UNIVERSITÉ PARIS-SUD 11

Base de programmation Objet en JAVA. 4ème partie.

Frédéric Vernier
(Université Paris-Sud / LRI / LIMSI-CNRS)
Frederic.Vernier,@limsi.fr

Ce cours reprend en grande partie le matériel pédagogique mis au point par Jérôme Nobécourt, Christian Jacquemin et Claude Barras pour l'enseignement de Java en 2001 et 2002 en FIIFO et celui trouvé sur Internet
(http://www.laltruiste.com/, etc.)

Plan 7: Swing

1. Interfaces Graphiques
2. Et la lumière fût …
3. IHM
4. AWT
5. Swing
6. Composant Graphique
7. Ex: JTextArea & JButton
8. Mais où est le conteneur ?
9. Composants
10. Exemples simples
11. Exemples moins simples
12. Exemples costauds
13. Questions
14. LayoutManager mise en pl.
15. BorderLayout
16. LayoutManager
17. Autres LayoutManager
18. Panes (Layered, Scroll,…)
19. JSplitPane et JTabbedPane
20. Et les fenêtres (JFrame)
21. Fenêtres spéciales
22. Barre de menu (détail)
23. Exemple (source 1-4)
24. JMenuBar
25. JPopupMenu &JToolTip

Interfaces Graphiques

- AKA Interfaces Hommes Machines (IHM)
  - Changement de monnaie : 1 Pixels = 6,57 chars
    - C'est plus cher en CPU
  - Le clavier est épaulé d'un mulot
  - Changement de paradigme
    - Avant : shell unix + base de donnée SQL
    - Après : windows + word et excel
  - Changement de programmation
    - Avant : lex, yacc, regexp => language de commande
    - Après : bouton**s**, fenêtre**s**, Menu**s**

Et la lumière fut …

- Apple démocratise le truc
- X-windows professionnalise le concept
- Microsoft mondialise la chose
- Standard de fait
  - Métaphore du bureau (corbeille, placement spatial des documents)
  - Lumière en haut à gauche => relief à l'interface
  - Eléments d'interface indépendants des applis
  - Souris = No1, clavier = texte ou raccourcis

IHM

- Spécialité de l'informatique
  - Liens avec les sciences sociales et l'ergonomie
  - Liens avec le design et le graphisme
- Application emblématiques
  - MacOS, Windows, Word, Excel, Photoshop, Gimp Netscape, IE, etc.
  - Exemples conscients ou inconscients
  - Bibliothèque orientée objet (OO) pour réaliser le même genre d'application
  - 1 bibliothèque par OS graphique (voire plus)

AWT

- Sun lance Java
  - Réalise que le fait d'avoir le même langage sur plusieurs plate-forme ne garantie plus le portage
  - Propose une bibliothèque standard d'objets d'interface multi-plate forme avec JAVA
    - MacOS, Unix, Window et Web browser
  - Succès malgré des limitations graves
    - Lenteur
    - Inconsistances
    - Manque d'éléments
- Sun se dit qu'il aurait pu faire mieux …

Swing

- Sun retire de AWT les éléments graphiques pour ne laisser que les mécanismes bas niveau (fenêtres, composants re-dessinables) et optimise AWT
- Sun rajoute une autre bibliothèque 100% java de composants graphiques utilisant AWT
  - Plus de composants
  - Nouveau modèle d'événement
  - Mécanismes pour étendre la bibliothèque et pour permettre aux IDE de manipuler des SWING

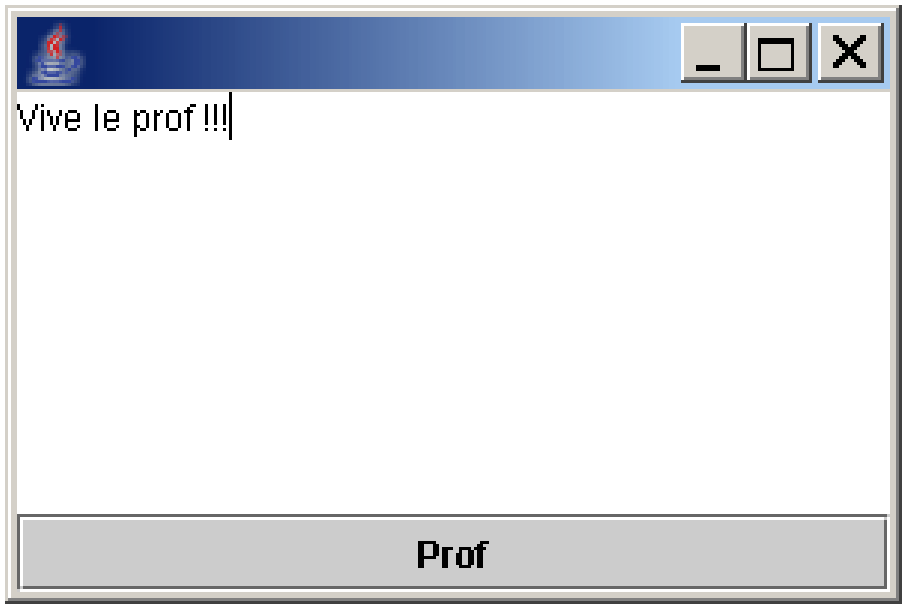
Composants graphiques

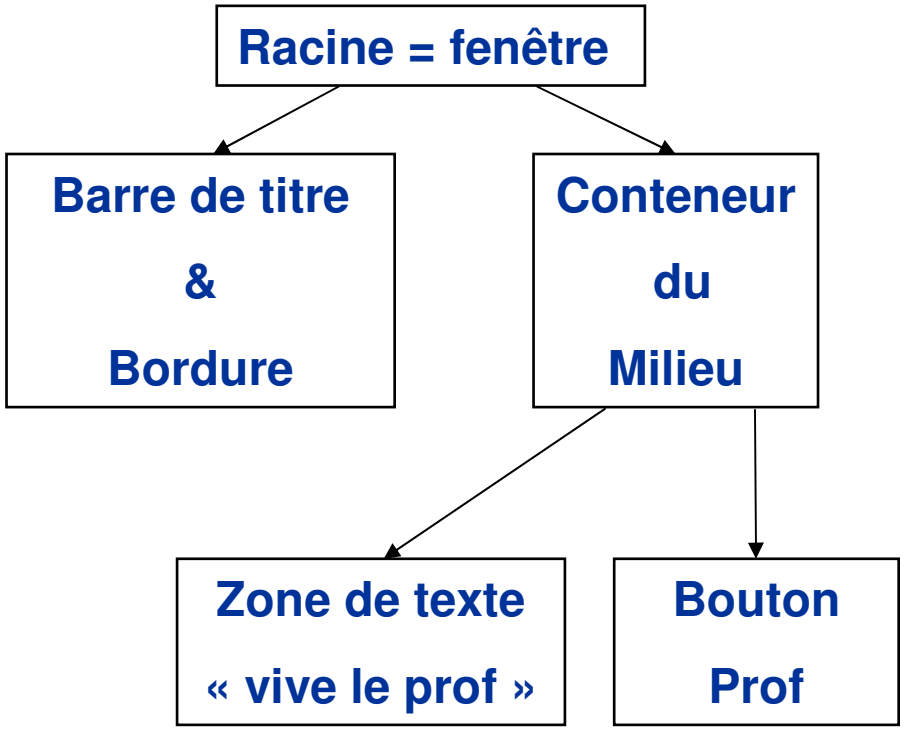
- Gestions de l'affichage de l'interface graphique utilisateur (GUI)
  - Ne redessiner que quand c'est nécessaire
  - Ne redessiner que la zone qui est nécessaire
- L'interface graphique utilisateur est une hiérarchie de composants graphiques
  - java.awt.Component = feuille
  - java.awt.Container = Nœud
- Un Container hérite de Component !!!!!!!!!!!!!!!!!!!
  - Nœud et Feuille ont une partie commune (dessin, événements)
  - Nœud gére en plus une liste de fils à graphiquement à l'écran
- Ne pas confondre hiérarchie d'héritage (container est fils de component) et hiérarchie SWING (component est fils de container)

Exemple : JTextArea et JButton

```
import java.awt.*;
import javax.swing.*;

public class TestTextArea extends JFrame {
  /**
   * cree une fenetre telle que l'objet TestTextArea dans lequel se trouve aussi
   * le main
  */
  public static void main(String[] ags) {
    TestTextArea testTextArea1 = new TestTextArea();
    testTextArea1.setSize (300, 200);
    testTextArea1.show();
  }// main()

  public TestTextArea() {
    // construit la fenetre
    super();

    // rajoute un text area dans la zone centrale
    getContentPane().add(BorderLayout.CENTER, new JTextArea());
    getContentPane().add(BorderLayout.SOUTH, new JButton("Prof"));
  }
}
```

Hiérarchie


Vive le prof !!!
Prof


Racine = fenêtre
Barre de titre & Bordure
Conteneur du Milieu
Zone de texte « vive le prof »
Bouton Prof

Annonces

- Partiel la semaine prochaine
- Polycopié à la repro.
- Contrôle continu à la séance 8 de TD
- Sujet de TER : Le jeu de Carcasonne
- Liste, Pile & Arbre binaire en Java

http://www.lri.fr/~chatalic/licence_pac/index.xhtml

Mais où est passé le conteneur ?

- Il est créé automatiquement lorsque vous créer la fenêtre
  - On n'a a pas accès car c'est une variable encapsulé
  - On peut obtenir sa référence :
    - MaFenetre.getContentPane()
  - Si il ne vous plaît pas :
    - **setContentPane**(Container contentPane)
- On peut obtenir le supérieur d'un composant
  - public Container **getParent**()
- La racine de la hiérarchie est la fenêtre principale de l'application

Composants

- Un composant graphique est un objet gérant
  - une zone rectangulaire de pixels
  - son affichage
  - son comportement vis-à-vis des actions de l'utilisateur
- Le dessin est géré par la fonction
  - `paint(Graphics Graphics_Arg) {...}`
  - Le reste du programme peut appeler la fonction repaint() sur l'instance d'un composant graphique.
  - repaint appellera paint(…)
- Un composant peut être transparent par endroit

Exemples simples

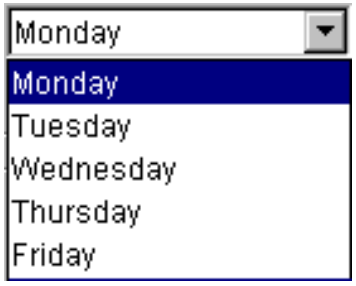
JLabel (avec ou sans image)

JButton

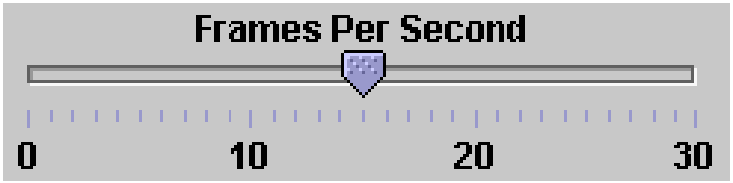
JTextField (+JPasswordField + JFormatedTextField)

UNIVERSITÉ PARIS-SUD 11

Exemples moins simples

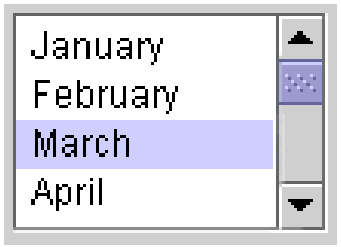


JComboBox



JSlider

JSpinner



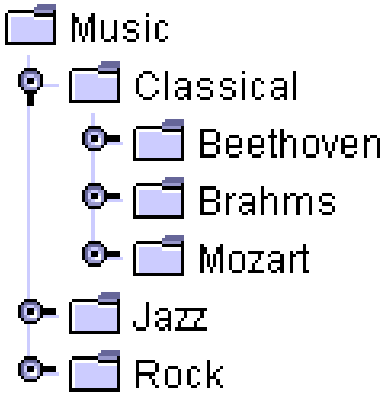
JList

JProgressBar

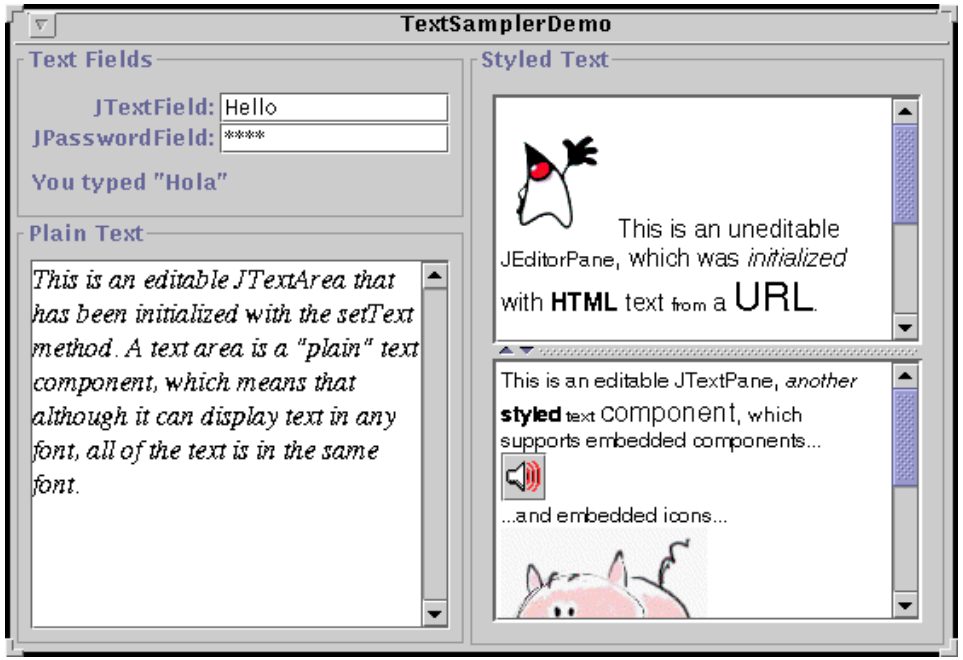
Exemples costauds !

First Name	Last Name	Favorite Food
Jeff	Dinkins	
Ewan	Dinkins	
Amy	Fowler	
Hania	Gajewska	
David	Geary	

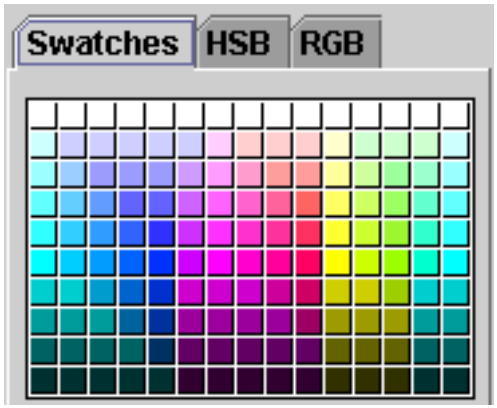
JTable



JTree

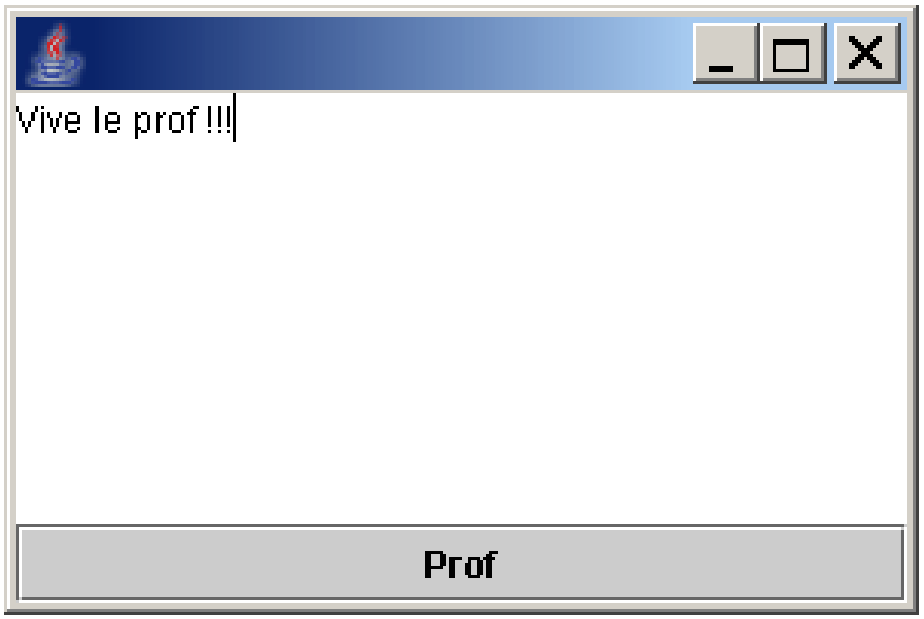


JEditorPane (HTML, rtf)



JColorChooser

On va pouvoir faire mieux que ça…



Mais ça n'explique pas tout !

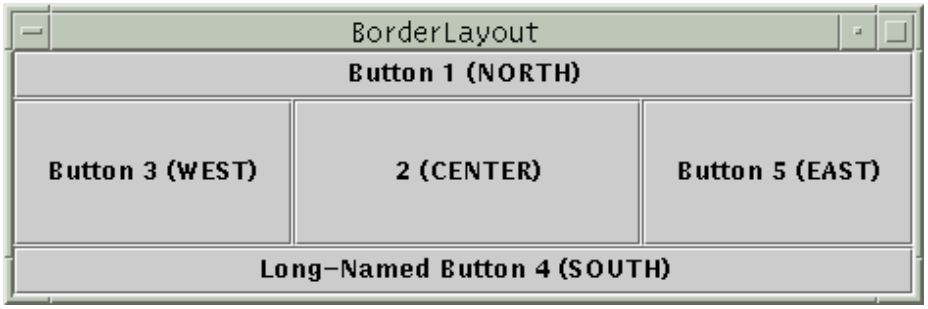
- Comment le bouton se retrouve en bas ?
- Comment le bouton prend moins de place que la zone de texte ?
- Que se passe t-il quand on re-taille la fenêtre?
- Et si l'utilisateur écrit 50 lignes de texte ?
- Et si on n'avait pas spécifié la taille de la fenêtre ?
- QUI GÈRE LA TAILLE ET LA POSITION ?

LayoutManager : Mise en place

- Les containers (qui sont d'honnêtes instances) ont besoin des services d'un collègue qui gère la taille et la position des fils
- Un Container propose :
  - LayoutManager **getLayout**()
  - void **setLayout**(LayoutManager mgr)
- On peut donc complètement gérer le layout manager à travers le container
- Le layout manager par défaut de la zone par défaut est un BorderLayout

BorderLayout

- Cinq Fils
  - Nord
  - Est
  - Sud
  - Ouest
  - Centre



- Nord prend toute la largeur disponible (getWidth()) mais juste sa hauteur préférée (getPreferredSize())
- … ainsi de suite et Center prend tout ce qui reste

```
container.add(myComponent, BorderLayout.NORTH);
```

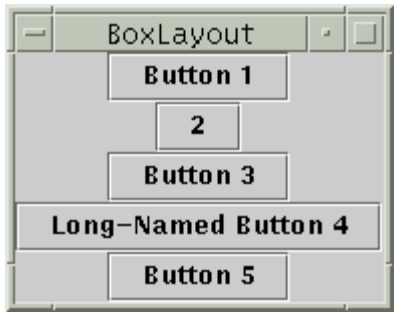
Layout Manager

- Gère la zone rectangulaire de son container
  - Récupère les attributs des fils du container
    - méthodes add() du container

```
Component add(Component comp)
Component add(Component comp, int index)
void add(Component comp, Object constraints)
void add(Component comp, Object constraints, int index)
```

  - Attribue une sous zone à chaque fils
    - Selon : MaximumSize, MinimumSize, PreferredSize
  - Gère le retaillage
  - Gère l'ajout et la suppression de composants

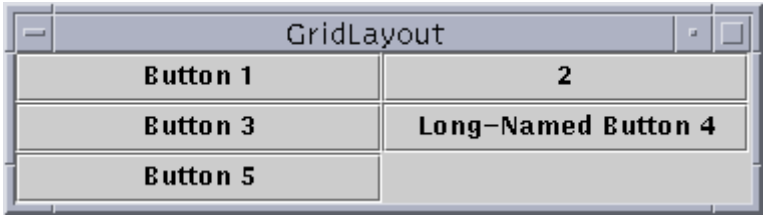
Autres LayoutManager



null

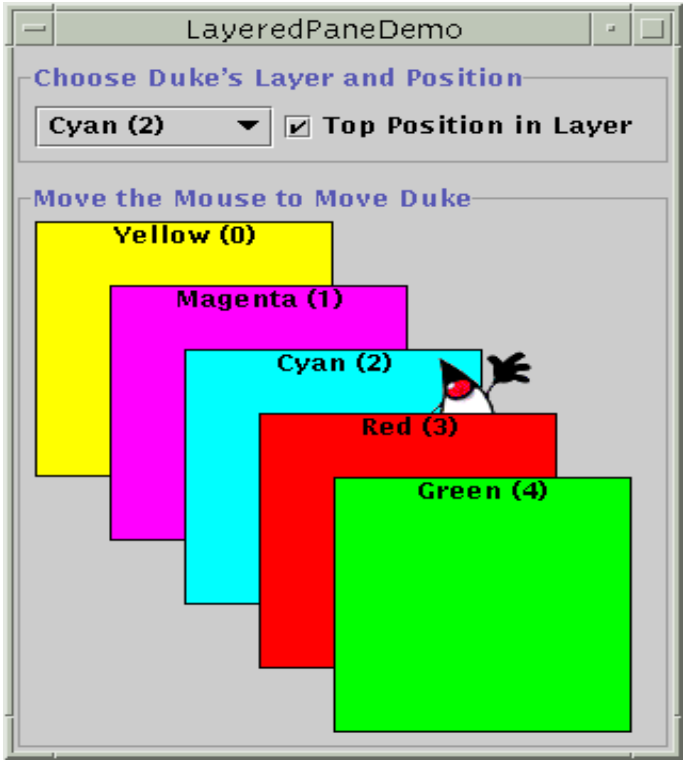
⇒setLocation(x, y)

⇒setSize(w, h)



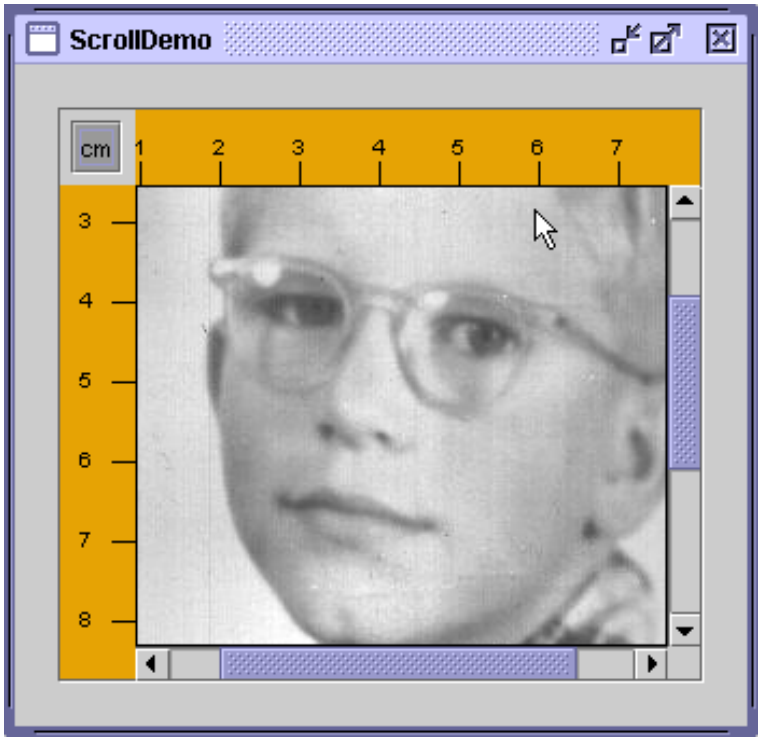
Et c'est tout ?

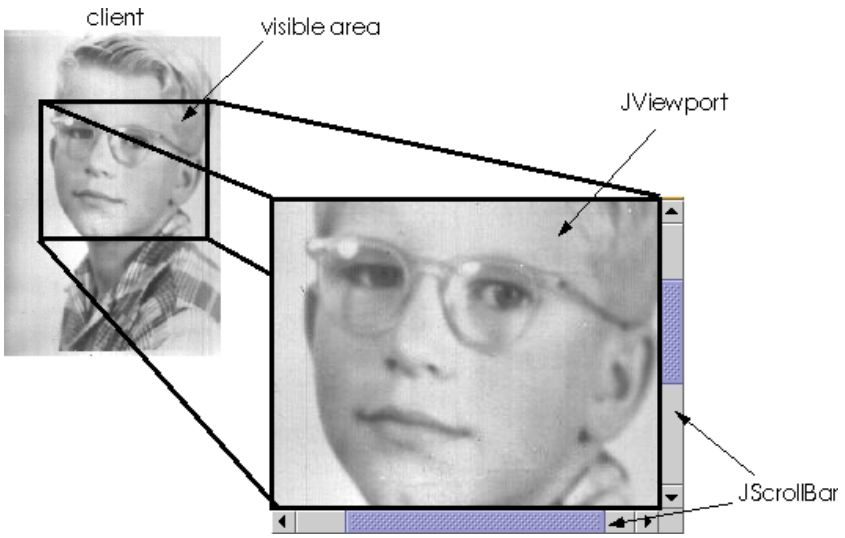
- LayoutManager = aider un container (invisible par défaut)
- Plus haut niveau : un container avec un bout d'interface qui gère ses fils
  - JLayeredPanes



JScrollPane

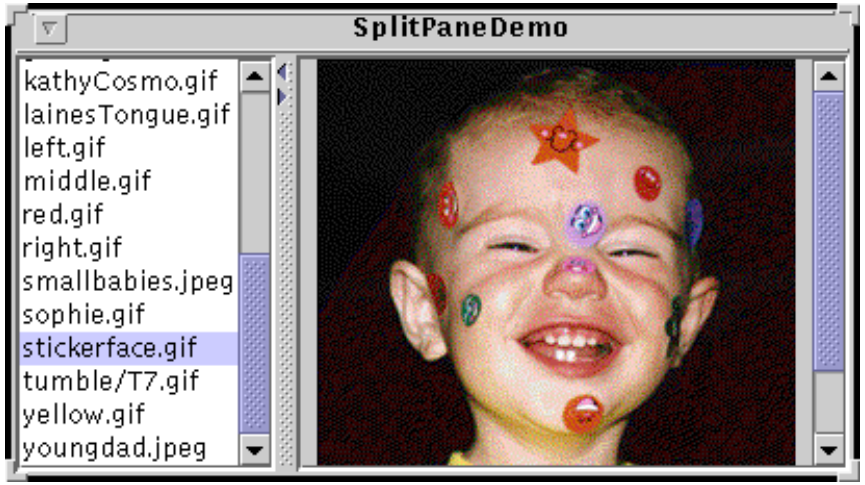
- Tailles dedans / dehors décorrélées

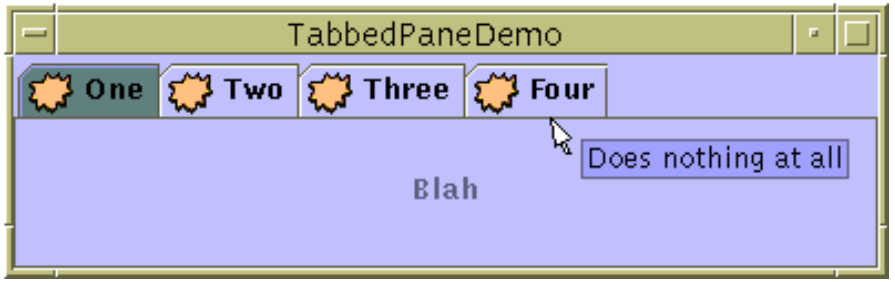




JSplitPane et JTabbedPane

- JSplitPane (Horizontal ou vertical) déplaçable



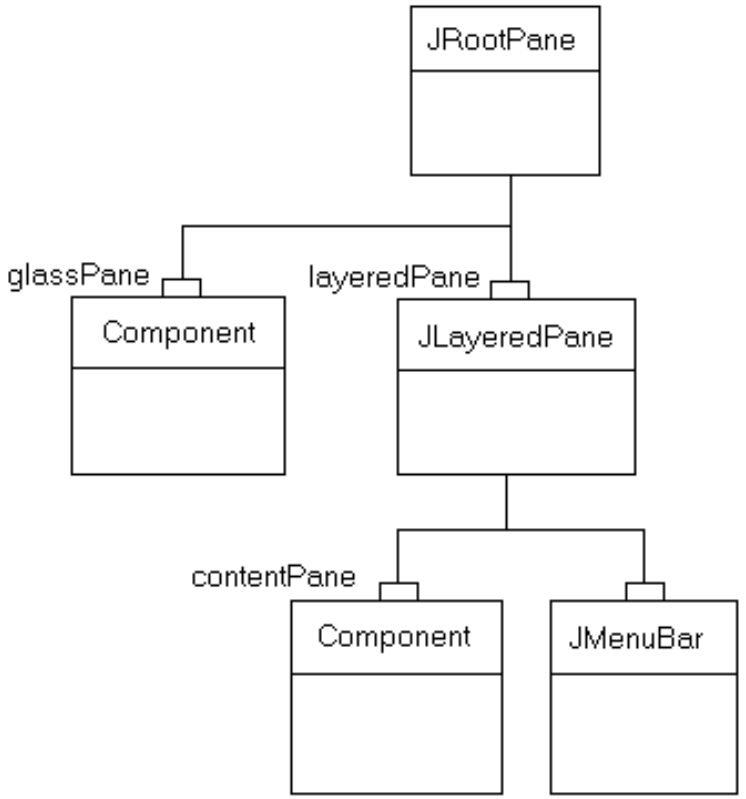


- JTabbedPane = comme CardLayout avec onglets visibles

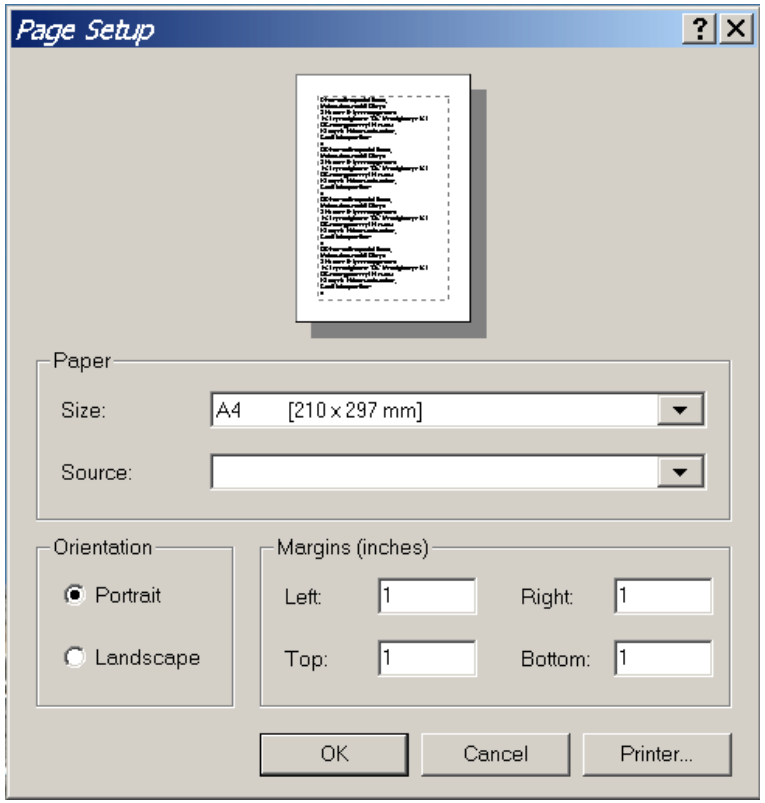
- JToolBar (groupe de boutons) magnétique

Et les fenêtres (JFrame)?

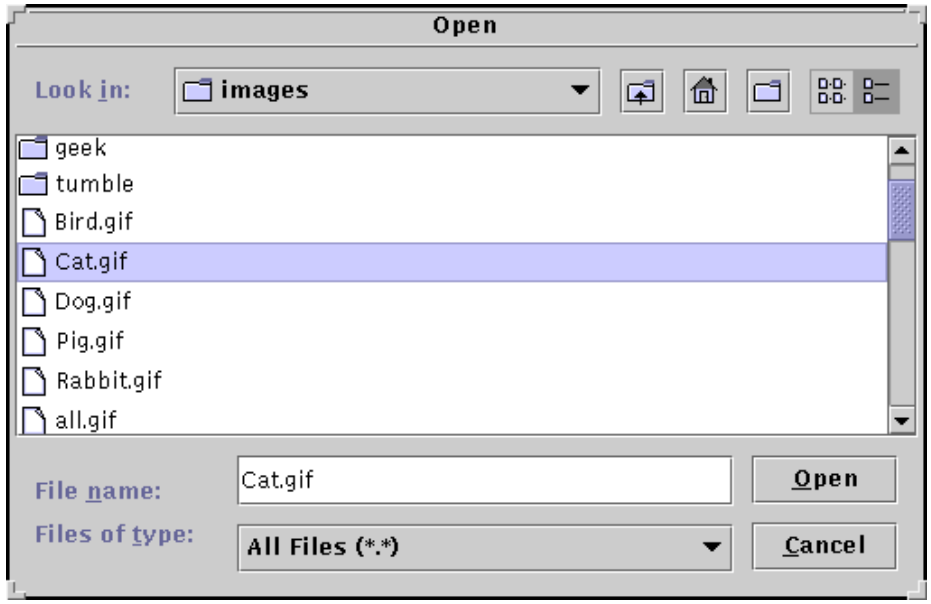
- Traitement spécial
  - Taille et position gérées par le programmeur
  - Position : en haut à gauche sur l'écran 1 par def.
- Cascade de Panes déjà prêts
  - ContentPane : pour vous !
  - GlassPane (tooltips, popup menu)
  - JMenuBar : un et un seul
- JWindow = JFrame sans bords
- Frame, Window = version AWT
  - Niveau plus bas
- Sauf fenêtres spéciales …



Fenêtres spéciales




```
PrinterJob job =
PrinterJob.getPrinterJob();
PageFormat pf =
job.pageDialog(job.defaultPage());
```



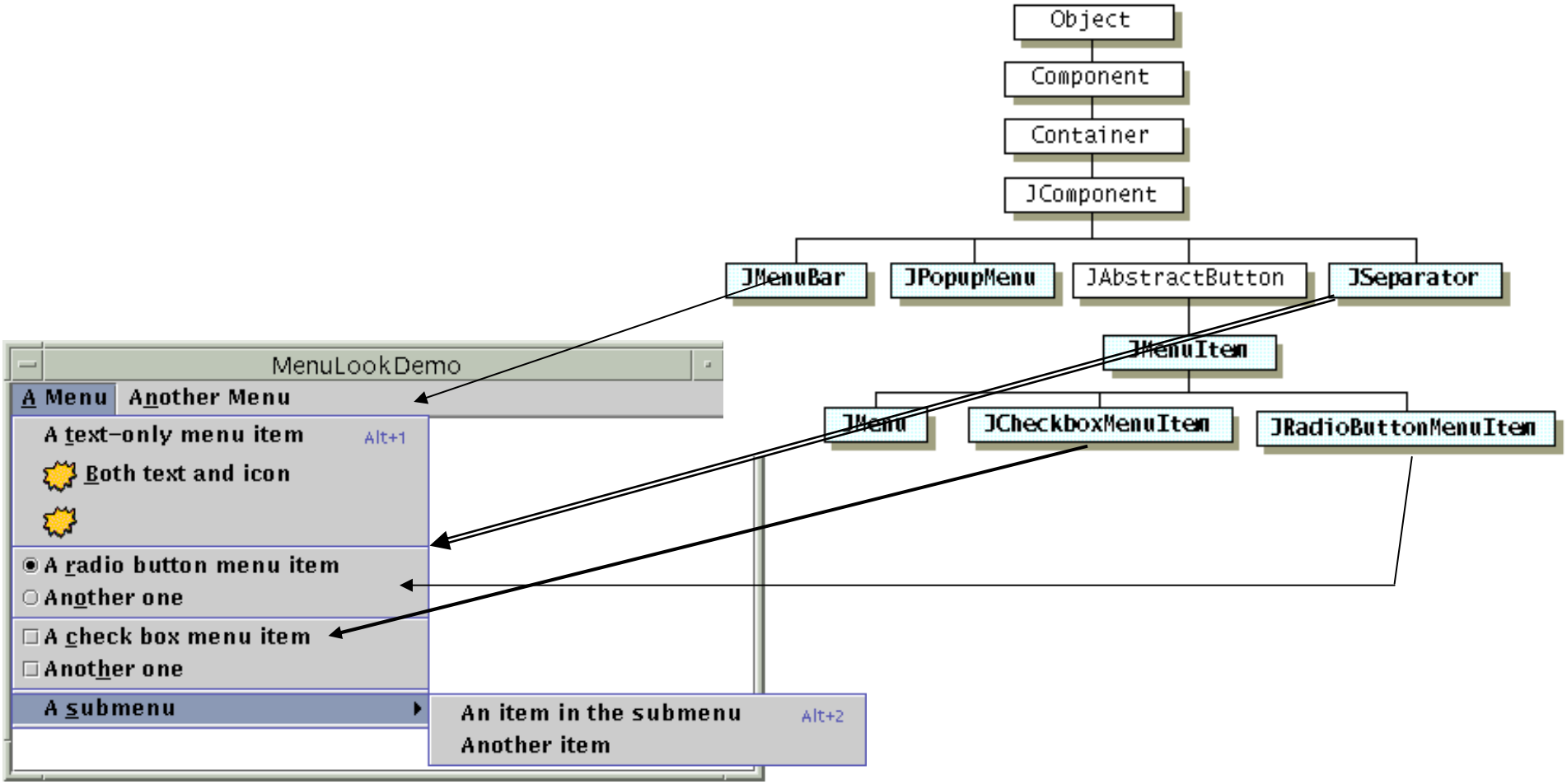
JFileChooser (open ou save)

JDialog (JOptionPane=Modal)

Icons + Yes/No

Barre de Menu en détail

JMenuBar → JMenu(File, Help, etc) → JMenuItem (open, exit)



UNIVERSITÉ PARIS-SUD 11

Source Menu 1/4

```
//Il était une fois dans le constructeur d'une sous classe de JFrame…
JMenuBar menuBar;
JMenu menu, submenu;
JMenuItem menuItem;
JCheckBoxMenuItem cbMenuItem;
JRadioButtonMenuItem rbMenuItem;
...
//Crée la barre de menu
menuBar = new JMenuBar();
setJMenuBar(menuBar);

//Construit le premier menu
menu = new JMenu("A Menu");
menu.setMnemonic(KeyEvent.VK_A);
menu.getAccessibleContext().setAccessibleDescription(
        "The only menu in this program that has menu items");
menuBar.add(menu);
//a group of JMenuItems
menuItem = new JMenuItem("A text-only menu item", KeyEvent.VK_T);
menuItem.setAccelerator(KeyStroke.getKeyStroke(
                        KeyEvent.VK_1, ActionEvent.ALT_MASK));
```

Source Menu 2/4

```
menuItem.getAccessibleContext().setAccessibleDescription(
        "This doesn't really do anything");
menu.add(menuItem);
menuItem = new JMenuItem("Both text and icon",
                         new ImageIcon("images/middle.gif"));
menuItem.setMnemonic(KeyEvent.VK_B);
menu.add(menuItem);

menuItem = new JMenuItem(new ImageIcon("images/middle.gif"));
menuItem.setMnemonic(KeyEvent.VK_D);
menu.add(menuItem);

//un groupe de bouton radio (reliés entre eux !)
menu.addSeparator();
ButtonGroup group = new ButtonGroup();

rbMenuItem = new JRadioButtonMenuItem("A radio button menu ");
rbMenuItem.setSelected(true);
rbMenuItem.setMnemonic(KeyEvent.VK_R);
group.add(rbMenuItem);
menu.add(rbMenuItem);
```

Source Menu 3/4

```
rbMenuItem = new JRadioButtonMenuItem("Another one");
rbMenuItem.setMnemonic(KeyEvent.VK_O);
group.add(rbMenuItem);
menu.add(rbMenuItem);

//Un groupe de checkboxes (boites à cocher)
menu.addSeparator();
cbMenuItem = new JCheckBoxMenuItem("A check box menu item");
cbMenuItem.setMnemonic(KeyEvent.VK_C);
menu.add(cbMenuItem);

cbMenuItem = new JCheckBoxMenuItem("Another one");
cbMenuItem.setMnemonic(KeyEvent.VK_H);
menu.add(cbMenuItem);

//un sous menu
menu.addSeparator();
submenu = new JMenu("A submenu");
submenu.setMnemonic(KeyEvent.VK_S);
```

UNIVERSITÉ PARIS-SUD 11

Source Menu 4/4

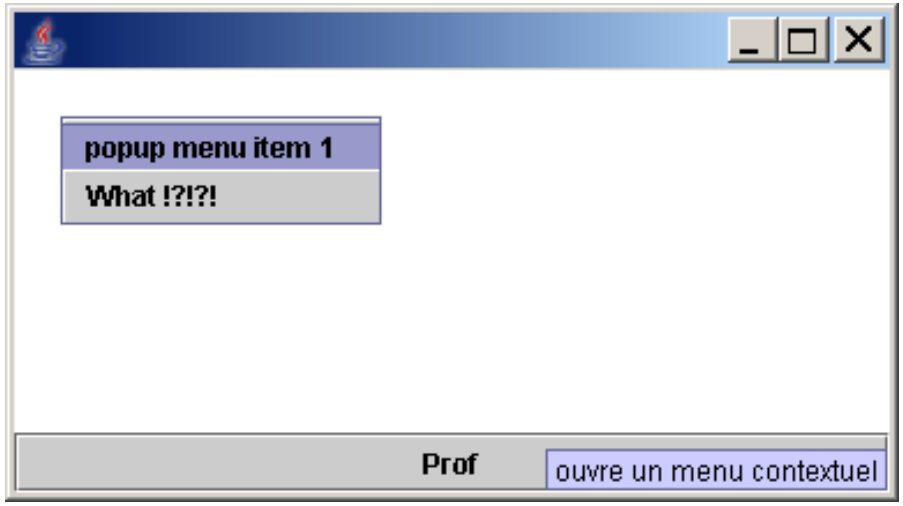
```
menuItem = new JMenuItem("An item in the submenu");
menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_2, ActionEvent.ALT_MASK));
submenu.add(menuItem);

menuItem = new JMenuItem("Another item");
submenu.add(menuItem);
menu.add(submenu);

//Construit un second menu dans la barre de menu
menu = new JMenu("Another Menu");
menu.setMnemonic(KeyEvent.VK_N);
menu.getAccessibleContext().setAccessibleDescription(
        "This menu does nothing");
menuBar.add(menu);
```

JMenuBar

- Mnemonic = naviguer au clavier
- Accelerator = raccourci clavier
- AccessibleDescription = handicap
- ButtonGroup (instance) pour relier des radio buttons
- JMenuBar-JMenu-JMenuItem(ou dérivés)
  - JMenuBar-JMenu-JMenu-JMenuItem (sous-menu)
- Ordre d'ajout (add) important (sinon insert()…)
- Semaine prochaine : action !

JPopupMenu et JToolTip

```
//Crée un menu contextuel
JPopupMenu jPopupMenu1 = new JPopupMenu();

//Comme un JMenuBar !
menuItem = new JMenuItem("popup menu item 1");
jPopupMenu1.add(menuItem);
menuItem = new JMenuItem("What !?!?!");
jPopupMenu1.add(menuItem);

//Affiche par-dessus jTextArea1 avec un certain décalage
 jPopupMenu1.show(jTextArea1, 20, 20);

 jButton1.setToolTipText(
   "ouvre un menu contextuel");
```



Plan 8: Dessin et Evénements

1. Composant graphique sur mesure
2. Void paint(Graphics G_Arg) {
3. Fonctions de dessin avec Graphics
4. Dessin avec Graphics2D
5. Fonctions de dessin de forme
6. Fonctions de dessin de texte
7. Fonctions de dessin d'image
8. Evénements
9. Exemple simple : Bouton
10. Objet d'événements
11. Evénements souris
12. Evénements souris rapide
13. Fenêtre (ou panel) écouteur
14. Classes emboîtées
15. Conseils

Mon propre composant graphique

- Hérite de Component, JComponent ou **JPanel**
- Re-Implante la fonction
  - `void paint (Graphics G_Arg) {...}`
- Hérite de repaint() pour lancer paint(…)
  - Asynchrone, gestion automatique du Graphics
- Décide de sa stratégie : setDoubleBuffered(b)
- Hérite de méthodes externes à tenir compte
  - setSize(), setEnable(), setBackground(Col), setFont(), setForeground(), etc.

void paint (Graphics G_Arg) {

- Une instance de Graphics est donné par java pour ce composant afin de dessiner
- Un Graphics possède un état :
  - Translation à l'origine pour le rendu :translate()
    - 0,0 = coin haut gauche par défaut
  - Zone de d'effet (!= rectangulaire) = Clip
    - Par défaut : tout, mais on peut se restreindre
  - Couleur de dessin
    - Color col1 = new Color (255, 0, 0); RGB mais aussi HSB
  - Police de caractère
    - Font font1 = new Font("SansSerif", Font.BOLD, 12);

Fonctions de dessin avec Graphics

- Exemple : public void drawLine (x1, y1, x2, y2)
  - Dépend de la couleur courante
- fillRect() / drawRect()=remplissage ou contour
  - Rect, Oval, String, Arc, Polygon, PolyLine
- Fonction clear() pour nettoyer
- Une fonction FontMetrics getFontMetrics()
  - Renvoi une instance qui mesure le texte (lon/lar)
- Fonction drawImage() pour le dessin d'image
  - Nécessite une instance de "Image"
  - Asynchrone. Possibilité d'écoute : ImageObserver

Dessin avec Graphics2D

- Fonction paint(Graphics) appelé par Java
  - Mais Graphics = Graphics2D sur version récente !
  - Transtypage
- Etat de dessin plus élaboré
  - Paint : peinture (Color, GradientPaint ou TexturePaint)
  - Font : police de caractère
  - Clip : zone de restriction du dessin
  - Stroke : pinceau = forme, épaisseur (1p), joins aux angles
  - Transform : Matrice affine de transformation
    - Translation, rotation, zoom, penchant
  - Composite : règle de superposition d'un pixel de couleur avec un autre
  - Liste de RenderingHint définissant la qualité de rendu

Fonctions de dessin de forme

- Interface « Shape » implémentée par
  - Area, CubicCurve2D, GeneralPath, Line2D, Polygon, QuadCurve2D, Rectangle, RectangularShape (Arc2D, Ellipse2D, Rectangle2D, RoundRectangle2D)
  - PathIterator : itérateur sur les segments
  - Fonction getBounds() pour le contour rectangulaire
  - Lissage = Antialiasing (beau mais lent)
    - `setRenderingHint(KEY_ANTIALIASING, VALUE_ANTIALIAS_ON);`

Fonctions de dessin de texte

- Ecrit dans tous les sens (rotation de texte, langage non occidental)
- La chaîne de caractère est transformée en itérateur de caractère
- Les caractères sont transformés en Shape
- Les formes (Shape) dessinés avec Paint
  - Plein / vide
  - Lissage (anti-alias)

```
setRenderingHint(KEY_TEXT_ANTIALIASING, VALUE_TEXT_ANTIALIAS_ON);
```

  - Gradient
  - Texte incurvé …

Fonctions de dessin d'image

- AffineTransform => rotation, mise à l'échelle
- Composite : Transparence

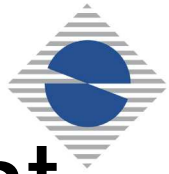
```
setRenderingHint(KEY_INTERPOLATION, VALUE_INTERPOLATION_BILINEAR);
```

Evénements

- Comment capturer les actions de l'utilisateur et activer les bonnes actions correspondantes?
  - Boucle d'événement : while () {switch(){}}
  - Callback : une fonction avec un nom précis
  - Abonnement : instance écouteuse
    + Encapsulation de callbacks
    + Écouteurs multiples (répartition du travail)
    - Beaucoup de classes à coder (1 par écouteur)\
  - Concrètement un écouteur c'est :
    - 1 Interface à respecter
- 2 niveaux
  - élément graphique spécifique = Haut niveau (Bouton)
  - élément graphique générique = Bas niveau (Component)

Exemple simple : bouton

```
JButton        jButton1 = new JButton("prof");

//-----------------1ere possibilité = Haut niveau --------------------
EcoutBoutHello ebh1     = new EcoutBoutHello ();
jButton1.addActionListener(ebh1);
...
class EcoutBoutHello implements ActionListener {   // pas public
  public void actionPerformed(ActionEvent ae1){
    System.out.println("Hello");
  }
}
//---------------------- OU : Bas niveau ----------------------------
EcoutBoutBye ebb1     = new EcoutBoutBye();
jButton1.addMouseListener(ebb1);
...
class EcoutBoutBye extends MouseAdapter {    // pas public
  public void mousePressed(MouseEvent me1){
    System.out.println("Bye Bye");
  }
}
```



Objet d'événement

- Les callbacks donnent au programmeur un argument qui décrit l'événement

```
JButton     jButton1   = new JButton("prof");
JMenuItem   JMenuItem1 = new JMenuItem("etudiant");

EcoutBoutHello ebh1     = new EcoutBoutHello ();
jButton1.addActionListener(ebh1);
jButton2.addActionListener(ebh1);

...

class EcoutBoutHello implements ActionListener {   // pas public
  public void actionPerformed(ActionEvent ae1){
    //if (ae1.getSource()==jButton1) ne marche pas car jButton1 pas ici
    if (ae1.getSource() instanceof JButton)
      System.out.println("Hello bouton");
    else ...
  }
}
```

- getSource() donne l'élément d'interface source de l'événement

Evénement Souris

```
EcoutBoutBye ebb1     = new EcoutBoutBye();
jButton1.addMouseListener(ebb1);
...
class EcoutBoutBye extends MouseAdapter { // pas public
  public void mousePressed(MouseEvent me1){
    if (me1.getClickCount()==2 && me1.getX()>100)
      System.out.println("Bye Bye mulot !");
  }
}
```

- getClickCount() = nombre de clicks successifs
- getX() et getY() renvoie la position du pointeur dans l'élément
- Une action c'est une action, un mulot ça peut être :
  - void mouseEntered(MouseEvent e) //entre au dessus de l'élément
  - void mouseExited(MouseEvent e)
  - void mousePressed(MouseEvent e)
  - void mouseReleased(MouseEvent e)
  - void mouseClicked(MouseEvent e) (press + release sans bouger)
- Adapteur (classe avec callback vides) ou écouteur (Interface)

Evénements souris rapides

- La souris génère aussi des événements quand elle bouge
  - Quantité plus importante (100/sec)
  - Traitement rapide sinon on va en rater

```
EcoutBoutMove ebb1     = new EcoutBoutMove();
jButton1.addMouseMotionListener(ebb1);
...
class EcoutBoutMove implements MouseMotionListener {
  public void mouseMoved(MouseEvent me1){
    if (me1.getX()>100)
      System.out.println("ca bouge a droite !");
  }
  public void mouseDragged(MouseEvent me1){
    if (me1.getX()<50)
      System.out.println("ca drag a gauche !");
  }
}
```

- Possibilité d'implémenter MouseListener ET MouseMotionListener
- Si traitement trop lent java peut abandonner des MouseMotion et donner tous les clicks/etc.

Fenêtre (ou panel) écouteur

- Technique répandue = callback dure

```
public class MaFenetre extends JFrame implements
   MouseMotionListener, ActionListener {
   JButton jbutton1 = new JButton("prof");
   JTextArea jTextArea1 = new JTextArea();

public MaFenetre {
   super("Ma belle fenetre");
   setLayout(new BorderLayout());
   getContentPane().add(BorderLayout.SOUTH, jButton1);
   getContentPane().add(BorderLayout.CENTER, jTextArea1);
   jButton1.addActionListener(this);
   jTextArea1.addMouseMotionListener(this);
}
```

Fenêtre (ou panel) écouteur

```
public void mouseDragged(MouseEvent me1){
  if (me1.getX()<50)
    System.out.println("ca drag a gauche !");
}

public void mouseMoved(MouseEvent me1){
  if (me1.getX()>100)
    System.out.println("ca bouge a droite !");
}

public void actionPerformed(ActionEvent ae1){
  // if (ae1.getSource() instanceof JButton) marche aussi
  if (ae1.getSource()==jButton1)
    System.out.println("Hello bouton numero 1");
}
```

Attention !

- Si 2 boutons (ou MenuItem) => même callback pour les 2
- Obligation de faire pleins de if ()

Classes emboîtées

- Pas très élégant mais très pratique pour les evts.

```
jButton1.addMouseMotionListener(new MouseMotionListener()
   {
    public void mouseMoved(MouseEvent me1){
      if (me1.getX()>100)
        System.out.println("ca bouge a droite !");
    }
    public void mouseDragged(MouseEvent me1){
      if (me1.getX()<50)
        System.out.println("ca drag a gauche !");
    });
```

Classes emboîtées + appel fct

- Possibilité de n'appeler qu'une seule fonction (callback souple)

```
jButton1.addMouseMotionListener(new MouseMotionListener() {
  public void mouseMoved(MouseEvent me1){myMouseMoved(me1);}
  public void mouseDragged(MouseEvent
   me1){myMouseDragged(me1);}
});

private void myMouseMoved(MouseEvent me1) {
  if (me1.getX()>100)
      System.out.println("ca bouge a droite !");
}

private void myMouseMoved(MouseEvent me1) {
 if (me1.getX()<50)
      System.out.println("ca drag a gauche !");
}
```

Conseils

- Listener (vous devez implémenter toutes les fonctions)
  - Facile de faire un copié-collé de la doc (explicite)
- Adapter (vous implémenter ce dont vous avez besoin)
  - Existe que si plus d'une fonction dans le prototype d'écouteur (Listener)
  - Ne marche pas si 2 adapteurs (ex: technique de la fenêtre)
- Stratégie de codage
  - Chercher les addXXListener() de l'élément graphique que vous utilisez
  - Cliques sur la doc. du Listener, regardez les fonctions …
  - Si vous n'arrivez pas à vous décider, regarder l'instance de Event si il propose bien les services dont vous avez besoin
- Ne recodez pas les raccourcis clavier avec des KeyEvents !
- Technique de la fenêtre écouteur pour les très petits exemples
- Classes emboîtées avec appel de fonction sinon
  - Pas besoin de passer la source à votre fct. (utilisez getSource())
  - Dans les evts. Rapides, appelez repaint() qui est asynchrone !