et dans `createGUI()` ajouter :

```
bi = new BufferedImage(l /*largeur*/,
                       h  /*hauteur*/,
                       /*type d'image*/
                       BufferedImage.TYPE_3BYTE_BGR);
big2d = (Graphics2D) bi.createGraphics();
//pour obtenir le fond blanc
big2d.setColor(Color.WHITE);
big2d.fillRect(0, 0, l, h);
```

Type d'image : une constante de la classe `BufferedImage`, ici 3 octets par pixel pour spécifier 3 couleurs de base (Blue, Green, Red).

Écrire une méthode pour dessiner dans `BufferedImage` et à la fin dessiner sur l'écran :

```
private void dessiner() {
  if (draw) {
    big2d.setColor(Color.BLACK);
    big2d.drawLine(cx, cy, rx, ry);
  }
  if (draw || finished) {
    big2d.setColor(color);
    BasicStroke stroke = new BasicStroke((float)
          thickness.getThickness());
    big2d.setStroke(stroke);
    double rayon = Math.sqrt((cx - rx) * (cx - rx) +
                             (cy - ry) * (cy - ry));
    big2d.drawOval((int) (cx - rayon), (int) (cy - rayon),
                   2 * (int) rayon, 2 * (int) rayon);
    big2d.setStroke(new BasicStroke(1.0f));
  }
  repaint();
}
```

Et enfin remplacer toutes les autres occurrences de `repaint()` par `dessiner()`.

Est-ce que ça marche ? Presque.

Comment corriger le problème ?

Et enfin remplacer toutes les autres occurrences de `repaint()` par `dessiner()`.

Est-ce que ça marche ? Presque.

Comment corriger le problème ?

- Dessiner dans `BufferedImage` uniquement les cercles terminés,
- afficher sur l'écran le contenu de `BufferedImage` et, éventuellement le dernier cercle non terminé et le rayon,
- dans `mouseReleased()` ajouter le dernier cercle dans `BufferedImage`.