Cours №4

S. Szulman

IUT Informatique Villetaneuse

Septembre 2011

Plan du cours

Le graphisme

- L'affichage graphique implique le partage d'une ressource unique l'écran entre plusieurs applications et même plusieurs fenêtres d'une même application
- L'affichage est réalisé par le système d'exploitation qui exécute une méthode prédéfinie :
  - public void paint(Graphics g) sous AWT
  - public void paintComponent(Graphics g) sous Swing

La méthode paintComponent

- Par défaut, la méthode paintComponent appelle la méthode ComponentUI.update() qui efface et redessine le fond si le composant est opaque (comme JPanel par défaut).
- Lorsque la méthode paintComponent est redéfinie, la méthode de la classe mère doit être appelée par super.paintComponent pour conserver l'appel à ComponentUI.update().
- Le paramètre de la méthode paintComponent est de type Graphics mais la classe réelle de ce contexte graphique est toujours Graphics2D. Afin de pouvoir utiliser toutes les possibilités du graphisme de Java 2, il faut le convertir en un objet de type Graphics2D.

Redéfinition de la méthode paintComponent

Pour dessiner, il faut redéfinir la méthode paintComponent de la manière suivante :

```
public void paintComponent(Graphics g) {
    // Appel de la m thode paintComponent de la classe m re
    super.paintComponent(g);
    // Conversion en un contexte 2D   ventuellement   pour utiliser
   // les m thodes de l classe Graphics2D
    Graphics2D g2 = (Graphics2D) g;
    // Utilisation de g2
```

La méthode paintComponent

- La méthode paintComponent est appelée à chaque fois que le composant nécessite d'être redessiné, par exemple s'il a été masqué.
- La méthode ne peut être appelée directement.
- Pour forcer le dessin d'un composant, il faut appeler la méthode repaint(). Le principe de fonctionnement est le suivant :
  La méthode repaint poste un appel à la méthode update mais cet appel est différé. Il est traité après le traitement des évènements en cours. Plusieurs appels à la méthode update peuvent être regroupés.

Le contexte graphique

- C'est l'outil de dessin. C'est une instance de la classe Graphics ou Graphics2D pour Java 2.
- La classe Graphics est abstraite et elle ne possède pas de constructeur public.
- Les instances nécessaires sont fournies par le système d'exploitation qui instanciera via la machine virtuelle une sous classe de Graphics dépendante de la plateforme utilisée.

Les opérations sur le contexte graphique

- Le tracé de formes géométriques
  exemple : dessiner une droite entre les points (x1,y1) et (x2,y2) : drawLine(x1, y1, x2, y2)
- Ecrire du texte drawString(texte,x1,y1) : ecrire un texte à partir du point (x1,y1).
  On peut modifier la fonte (méthode setFont) et la couleur méthode setColor
- il existe bien d'autres méthodes ....

Programmation

- Définir une classe spécifique (héritant de JComponent ou d'une de ses sous-classes) correspondant au composant dans lequel on veut faire du dessin
- Rédéfinir dans cette classe la méthode void paintComponent (Graphics g) où on met le code indiquant ce qu'il faut dessiner dans le composant après avoir fait un appel à super.paintComponent(g).

Exemple

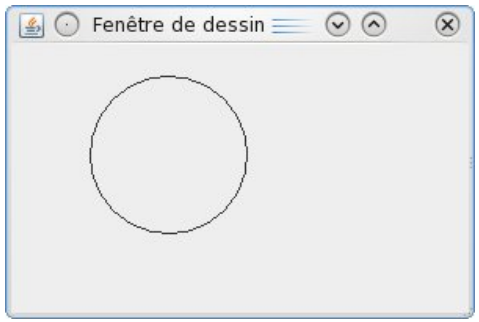
Tracer un cercle bleu de centre (100,70) et de rayon 50

- Créer une sous-classe de JPanel
- Redéfinir paintComponent(Graphics g)
  Utiliser g pour faire le bon dessin :
  Utilisation de la méthode drawOval

Exemple

```
public class FenAfficheCercle extends JFrame    {

public FenAfficheCercle(String titre, int w, int h)     {     super(titre);
       this.initialise();
       this.setSize(w,h);
       this.setVisible(true);
}
public void initialise()        {
        PanelCercle pan=new PanelCercle ();
        this.add(pan,BorderLayout.CENTER);
}
public static void main(String[] args) {
        new FenAfficheCercle("Fen tre de dessin",300,200);
        }
}
public class PanelCercle extends JPanel {
  public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawOval(50,20,100,100);
        }
 }
```



Deuxième Exemple

Tracer la diagonale d'une fenêtre quelque soit la fenêtre

- Utiliser drawLine de la classe graphics
- Prend en paramètre :
  - les coordonnées du coin haut gauche : toujours (0,0)
  - Les coordonnées du coin bas droit :Attention : pas définitivement fixés car changent suivant la dimension de la fenêtre qui vont être calculées dynamiquement largeur fenêtre= this.getWidth() ; hauteur fenêtre = this.getHeight() ;

Deuxième exemple


```
public class FenAfficheDiagonale extends JFrame {
public FenAfficheDiagonale(String titre, int w, int h)  {
        super(titre);
        this.initialise();
          this.setSize(w,h);
        this.setVisible(true);}

        public void initialise()        {
           PanelDiagonale pan=new PanelDiagonale();
           this.add(pan, BorderLayout.CENTER);
        }
        public static void main(String[] args) {
                new FenAfficheDiagonale("Fen tre de dessin diago",300,200);
        }
}
public class PanelDiagonale extends JPanel {

  public void paintComponent(Graphics g) {
        super.paintComponent(g);
        int larg= this.getWidth();
        int haut= this.getHeight();
        g.drawLine(0, 0, larg, haut);
        }
 }
```

Fonctionnement de l'affichage

- FenAfficheDiagonale contient un composant, instance de PanelDiagonale. Quand FenAfficheDiagonale s'ouvre, se redimensionne, la fenêtre est peinte puis son seul fils (un PanelDiagonale) est peint en appelant la méthode paintComponent définie dans PanelDiagonale
- Mais on ne sait pas quand l'appel a lieu
- C'est le composant qui doit connaître, à tout instant, les informations sur l'affichage qu'il doit réaliser Solution : le composant doit avoir ces informations en variables d'instance

Exemple suite

Choisir, dans un menu, la couleur d'affichage de la diagonale

- Analyse : qu'est-ce qui change ?
- Au niveau graphique : Munir la fenêtre d'un menu
- Au niveau événementiel : Créer une inner-classe CouleurListener écoutant les événements du menu, implémentant ActionListener Que doit faire la méthode actionPerformed ? Récupérer la couleur sélectionnée Modifier la couleur d'affichage du panneau graphique Et surtout, le panneau graphique doit toujours s'afficher dans la couleur choisie

Analyse suite

Le panneau graphique doit connaître à tout moment la couleur dans laquelle va s'afficher le rectangle

- mettre une variable d'instance Color coul dans PanelDiagonale
- coul sera utilisé dans la méthode paintComponent
- Il doit y avoir une communication entre l'objet fenêtre et l'objet panneau graphique pour que l'écouteur CouleurListener puisse changer cette couleur :
- Mettre le panneau graphique en variable d'instance de la fenêtre
- Créer une méthode setColor dans PanelDiagonale
- L'écouteur CouleurListener doit pouvoir forcer le réaffichage du panneau graphique. Pour cela utiliser la méthode repaint() qui permet de forcer la méthode paintComponent à s'éxécuter.

Programme

```
public class FenDiagonaleMenu extends JFrame  {
          private PanelDiagonale pan;
          private static final String rouge = "Rouge";
          private static final String bleu = "Bleu";

        public FenDiagonaleMenu(String titre, int w, int h)     {
                super(titre);
                this.initialise();
            this.setSize(w,h);
            this.setDefaultCloseOperation(EXIT_ON_CLOSE);
            this.setVisible(true);
                }

        public void initialise()        {
                pan=new PanelDiagonale();
                this.add(pan, BorderLayout.CENTER);
                this.initialiseMenu();
                }
public void initialiseMenu()  {
    JMenuBar jmb = new JMenuBar();
    this.setJMenuBar(jmb);

    JMenu mdef = new JMenu ("Couleur");
    jmb.add(mdef);
    JMenuItem iBleu= new JMenuItem (FenDiagonaleMenu.bleu);
        mdef.add(iBleu);
    JMenuItem iRouge= new JMenuItem (FenDiagonaleMenu.rouge);
        mdef.add(iRouge);

     iRouge.addActionListener(new CouleurListener());
     iBleu.addActionListener(new CouleurListener());
     }
```

Programme suite

```
class CouleurListener implements ActionListener  {

public void actionPerformed(ActionEvent e) {

        String s=e.getActionCommand();
        if (s.equals( s.equals(FenDiagonaleMenu.rouge))
            pan.setColor(Color.red);
                else
      pan.setColor(Color.blue);
        //pour forcer le reaffichage
        pan.repaint();
        }
}// fin CouleurListener
}// fin FenDiagonaleMenu
public class PanelDiagonale extends JPanel {
 private Color coul;

public PanelDiagonale ()  {
        coul=Color.black;
        }

public void setColor(Color c)   {
        coul=c;
}
public void paintComponent(Graphics g) {
        super. paintComponent(g);
        g.setColor(coul);
        int larg= this.getWidth();
        int haut= this.getHeight();
        g.drawLine(0, 0, larg, haut);
    }
}
```

Afficher une image

- Méthode de la classe Graphics : drawImage(Image im , int x, int y, ImageObserver obs) (x,y) coordonnées d'affichage dans le composant
- ImageObserver est une interface.L'argument de type ImageObserver passé aux méthodes est souvent this (doit être un objet d'une classe qui implémente l'interface ImageObserver, par exemple Component).

Afficher une image 2

Première étape : Créer une instance de la classe Image

```
Lecture dans un fichier local
String nom =  bleu .gif ;
Image image;
image = Toolkit.getDefaultToolkit().getImage(nom);

Lecture sur Internet
URL u = new URL( http ://www.quelquepart.com/image.jpg );
Image image = Toolkit.getDefaultToolkit().getImage(u);
```

Afficher une image 3

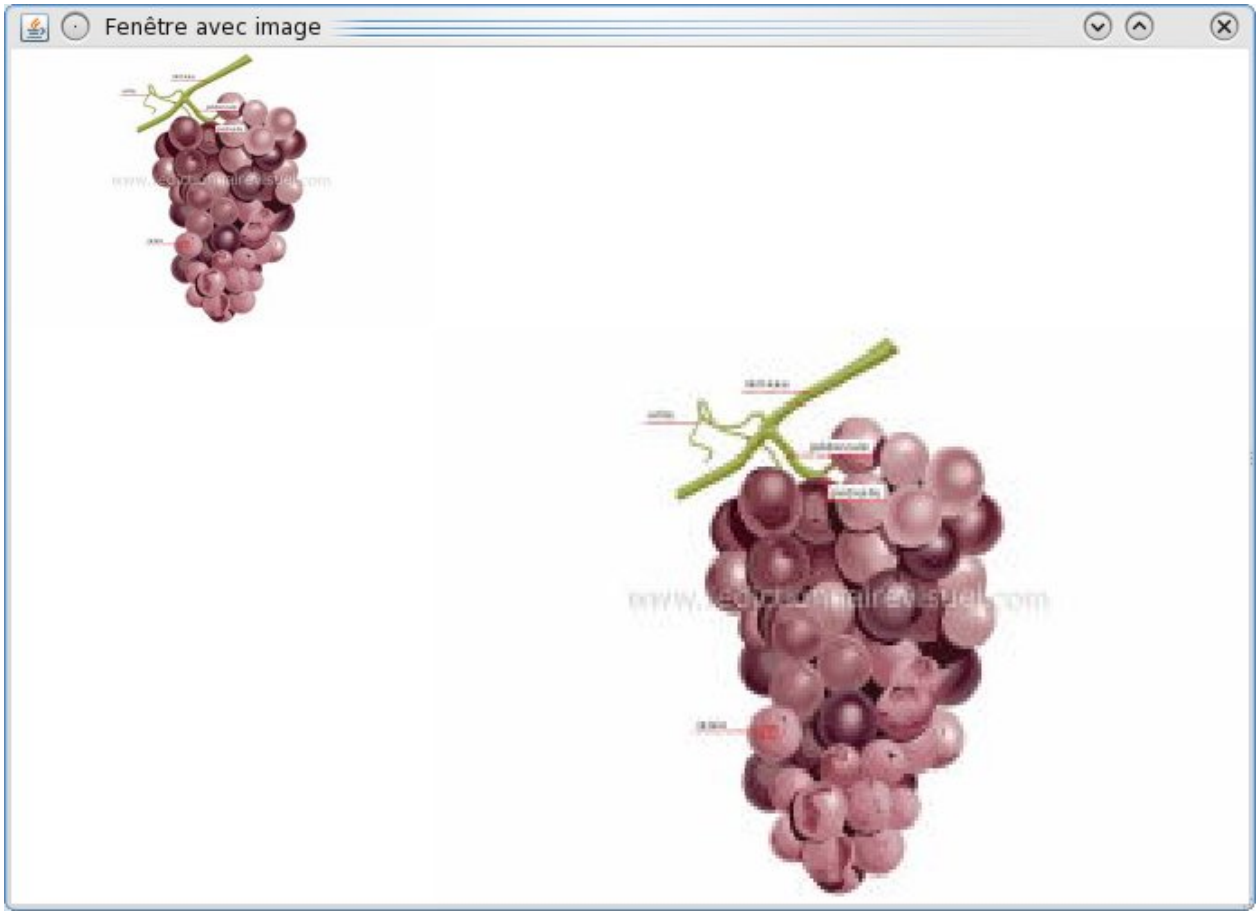
Utiliser la méthode drawImage de la classe Graphics

```
public class PanelImage extends JPanel  {
private Image image;

public PanelImage(String nom)   {
        image=Toolkit.getDefaultToolkit().getImage(nom);
   }
public void paintComponent(Graphics g) {
               super. paintComponent(g);
             g.drawImage(image, 0, 0, this);
}
```

Affichage de l'image avec déformation éventuelle drawImage(Image img, int x, int y, int width, int height,ImageObserver observer)

- affiche l'image adaptée au rectangle indiqué
- l'image est mise à la bonne échelle



Programme

```
class PanelImageEchelle extends JPanel {
  private Image im;

  public PanelImageTriple(String nomImage)  {
      im = Toolkit.getDefaultToolkit().getImage(nomImage);
     this.setBackground(Color.white);
  }
  public void paintComponent(Graphics g){
    super.paintComponent(g);
    int hImage= im.getHeight(this);
    int lImage = im.getWidth(this);
    int hPanel= this.getHeight();
    int lPanel = this.getWidth();

     g.drawImage(im, 0, 0, this); //85x62 image

     g.drawImage(im, lImage, hImage, lPanel-lImage, hPanel-hImage, this);
        }
```

Le tableau de panneaux

La classe JTabbedPane permet de créer un tableau de panneaux. Chaque panneau est accessible par un onglet.

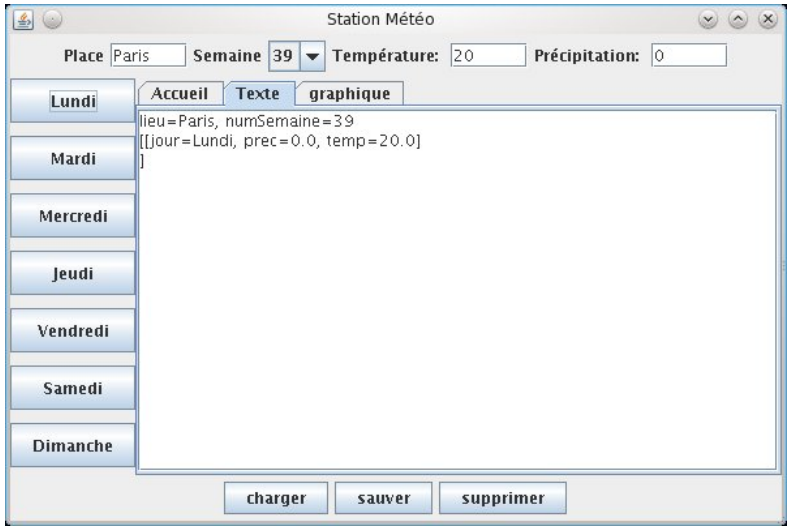


FIGURE : tableau de panneaux

Constructeurs

`JTabbedPane()`	Crée un panneau à onglets, les onglets sont placés en haut.
`JTabbedPane(int tabPlacement)`	Crée un panneau à onglets, les onglets sont placés : • en haut si *tabPlacement* vaut *JTabbedPane.TOP* • en bas si *tabPlacement* vaut *JTabbedPane.BOTTOM* • à gauche si *tabPlacement* vaut *JTabbedPane.LEFT* • à droite si *tabPlacement* vaut *JTabbedPane.RIGHT*
`JTabbedPane(int tabPlacement, int tabLayoutPolicy)`	La stratégie de placement (quand ils ne tiennent pas sur une ligne) des onglets est : • *JTabbedPane.WRAP_TAB_LAYOUT* : Les onglets passent à la ligne. • *JTabbedPane.SCROLL_TAB_LAYOUT* : une barre de défilement apparait.

FIGURE : Les constructeurs de la classe JTabbedPane

Une méthode

La méthode public void add(Component c, Object contrainte)
Ajoute un nouveau composant en tant qu'onglet, ayant pour titre le nom du composant obtenu par getName(). Si contrainte est une chaîne de caractère ou une icône, elle sert de titre.
Exemple

```
JTabbedPane jTabPane = new JTabbedPane() ;
JPanel panAccueil = new JPanel() ;
jTabPane.add(panAccueil,"Accueil") ;
```