Cours n°9

Interfaces graphiques II

Sommaire

1. Interaction graphique-utilisateur
    1. Evénements d'interaction
    2. Détection des événements d'interaction
    3. Traitement des événements d'interaction
    4. Exemples
2. Gestion des documents textuels
    1. Interfaces de modélisation
    2. Visualisation et édition
3. Formes géométriques et images
    1. Formes géométriques 2D
    2. Images bitmap

Définitions (1/2)

Action utilisateur

Déclenchement d'un dispositif d'entrée de l'ordinateur
clavier, souris au minimum
entrée vocale (prochaine version de Windows)
joystick, gant 3D

Action clavier

Dynamique (appui ou relâchement d'une touche),
Statique (touche enfoncée ou relâchée),
Combinaison et suite de touches (raccourci clavier)

Action souris

Dynamique,
mouvement de la souris, rotation de la molette
appui ou relâchement d'un bouton (click ou double click)
Statique (bouton enfoncée ou relâchée)

Multiplicité de combinaisons

Difficulté d'apprentissage (définition de chartes d'interactions)

Définitions (2/2)

Interaction utilisateur

Détection par un programme d'une action utilisateur

Interaction graphique-utilisateur

Interaction utilisateur par l'intermédiaire d'un composant graphique

Evénement d'interaction

Objet informatique créé lors d'une interaction utilisateur

Programmation événementielle

Type de programmation
plus complexe pour le programmeur,
plus simple pour l'utilisateur (cf. ergonomie)
Choix de la suite des instructions exécutées en fonction des interactions
Multiples points d'entrée dans le programme
un seul en programmation classique

Evénements en Java

Classe racine EventObjet

Une seule méthode Objet **getSource**()
Référence de l'objet qui a produit l'événement

Plusieurs paquetages de classes dérivées

javax.sql (connexion aux bases de données)
javax.net.ssl (connexion à un terminal distant)
javax.sound.sampled (acquisition-restitution du son)
javax.naming.event (système de gestion de fichiers)
java.util.prefs (modification d'une structure d'arbres)
javax.print.event (impression)
java.awt.AWTEvent (interactions graphiques Awt)
javax.swing.event (interaction avec les composants graphiques Swing)

Evénements d'interactions graphiques AWT

Evénements de bas niveau

MouseEvent (événement souris)
déplacement du curseur et de la molette,
modification des états des boutons

KeyEvent (événement clavier)
touche enfoncée ou relâchée

WindowEvent (événement fenêtre)
ouverture, fermeture,
iconification, déiconification,
modification de focus

FocusEvent (sélection d'un composant graphique)

Evénements de haut niveau

ActionEvent (action sur un composant graphique sensible à une action)

Evénements d'interactions graphiques Swing

Interactions avec les composants atomiques

MenuEvent Choix dans un menu

ItemEvent Choix d'un Item

DocumentEvent Modification d'un document Swing (texte)

HyperlinkEvent Activation d'un lien dans un document Swing (HTML)

Interactions avec les composants de gestion de structures de données

ListSelectionEvent Modification de la sélection d'un Item dans une liste (Jlist)

ListDataEvent Modification du contenu d'un ou plusieurs Item dans une liste (Jlist)

TreeModelEvent Modification de l'arbre (JTree)

TreeSelectionEvent Modification du nœud sélectionné (JTree)

Ecouteur d'événements (event listener)

Objet permettant de détecter un certain type d'événements

MouseListener (Clic de souris, Entrée ou sortie de la souris d'une surface)

MouseMotionListener (Déplacement sur un composant)

KeyListener (Evénement clavier)

WindowsListener (Modification de fenêtre)

ActionListener (clic sur un bouton, retour-chariot dans un champ de saisie)

…

MenuListener (Modification d'un élément dans un menu)

DocumentListener (Modification d'un document Swing)

HyperlinkListener (Activation d'un lien dans un document Swing HTML)

…

Ecouteurs et composants graphiques

Ajout d'un écouteur à un composant graphique

Méthodes addxxxListener(Object ot) avec xxx les différents types d'écouteurs et ot l'objet traitant l'événement

Information sur les interactions utilisateurs avec un composant graphique donnée

JPanel

addComponentListener, addFocusListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener

JButton

addActionListener, addChangeListener

JMenu

addMenuListener

JMenuItem

addMenuDragMouseListener, addMenuKeyListener, addItemListener

Principes

Déclenchement d'une méthode appartenant à l'objet de traitement à la détection d'un événement

Traitement de l'événement dans le composant graphique où l'événement a eu lieu

Ajout d'un écouteur dont le paramètre ot est la valeur *this*

Traitement de l'événement dans un autre objet

Choix d'un composant graphique père (en général)

Centralisation de traitement de plusieurs événements de même type

Recherche de l'objet graphique source de l'événement (getSource)

Programmation événementielle

Méthodes de traitement (bas niveau 1/2)

Méthodes de MouseListener

void **mouseClicked**(MouseEvent e)
déclenchement en cas de clic de souris sur le composant graphique écouteur (CGE)
void **mouseEntered**(MouseEvent e)
déclenchement en cas d'entrée du curseur de la souris dans le CGE
void **mouseExited**(MouseEvent e)
déclenchement en cas de sortie du curseur de la souris du CGE
void **mousePressed**(MouseEvent e)
déclenchement en cas d'appui sur un bouton de la souris du CGE
void **mouseReleased**(MouseEvent e)
déclenchement en cas de relâchement d'un bouton de la souris du CGE

Méthodes de KeyListener

void **keyTyped**(KeyEvent e)
déclenchement en cas de frappe sur une touche
void **keyPressed**(KeyEvent e)
déclenchement en cas d'appui sur une touche
void **keyReleased**(KeyEvent e)
déclenchement en cas de relâchement d'une touche

Méthodes de traitement (bas niveau 2/2)

Méthodes de WindowListener

void **windowActivated**(WindowEvent e)
déclenchement quand la fenêtre devient active (focus)

void **windowDeactivated**(WindowEvent e)
déclenchement quand la fenêtre devient inactive

void **windowOpened**(WindowEvent e)
déclenchement quand la fenêtre devient visible

void **windowClosed**(WindowEvent e)
déclenchement quand la fenêtre est fermée

void **windowIconified**(WindowEvent e)
déclenchement quand la fenêtre est iconifiée

void **windowDeiconified**(WindowEvent e)
déclenchement quand la fenêtre est déiconifiée

Méthodes de traitement (composants atomiques)

Méthodes de ActionListener

void **actionPerformed**(ActionEvent e)
déclenchement en cas d'action sur un composant graphique écouteur

Méthodes de MenuListener

void **menuSelected**(MenuEvent e)
déclenchement en cas de sélection d'un menu

void **menuDeselected**(MenuEvent e)
déclenchement en cas en cas de désélection d'un menu

void **menuCanceled**(MenuEvent e)
déclenchement en cas en cas d'effacement du menu

Méthodes de ItemListener

void **itemStateChanged**(ItemEvent e)
déclenchement en cas de sélection ou de désélection d'un Item

Méthodes de traitement (structure de données)

Méthodes de ListSelectionListener

void **valueChanged**(ListSelectionEvent e)
déclenchement en cas de changement de la sélection

Méthodes de TreeSelectionListener

void **valueChanged**(TreeSelectionEvent e)
déclenchement en cas de changement de la sélection

Méthodes de TreeExpansionListener

void **treeExpanded**(TreeExpansionEvent e)
déclenchement en cas d'agrandissement de la vue

void **treeCollapsed**(TreeExpansionEvent e)
déclenchement en cas de rétrécissement de la vue

Evénement de bas niveau

PanneauDessinSouris.java

```
public class PanneauDessinSouris extends JPanel implements
MouseMotionListener {

private int X,Y;
// ajout d'un écouteur sur le panneau
public PanelDessinSouris() {addMouseMotionListener(this);}
// traitement de tous les événements du panneau de type Mouse Motion
public void mouseMoved(MouseEvent e) { X=e.getX(); Y=e.getY();
repaint(); }

public void mouseDragged(MouseEvent e) {}

public void paintComponent(Graphics g) { g.drawLine(0, 0, X, Y); } }

public static void main(String[] args) {
JFrame fen = new JFrame("Panneau de dessin à la so
fen.setSize(380, 380);
fen.getContentPane().setBackground(Color.CYAN);
PanneauDessinSouris p = new PanneauDessinSouris();
p.setForeground (Color.MAGENTA);
fen.getContentPane().add(p);
fen.setVisible(true);
 }
```

Panneau2Boutons.java

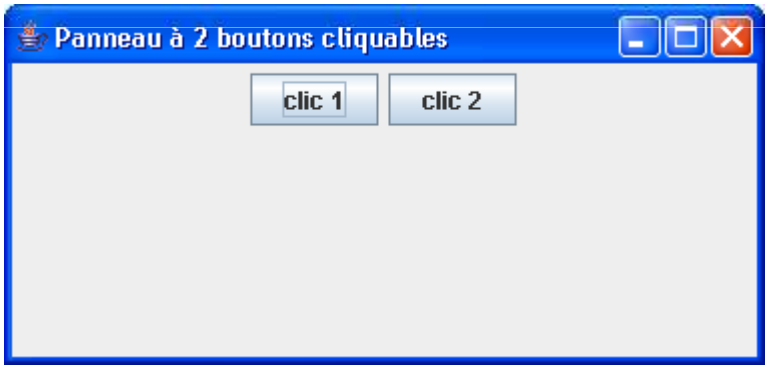
Evénement sur un composant atomique (1/5) - Panneau

```
public class Panneau2Boutons extends JPanel
implements ActionListener {
private JButton bt1, bt2;

// création des boutons
public Panneau2Boutons() {
bt1 = new JButton("clic 1"); this.add(bt1);
bt2 = new JButton("clic 2"); this.add(bt2);

// ajout des écouteurs sur chaque bouton
bt1.addActionListener(this);
bt2.addActionListener(this);
}

// traitement de tous les événements du panneau de type Action
public void actionPerformed(ActionEvent ev) {
if (ev.getSource() == bt1)
System.out.println("clic 1");
else System.out.println("clic 2");
} }
```

testPanneau2Boutons.java

Evénement sur un composant atomique (2/5) - Panneau

```
public static void main(String[] args) {
        JFrame fen = new JFrame("Panneau à 2 boutons");
      fen.setSize(380, 180);

Panneau2Boutons p = new Panneau2Boutons();
fen.getContentPane().add(p);
fen.setVisible(true);
 }
```




```
clic 1
clic 1
clic 2
clic 1
clic 2
```

MenuDetecterChoix.java

Evénement sur un composant atomique (3/5) - Menu

```
public class MenuDétecterChoix extends JMenuBar
implements MenuListener {

private JMenu[] menus = null;
private int nbMenus;

public MenuDétecterChoix(String[] nomMenus) {
nbMenus = nomMenus.length;
menus = new JMenu[nbMenus];
for (int i = 0;i < nbMenus;i++) {
// création des menus
        menus[i] = new JMenu(nomMenus[i]);
// ajout des boutons à la barre de menus
        add(menus[i]);
// ajout des écouteurs sur chaque menu
        menus[i].addMenuListener(this);
}

}
```

MenuDetecterChoix.java

Evénement sur un composant atomique (4/5) - Menu

```
// traitement de tous les événements de la barre de menu de type
Menu
public void menuSelected(MenuEvent arg0) {
for (int i = 0;i < nbMenus;i++) {
 if (arg0.getSource() == menus[i])
System.out.println("menu " + i + "sélectionné");
        }
}

public void menuSelected(MenuEvent arg0) {
for (int i = 0;i < nbMenus;i++) {
 if (arg0.getSource() == menus[i])
System.out.println("menu " + i + "désélectionné");
        }
}

public void menuCanceled(MenuEvent arg0) {}
}
```

testMenuDetecterChoix.java

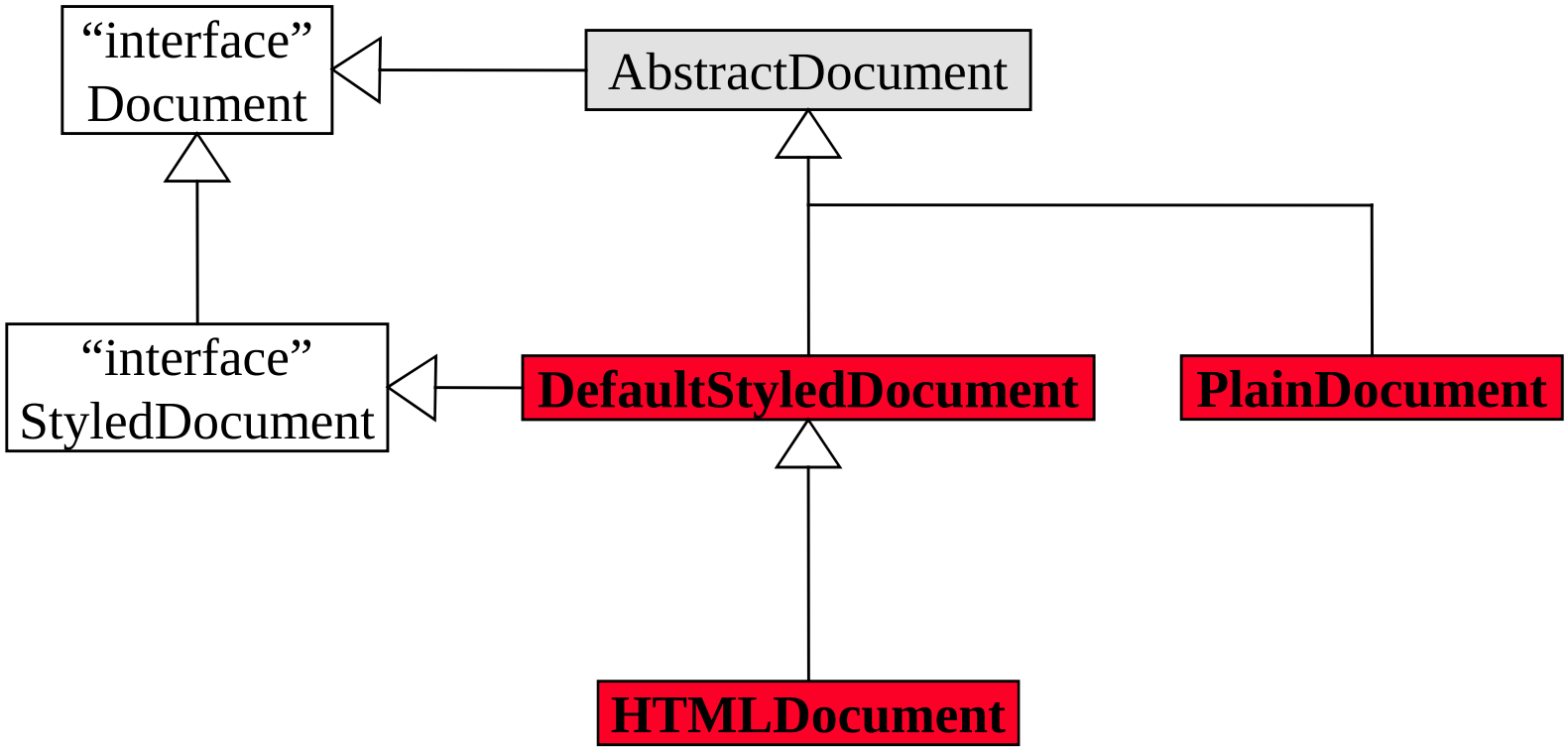
Evénement sur un composant atomique (5/5) - Menu

```
public static void main(String[] args) {
JFrame fen = new JFrame("Détection du choix dans un menu");
fen.setSize(380, 180);
String[] nomMenus = {"orange", "pomme", "banane","raisin"};
MenuDétecterChoix m = new MenuDétecterChoix(nomMenus);
fen.getContentPane().add(m);
fen.setVisible(true);
}
```


```
menu 3 sélectionné
menu 3 désélectionné
menu 2 sélectionné
menu 2 désélectionné
menu 1 sélectionné
```

Modélisation

Trois classes concrètes



Document

Spécification du contrat que toutes les types de document doivent implémenter

Texte du document
String **getText** (int début, int taille)
void **getText** (int début, int taille, Segment cc)

Nombre de caractères du document
int **getLength** ()

Ajout d'un texte cc à l'indice i du document
void **insertString**(int i, String cc, AttributeSet a)

Suppression d'une partie du texte à partir de l'indice i du document
void **remove**(int i, int taille)

Positionnement d'une marque au ième caractère
Position **createPosition** (int i)

Ajout d'un écouteur d'événement modification de document
void **addDocumentListener** (DocumentListener dl)

Interface Position

Indice correspondant à la position d'une marque int **getOffset**()

StyledDocument - Typographie

Création, Gestion et Utilisation de styles de typographie

Création des styles

Applications des méthodes de la classe StyleConstants

```
static void setBackground(Style s, Color  c)          // couleur de fond
static void setForeground(Style s, Color c)           // couleur du caractère

static void setFontFamily(Style s, String  cc)        // police
static void setFontSize(Style s, int corps)           // corps

static void setBold(Style s, boolean b)               // gras
static void setItalic(Style s, boolean  b)            // italique
static void setSubscript(Style s, float x)            // indice
static void setSuperscript(Style s, float x)          // exposant

static void setAlignment(Style s, int alig)           // alignement (gauche, droit,
..)
static void setFirstLineIndent(Style s, float x)      // Indentation
static void setLeftIndent(Style s, float x)
static void setRightIndent(Style s, float x)

static void setSpaceAbove(Style s, float x)
static void setSpaceBelow(Style s, float x)
static void setLineSpacing(Style s, float x)          // taille des lignes
```

StyledDocument - Styles

Gestion des styles

Ajout d'un style s dans la hiérarchie h des styles précédemment définis
Style **addStyle**(Style s, Style h)

Recherche du style s
Style **getStyle**(String cc)

Suppression du style s
void **removeStyle**(String cc)

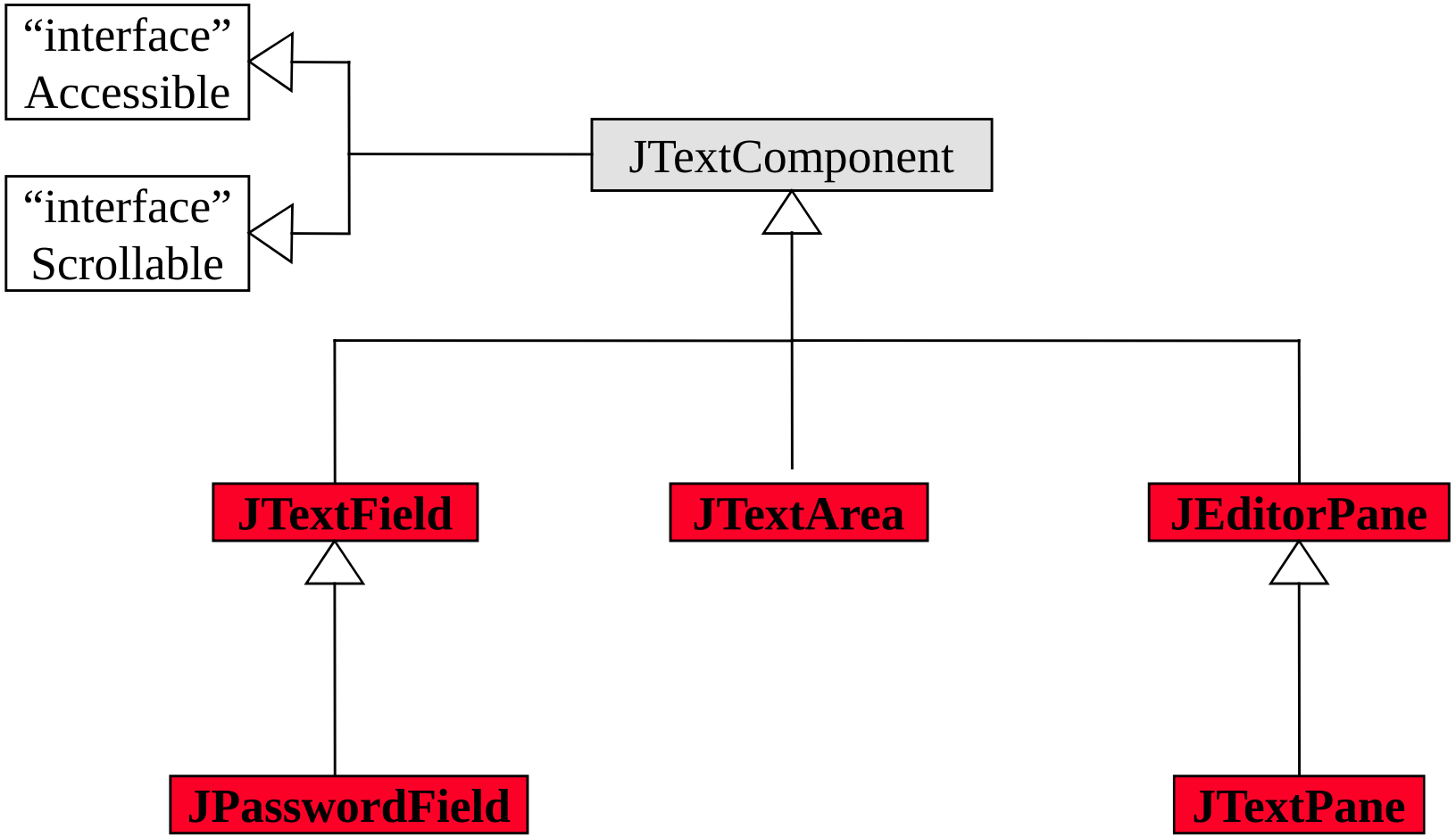
Recherche du style du paragraphe contenant le caractère d'indice i
Style **getLogicalStyle**(int i)

Utilisation des styles

Application ou ajout du style s à une partie du texte
void **setCharacterAttributes**(int i, int taille, Style s, boolean app)
Application du style s au paragraphe contenant le caractère d'indice i
void **setLogicalStyle**(int pos, Style s)

Introduction

Cinq classes concrètes



Classe JTextComponent

Classe abstraite, ancêtre de tous les classes de visualisation de texte

Définition de plus de 100 méthodes

Lien avec un document

void **setDocument** (Document doc)

Gestion du sélecteur (caret)

void **setCaretPosition**(int i) int **getCaretPosition**()

Sélection d'une zone de texte

void **select**(int début, int fin) void **selectAll**()

Lecture et écriture d'un fichier texte

void **write**(Writer out) void **read**(Reader in, Object desc)

Méthodes interactives d'édition (possibilité de redéfinition)

void **copy**() void **cut**() void **paste**()

Ecouteurs

void **addCaretListener**(CaretListener listener)

Classe JTextArea

Composant de visualisation de documents textuels simples

Compatibilité avec les documents de type PlainDocument,
Vue d'une partie du document (implémentation de l'interface Scrollable),
Contrôle des coupures de fin de ligne

Constructeurs

Constructeur vide

JTextArea()

Construction à partir du document doc avec l lignes, c colonnes, et le titre t

JTextArea(Document doc, String t, int l, int c)

JTextArea(Document doc)

JTextArea(String t, int l, int c)

JTextArea(String t)

JTextArea(int l, int c)

Gestion des fin de ligne et des tabulations

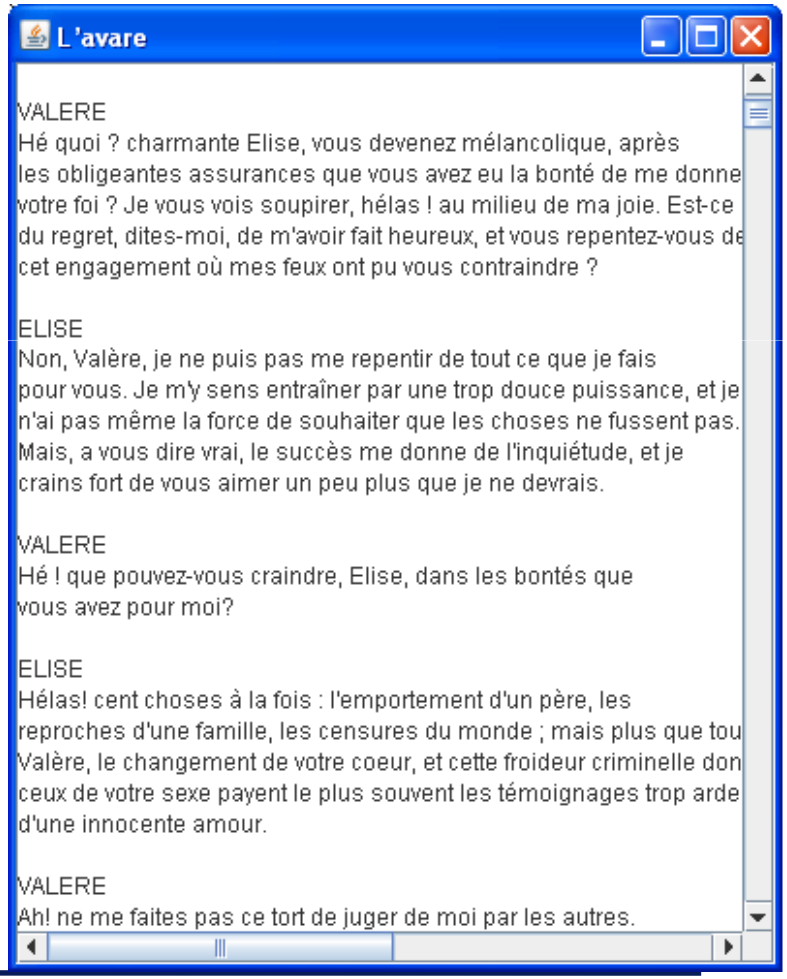
void **setLineWrap**(boolean wrap) void **setWrapStyleWord**(boolean word)

void **setTabSize**(int taille)

JTextArea – Programme

testJTextArea.java

```
PlainDocument pdoc = new PlainDocument();
String NomFichier = "livres/avare.txt";
pdoc.putProperty("Titre" , NomFichier);

JTextArea ta = new JTextArea(pdoc);

try {
ta.read(new FileReader(NomFichier), null);
}
catch (IOException e) {}


JFrame fen = new JFrame("L'avare");
fen.setSize(400, 500);
Container jp = fen.getContentPane();
jp.add(ta);
jp.add(new JScrollPane(ta));
fen.setVisible(true);
```



Classe JEditorPane

Composant de visualisation de documents textuels enrichis

Lecture des formats texte pur, RTF et HTML,
Initialisation à partir d'une URL
Gestion des hyperliens

Constructeurs

Constructeur vide

JEditorPane()

Construction à partir de la chaîne cc de format f (txt, rft, html)

JEditorPane(String f, String cc)

Construction à partir de l'URL u

JEditorPane(URL u)

Construction à partir d'une chaîne de caractère correspondant à une URL

JEditorPane(String url)

Classe JTextPane

Composant de visualisation de documents textuels enrichies

Classe dérivée de JEditorPane,
Compatibilité avec les documents de type DefaultStyledDocument,
Attachement d'images et de composants à des parties du texte

Constructeurs

Constructeur vide

JTextPane()

Construction à partir du document doc

JTextPane(StyledDocument doc)

Liens avec les documents

void **setDocument**(Document doc)

void **setStyledDocument**(StyledDocument doc)

testJTextPane.java

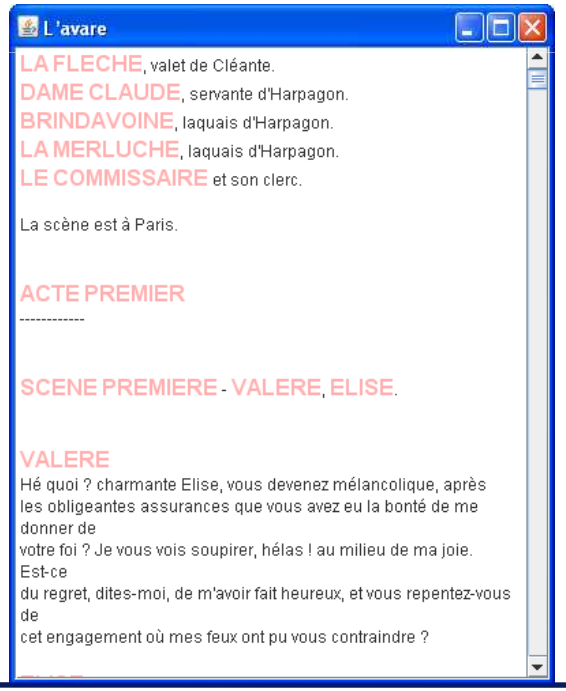
Classe JTextPane – Programme (1/2)

```
String NomFichier = "livres/avare.txt";
JTextPane tp = new JTextPane();

// lecture du document
try {
tp.read(new FileReader(NomFichier), null); }
catch (IOException e) {}
String cc = null;
try {cc = dsdoc.getText(0, dsdoc.getLength()); }
catch (BadLocationException e) {}


// définition du style Personnage
StyledDocument dsdoc = tp.getStyledDocument();
dsdoc.putProperty("Titre" , "L'avare");
Style s = dsdoc.addStyle("Personnage" , null);
StyleConstants.setForeground(s, Color.PINK);
StyleConstants.setFontSize(s, 16);
StyleConstants.setBold(s, true);
```

testJTextPane.java

Classe JTextPane – Programme (2/2)

```
//enrichissement du texte
BreakIterator bi1 = BreakIterator.getWordInstance();
bi1.setText(cc);
int debut = bi1.first();
for (int fin = bi1.next(); fin != BreakIterator.DONE; debut = fin, fin
= bi1.next()) {
if  (cc.substring(debut, fin).matches("[A-Z]*"))
dsdoc.setCharacterAttributes(debut, fin-debut, s, true);
}

//affichage
JFrame fen = new JFrame("L'avare");
fen.setSize(400, 500);
Container jp = fen.getContentPane();
jp.add(tp);
jp.add(new JScrollPane(tp));
fen.setVisible(true);
```



Formes géométriques 2D

Classe Graphics (package awt)

Méthodes applicables sur les objets conteneurs (de préférence JPanel),
drawXXX dessine une forme géométrique,
fillXXX remplit une forme géométrique avec XXX (Arc, Line, Oval, Polygon, Rect)
setColor (couleur sur 24 bits)
copyArea (copie),
setFont (type de fontes),
drawString (afficher une chaîne de caractères), ….

Instanciation d'un objet de type Graphics à partir d'un conteneur

Dessin à la volée

Récupération d'un contexte graphique (getGraphics),
Dessin,
Libération du contexte graphique (dispose)

Dessin protégé contre les modifications de fenêtre

Redéfinition dans une classe dérivée du conteneur de la méthode paintComponent,
Appel à la méthode repaint() en cas de modification du dessin

Dessin.java

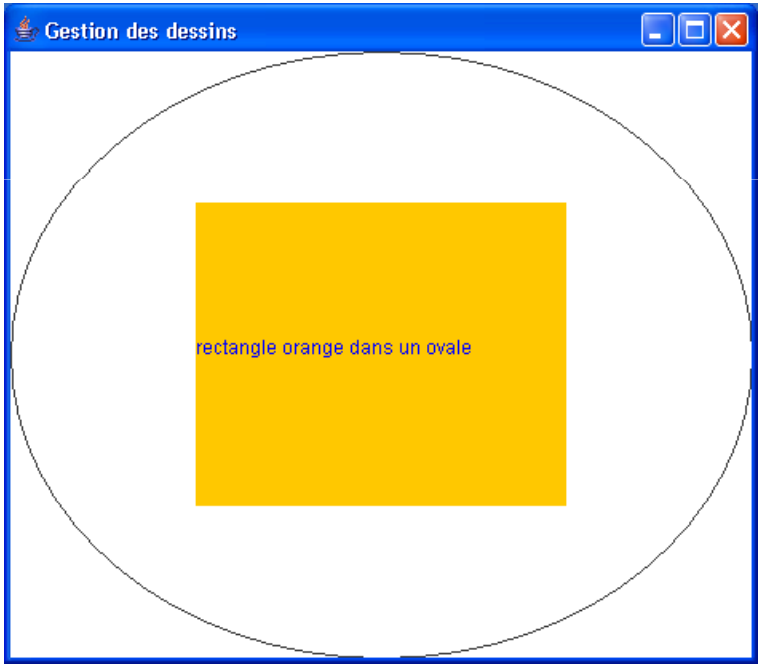
Dessin dans un panneau – Code (1/2)

```
public class Dessin extends JPanel {

   public void paintComponent(Graphics g) {

 // lecture de la taille du paneau
 int height  = this.getHeight();
 int width = this.getWidth();

 // effacement du dessin précédent
 g.clearRect(0, 0, width, height);

 // nouveau dessin
 g.drawOval(0, 0, width, height);
 g.setColor(Color.ORANGE);
 g.fillRect(width/4, height/4, width/2, height/2);
 g.setColor(Color.BLUE);
 g.drawString("rectangle orange dans un ovale", width/4, height/2);
} }
```

testDessin.java

Dessin dans un panneau – Code (2/2)

```
// classe de test

public static void main(String[] args) {
        JFrame fen = new JFrame("Gestion des dessins");
        fen.setSize(280, 280);

JPanel p = new JPanel();
fen.getContentPane().add(p);
p.setLayout(new GridLayout(1,1));

        Dessin d = new Dessin();
p.add(d);

fen.setVisible(true);
 }
```



Gestion d'images bitmap

Classe ImageIcon (package swing)

Constructeurs

public **ImageIcon**()

public **ImageIcon**(Image im)

public **ImageIcon**(String nomFichier)

public **ImageIcon**(URL nomURL)

Méthodes

public int **getIconHeight**()	hauteur en pixel
public int **getIconWidth**()	largeur en pixel
public Image **getImage**()	accès à l'objet Image correspondant public void **setImage**(Image image)	Initialisation par un objet Image

Classe Image (package awt)

Lecture par les méthodes getToolkit.getImage() d'un conteneur
fichier, URL

Affichage par les méthodes drawImage de la classe Graphics

Bitmap.java

Affichage d'une image dans un panneau – Code (1/2)

```
public class Bitmap extends JPanel {
    Image im;

public Bitmap(String nomImage) {
im = getToolkit().getImage(nomImage);
}

public void paintComponent(Graphics g) {

 // lecture de la taille du paneau
int height  = this.getHeight();
       int width = this.getWidth();

 // effacement du dessin précédent
       g.clearRect(0, 0, width, height);

// affichage de l'image
g.drawImage(im, 0, 0, this);
}
}
```

testBitmap.java

Affichage d'une image dans un panneau – Code (2/2)

```
// classe de test

public static void main(String[] args) {
        JFrame fen = new JFrame("Gestion des dessins");
        fen.setSize(280, 280);

JPanel p = new JPanel();
fen.getContentPane().add(p);
p.setLayout(new GridLayout(1,1));

        Bitmap b = new Bitmap ("cours09/Sorbonne.jpg");
p.add(b);

fen.setVisible(true);
 }
```