Outils Graphiques

Plan

- Outils Graphiques
  - Quand / Pourquoi ?
  - Dessin
  - Ce qui peut être réalisé
  - Java 2D

Elaborer une Application Graphique…

- Thématique d'un cours dans sa totalité…

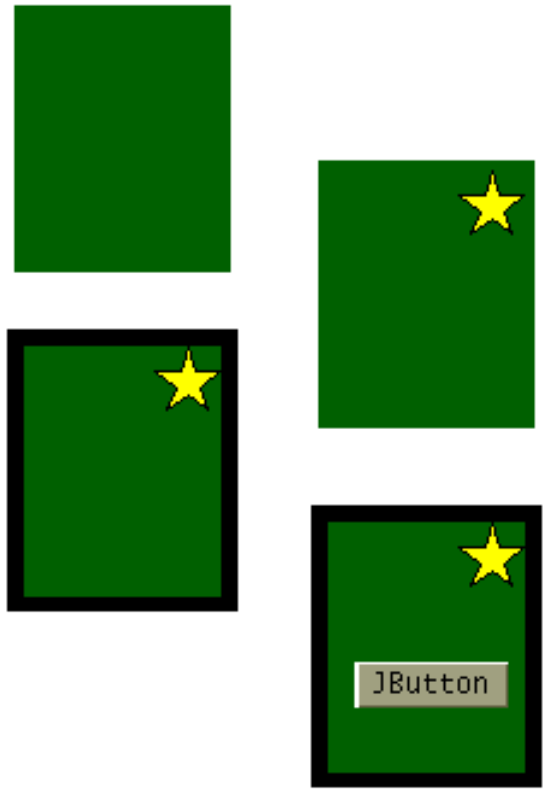
- Utilisation des tutoriels et docs API à :
- http://java.sun.com/docs/books/tutorial/uiswing/14painting/index.html
- http://java.sun.com/docs/books/tutorial/2d/TOC.html#display

Quand Ai-je Besoin d'Utiliser des Outils Graphiques ?

- Changement de l'apparence d'un composant existant
- Dessin de formes/texte à l'écran
- Animations
- "Invention" / création sur mesure d'un composant

Ordre de Dessin

- Visibilité basée sur la hiérarchie de composition
  - Arrière-plan
  - Dessin sur mesure
  - Bordures
  - Composants



Outils Graphiques Swing Disponibles

- Outils Java AWT Graphics
  - Basiques mais souvent suffisants
  - Formes
  - Texte
  - Images
- Outils Java 2D Graphics
  - Extension des outils AWT avec e.g. gradients, textures etc.

Dessin sur Mesure

- Entre l'arrière-plan et les bordures
- Méthode : hériter d'un composant puis adapter sur mesure
  - JPanel recommendé
    - e.g. class myPaintingPanel extends JPanel {
  - Possibilité d'utiliser des composants seuls
    - e.g. class myPaintingButton extends JButton {
- Code dans la méthode outrepassée paintComponent()
  - public void paintComponent (Graphics g)
  - Ne pas reproduire les fonctionnalités de dessin existantes!
  - Appel à super.paintComponent(g)

Dessin sur Mesure - Exemple

```
import javax.swing.SwingUtilities;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.BorderFactory;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;

public class PaintFenetre extends JFrame {
  public PaintFenetre () {
      super();
      setContentPane(new MyPanel());
      setVisible(true); }

  class MyPanel extends JPanel {
      public MyPanel() {
              setBorder(BorderFactory.createLineBorder(Color.black)); }
      public Dimension getPreferredSize() {
              return new Dimension(250,200); }
      public void paintComponent(Graphics g) {
              super.paintComponent(g);
              // Draw Text
              g.drawString("Dessin dans le Panel sur mesure !",10,20); }
  }
}
```

Dessin de Zones sur Mesure

- Tailles de composants en pixels
  - (0,0) correspond au coin supérieur gauche
  - (*largeur*-1,*hauteur*-1) au coin inférieur droit
- Appel à la méthode `repaint` pour redessiner
  - repaint ()
    - Paint le composant et d'autres éventuellement si transparent
  - repaint (int leftx,int lefty,int width,int height)
    - Repaint seulement une zone spécifique

Outils Graphiques – Formes & Texte (1)

- Propriétés enregistrées dans des objets de type `Graphics`
  - Représentant un 'état'
    - E.g. Color, Font, etc.
- Méthodes de la classe `Graphics` peuvent dessiner...
  - Formes
    - lignes, rectangles, rectangles 3D, rectangles avec bords arrondis, figure ovales, arcs, polygônes
  - Texte
    - Dessin d'une chaîne de caractères à l'écran comme un élément graphique

Outils Graphiques – Formes & Texte (2)

- Dessiner une forme à l'écran
- Utilisation de la méthode `paintComponent`

```
void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.setColor(Color.RED);
    g.drawRect(x, y, largeur, hauteur);
}
```

- x et y sont les coordonnées du coin supérieur gauche

Outils Graphiques – Formes & Texte (3)

- Dessiner une chaîne de caractères à l'écran
- Utilisation de la méthode `paintComponent`

```
void paintComponent(Graphics g) {
   super.paintComponent(g);
   g.setFont(new Font("Serif",Font.PLAIN,12));
   g.drawString("Vive Swing!", x, y);
  }
```

- x et y sont les coordonnées du point de départ à l'écriture du texte

Exercice

- Mettre en œuvre une application consistant en une fenêtre avec un rectangle vert de bordure noire qui se déplace suite à un clic de la souris.
- Vous proposerez un menu avec un onglet '*A propos*' qui consiste en un item '*?*' présentant le concepteur de l'application dès lors que l'item est cliqué.
- On proposera à l'utilisateur désirant fermer la fenêtre de confirmer son choix à l'aide d'une boîte de dialogue.

Digression : Adaptateurs pour le traitement d'événements (1)

- Classes implémentant les interfaces d'écouteurs doivent implémenter toutes les méthodes
  - e.g. **`MouseListener`** a plusieurs méthodes : **`mouseClicked, mouseDragged, mousePressed`**…
- Peut engendrer du code superflu
  - Si vous avez seulement besoin de **`mouseClicked`** , les autres méthodes doivent tout de même être implémentées (mais vides)
- Alternative….

Digression : Adaptateurs pour le traitement d'événements (2)

- ... héritage à partir d'une classe `MouseAdapter`
  - hérite des méthodes de `MouseListener`. E.g.:

```
public class MaClasse extends MouseAdapter {
    ...
    unObjet.addMouseListener(this);
    ...
    public void mouseClicked(MouseEvent e) {
        //Traitement de l'événement...
    }
}
```

- Pas de volonté/possibilité d'héritage à partir d'une classe adaptateur ?
  - Utilisation de classes internes ou anonymes…