

IHM en Java – l'API Swing

Licence 3 – Université du Havre
Bruno Mermet



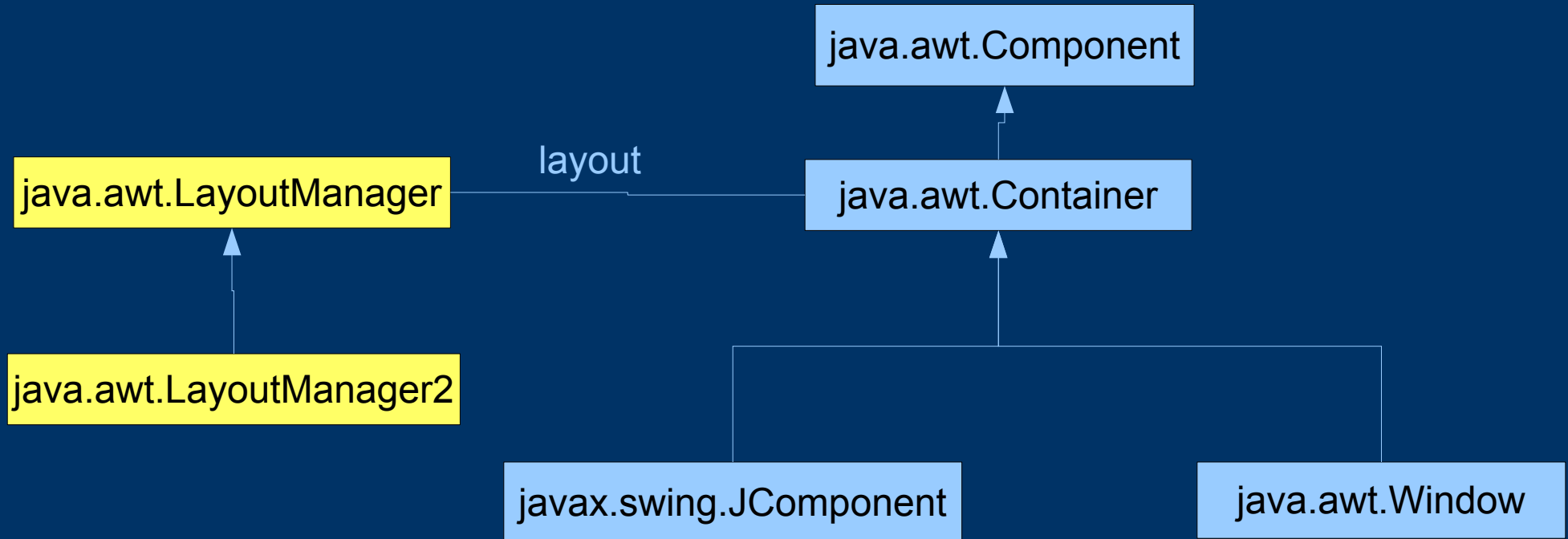
Principes généraux

- Applications graphiques = applications multi-threads
 - Un ou plusieurs threads pour l'application
 - Un thread dédié à la gestion des événements (prise en compte des actions de l'utilisateur : frappes clavier, clics souris)
 - L'interface graphique est constituée de *composants*
 - Pour prendre en compte les interactions de l'utilisateur avec un composant, l'application doit définir des objets observateurs de ces composants
 - Certains composants sont des *conteneurs* de composants
 - La disposition des composants dans les conteneurs est gérée par des *gestionnaires de présentation*
 - *Look-and-feel* paramétrable
-
-

Composants lourds / légers



Architecture Swing Générale



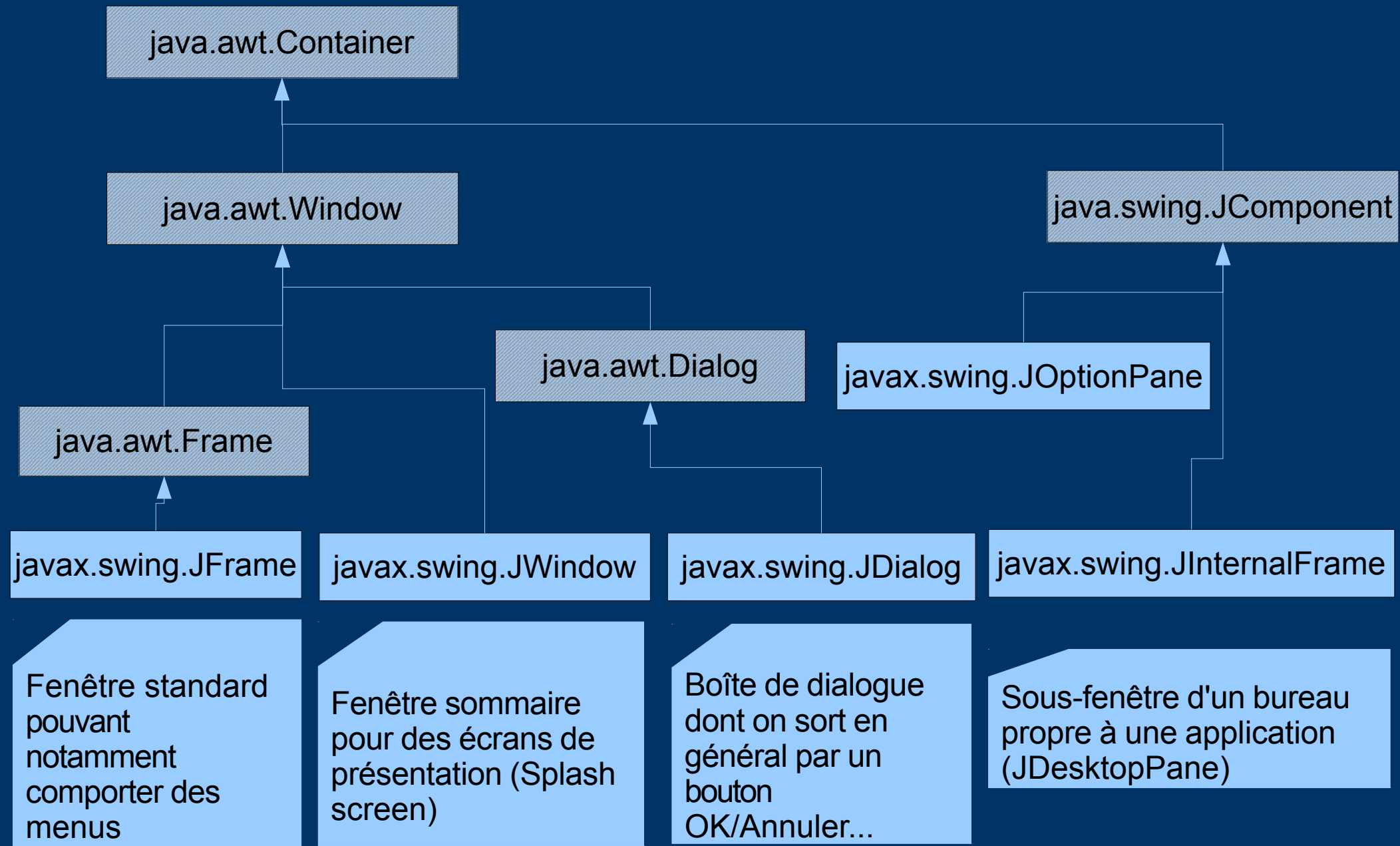
Remarque :

contrairement à awt, l'héritage ne permet pas de distinguer un "conteneur" d'un autre composant

Les conteneurs

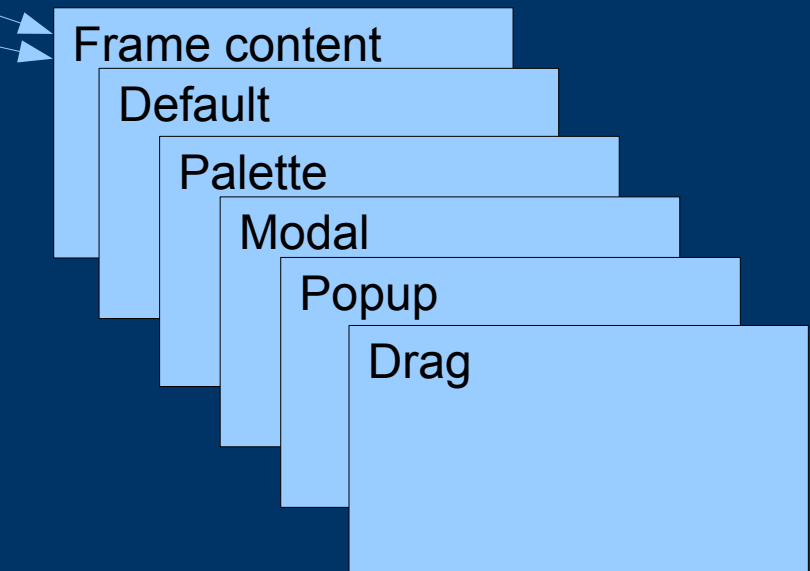
- Permettent d'inclure plusieurs composants
 - Sont des composants
 - Peuvent être insérés partout où des composants peuvent être insérés
 - Sont associés à un gestionnaire de présentation (LayoutManager) qui organise l'affichage des différents composants du conteneur
 - Peuvent être des "fenêtres" ou pas
-
-

Les conteneurs Swing de type "Fenêtre"



Le composant *JRootPane*

- Contenu de tous les conteneurs de type "fenêtre"
- Constitué de 4 parties
 - Glass pane (invisible par défaut, transparent, devant les autres)
 - Layered pane : pour une gestion des affichages en couches
 - Content pane : contient les composants de la fenêtre
 - Menu bar (optionnel)
- Gestion des couches



Ajouter des composants à une fenêtre

- Définir un nouveau conteneur comme "ContentPane" (préférable) et lui associer un gestionnaire de présentation

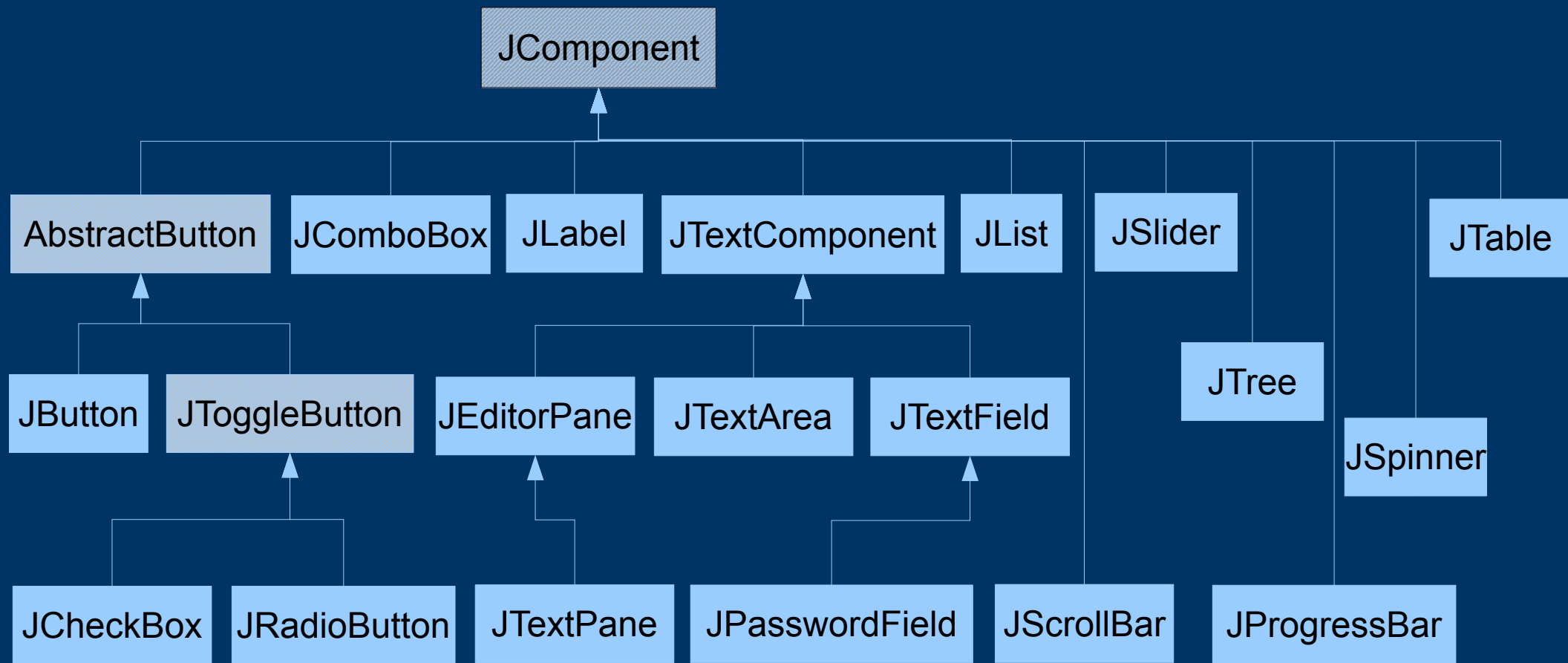
Exple :

- `JPanel contenu = new JPanel();`
 - `contenu.setLayoutManager(new BorderLayout());`
 - `maFenetre.setContentPane(contenu);`
 - Ajouter les composants au conteneur en question
 - Soit `contenu.add(new JButton())`
 - Soit `maFenetre.getContentPane().add(new JButton())`
 - Soit (raccourci) `maFenetre.add(new JButton())`
-
-

Les autres conteneurs Swing

- JPanel : conteneur général "de base"
 - JScrollPane : conteneur avec ascenseurs
 - JLayeredPane : conteneur à plusieurs couche
 - JTabbedPane : conteneurs d'onglets
 - JDesktopPane : Sous-bureau pouvant contenir des sous-fenêtres (JInternalFrame)
 - JApplet : pour les applets !
 - JSplitPane : pour partager une fenêtre en 2
-
-

Quelques composants Swing



Aperçus des composants Swing (d'après java.sun.com)

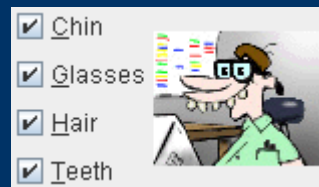
JLabel



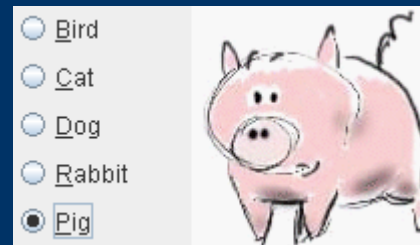
JButton



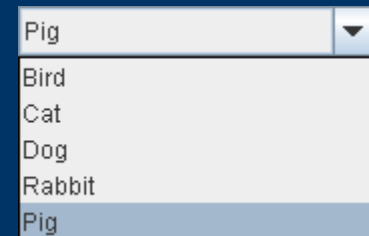
JCheckBox



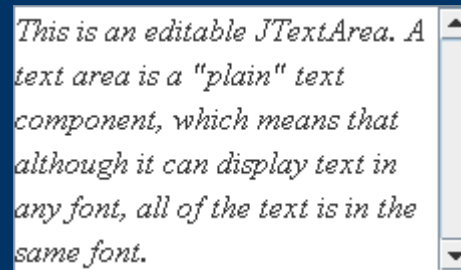
JRadioButton



JComboBox



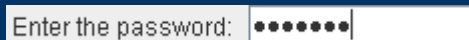
JTextArea



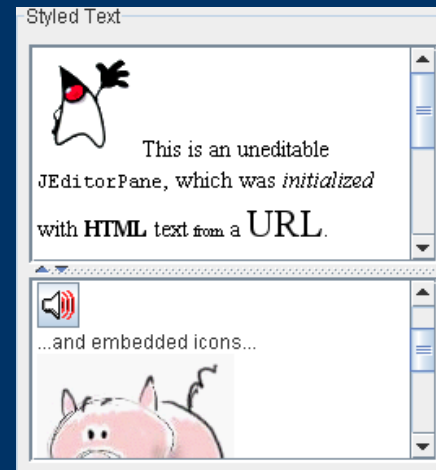
JTextField



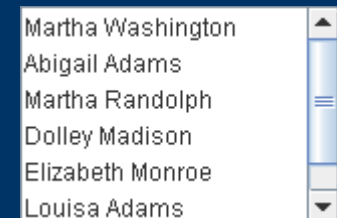
JPasswordField



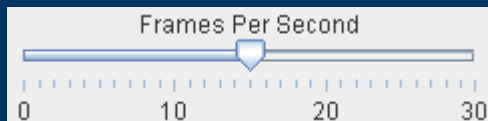
JEditorPane/JTextPane



JList



JSlider



JTable

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34\$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541!fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail...	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

JSpinner



JProgressBar



JTree



Aperçu d'un composant : JLabel

(1)

- But : afficher un texte, et éventuellement une icône associée
- Constructeurs

JLabel()

JLabel(Icon image)

JLabel(Icon image, int horizontalAlignment)

JLabel(String text)

JLabel(String text, Icon icon, int horizontalAlignment)

JLabel(String text, int horizontalAlignment)

- Méthodes définies directement ou redéfinies

checkHorizontalKey(int key, String message), checkVerticalKey(int key, String message),
getAccessibleContext(), getDisabledIcon(), getDisplayedMnemonic(), getDisplayedMnemonicIndex(),
getHorizontalAlignment(), getHorizontalTextPosition(), getIcon(), getIconTextGap(), getLabelFor(),
getText(), getUI(), getVerticalTextPosition(), imageUpdate(Image img, int infoflags, int x, int y, int w,
int h), paramString(), setDisabledIcon(Icon disabledIcon), setDisplayedMnemonic(char aChar),
setDisplayedMnemonic(int key), setDisplayedMnemonicIndex(int index), setHorizontalAlignment(int
alignment), setHorizontalTextPosition(int textPosition), setIcon(Icon icon), setIconTextGap(int
iconTextGap), setLabelFor(Component c), setText(String text), setUI(LabelUI ui),
setVerticalAlignment(int alignment), setVerticalTextPosition(int textPosition), updateUI()

Aperçu d'un composant : JLabel

(2)

- Méthodes de JComponent

addAncestorListener, addNotify, addVetoableChangeListener, computeVisibleRect, contains, createToolTip, disable, enable, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getDebugGraphicsOptions, getDefaultLocale, getFontMetrics, getGraphics, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, **getMaximumSize**, **getMinimumSize**, getNextFocusableComponent, getPopupLocation, **getPreferredSize**, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getX, getY, grabFocus, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPaintingForPrint, isPaintingTile, isRequestFocusEnabled, isValidRoot, paint, paintBorder, paintChildren, paintComponent, paintImmediately, paintImmediately, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyEvent, processMouseEvent, processMouseMotionEvent, putClientProperty, registerKeyboardAction, registerKeyboardAction, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, **setBorder**, setComponentPopupMenu, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setEnabled, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, **setMaximumSize**, **setMinimumSize**, setNextFocusableComponent, **setOpaque**, **setPreferredSize**, setRequestFocusEnabled, **setToolTipText**, setTransferHandler, setUI, setVerifyInputWhenFocusTarget, setVisible, unregisterKeyboardAction, update

Aperçu d'un composant : JLabel

(3)

- Methodes de Container

add, add, add, add, add, addContainerListener, addImpl, addPropertyChangeListener, addPropertyChangeListener, applyComponentOrientation, areFocusTraversalKeysSet, countComponents, deliverEvent, doLayout, findComponentAt, findComponentAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents, getComponentZOrder, getContainerListeners, getFocusTraversalKeys, getFocusTraversalPolicy, getLayout, getMousePosition, insets, invalidate, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, layout, list, list, locate, minimumSize, paintComponents, preferredSize, printComponents, processContainerEvent, processEvent, remove, remove, removeAll, removeContainerListener, setComponentZOrder, setFocusCycleRoot, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setLayout, transferFocusBackward, transferFocusDownCycle, validate, validateTree

Aperçu d'un composant : JLabel

(4)

- Méthodes de Component

action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener, addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener, addMouseMotionListener, addMouseWheelListener, bounds, checkImage, checkImage, coalesceEvents, contains, createImage, createImage, createVolatileImage, createVolatileImage, disableEvents, dispatchEvent, enable, enableEvents, enableInputMethods, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange, getBackground, getBounds, getColorModel, getComponentListeners, getComponentOrientation, getCursor, getDropTarget, getFocusCycleRootAncestor, getFocusListeners, getFocusTraversalKeysEnabled, getFont, getForeground, getGraphicsConfiguration, getHierarchyBoundsListeners, getHierarchyListeners, getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests, getKeyListeners, getLocale, getLocation, getLocationOnScreen, getMouseListeners, getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName, getParent, getPeer, getPropertyChangeListeners, getPropertyChangeListeners, getSize, getToolkit, getTreeLock, gotFocus, handleEvent, hasFocus, hide, inside, isBackgroundSet, isCursorSet, isDisplayable, isEnabled, isFocusable, isFocusOwner, isFocusTraversable, isFontSet, isForegroundSet, isLightweight, isMaximumSizeSet, isMinimumSizeSet, isPreferredSizeSet, isShowing, isValid, isVisible, keyDown, keyUp, list, list, list, location, lostFocus, mouseDown, mouseDrag, mouseEnter, mouseExit, mouseMove, mouseUp, move, nextFocus, paintAll, postEvent, prepareImage, prepareImage, processComponentEvent, processFocusEvent, processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent, processMouseWheelEvent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, resize, resize, setBounds, setBounds, setComponentOrientation, setCursor, setDropTarget, setFocusable, setFocusTraversalKeysEnabled, setIgnoreRepaint, setLocale, setLocation, setLocation, setName, setSize, setSize, show, show, size, toString, transferFocus, transferFocusUpCycle

Aperçu d'un composant : JLabel

(5)

- Methodes de Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait



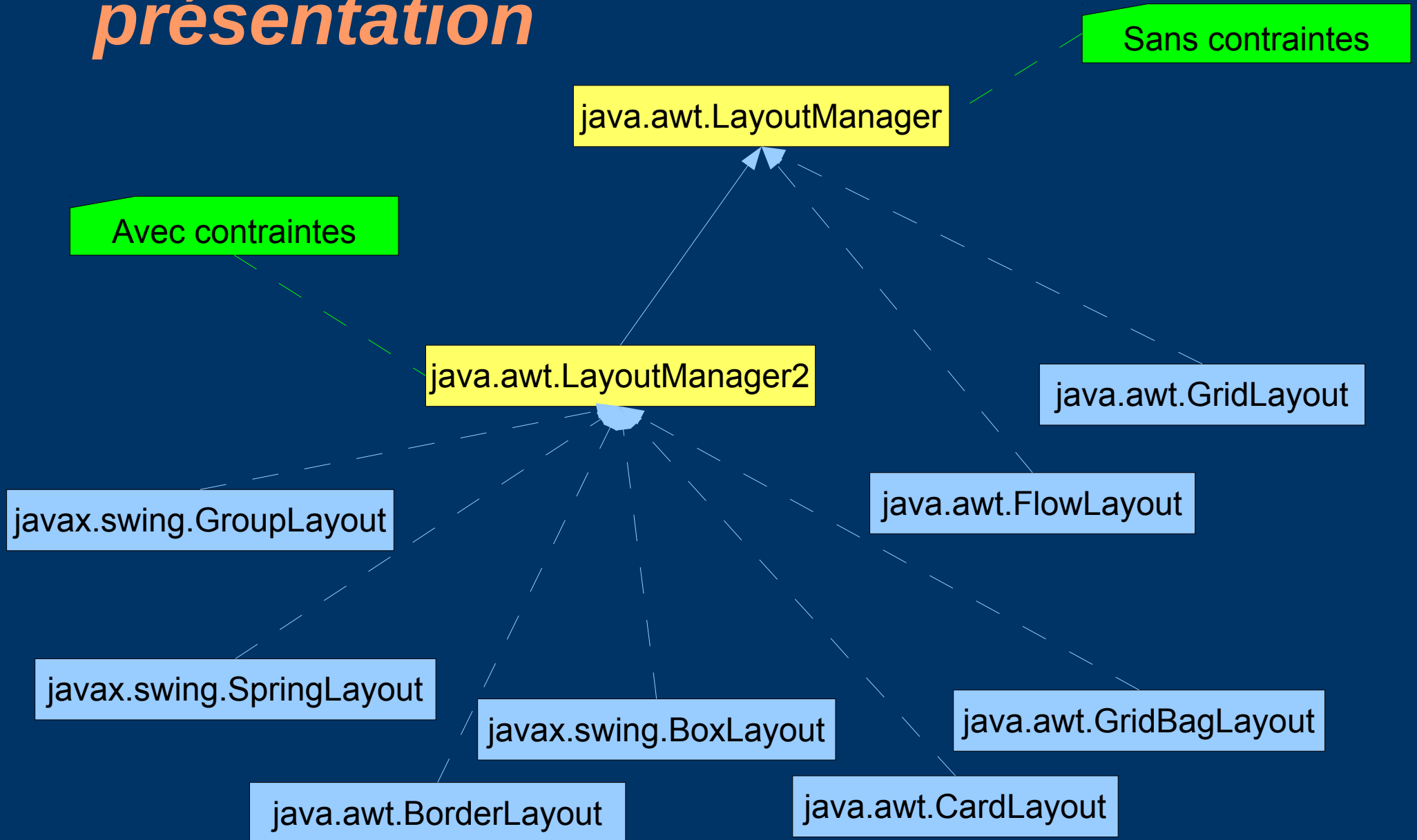
Affichage d'un composant

- Effectué par la méthode `paint()`, devant être appelé par le Thread chargé des événements :
 - Soit automatiquement (notamment à la demande du conteneur)
 - Soit manuellement dans une méthode exécutée par le Thread chargé des événements
 - Soit par un `SwingUtilities.invokeLater(monThread)`
 - Taille : 3 tailles
 - `MinimumSize`
 - `MaximumSize`
 - `PreferredSize` (utilisé par défaut, notamment dans le cas d'un `pack()`)
 - Visibilité
 - `setVisible(true)` / `setVisible(false)`
 - `setOpaque(true)` / `setOpaque(false)`
-
-

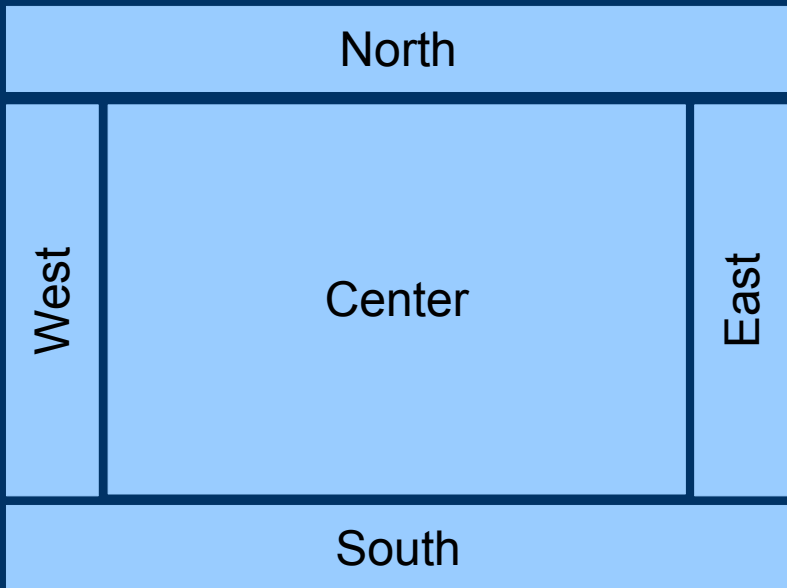
Les gestionnaires de présentation



Les principaux gestionnaires de présentation



BorderLayout



- Détermine 5 zones (schémas)
- Chaque composant est ajouté au conteneur dans une des zones
- Les zones périphériques ont pour taille la *taille préférée** du composant qu'elles contiennent
- La zone centrale a pour taille le reste

* largeur(West) = largeurPréférée(West)

hauteur(West) = max(hauteurPréférée(West), hauteurPréférée(East))

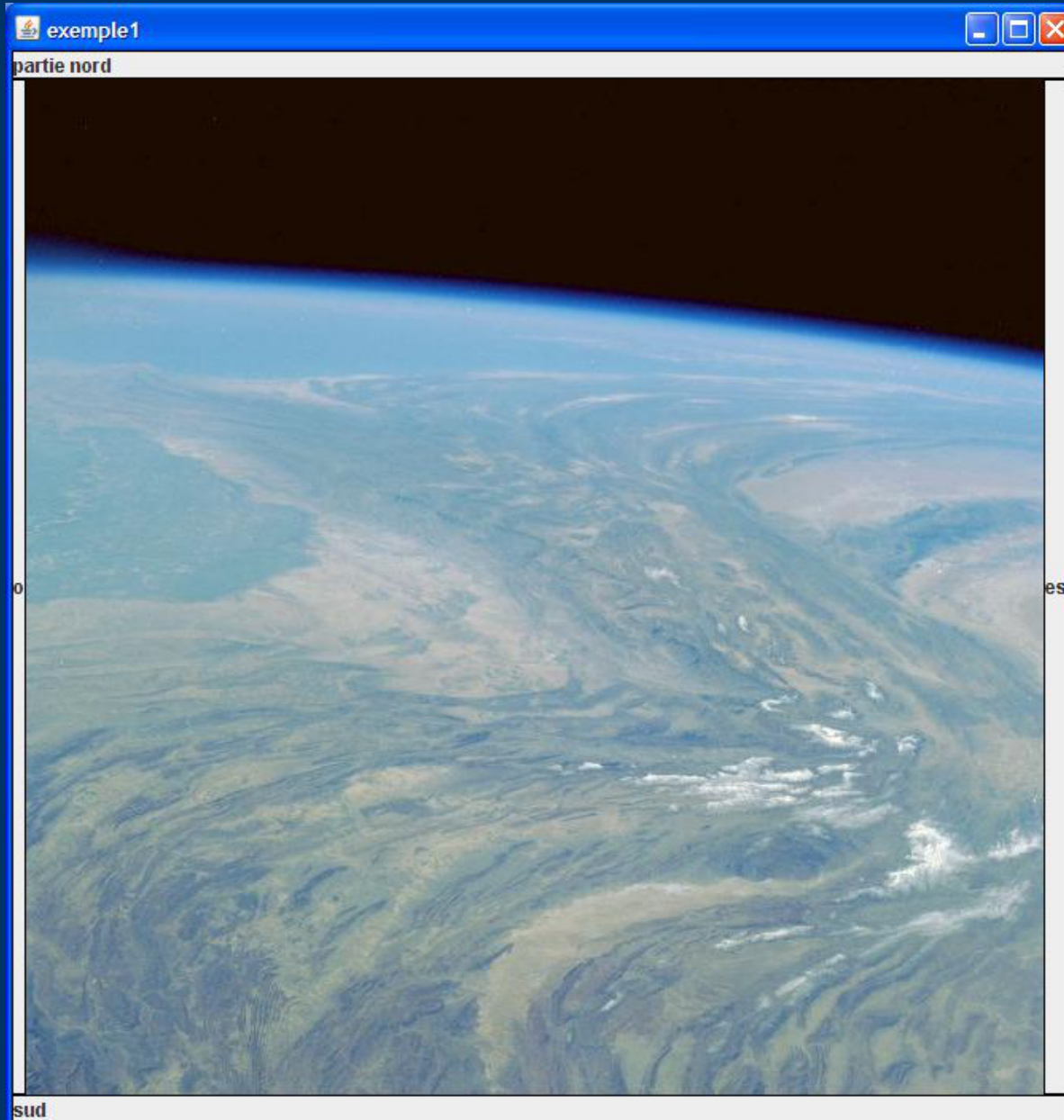
...

Exemple (code)

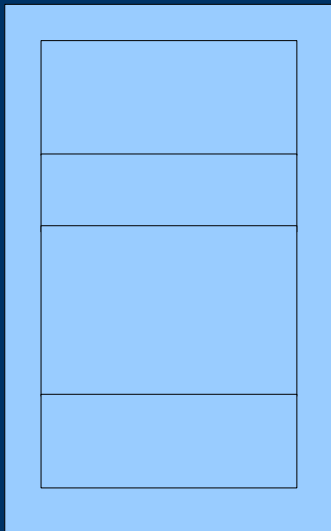
```
public class Exemple1 extends JFrame {  
  
    public Exemple1() {  
        super("exemple1");  
        Container cont = getContentPane();  
        cont.setLayout(new BorderLayout());  
        JLabel nord = new JLabel("partie nord");  
        nord.setBorder(new  
        LineBorder(Color.BLACK));  
        JLabel sud = new JLabel("sud");  
        sud.setBorder(new  
        LineBorder(Color.BLACK));  
        JLabel ouest = new JLabel("o");  
        ouest.setBorder(new  
        LineBorder(Color.BLACK));  
        JLabel est = new JLabel("e\ns\nnt");  
        est.setBorder(new  
        LineBorder(Color.BLACK));  
        cont.add(nord, BorderLayout.NORTH);  
        cont.add(sud, BorderLayout.SOUTH);  
        cont.add(ouest, BorderLayout.WEST);  
        cont.add(est, BorderLayout.EAST);
```

```
JLabel centre = new JLabel(new  
        ImageIcon("earth.jpg"));  
        cont.add(centre, BorderLayout.CENTER);  
        pack();  
        setVisible(true);  
  
        setDefaultCloseOperation(JFrame.EXIT_ON_  
        CLOSE);  
    }  
  
    public static void main(String[] args) {  
        Exemple1 ex = new Exemple1();  
    }  
}
```

Exemple (Affichage)

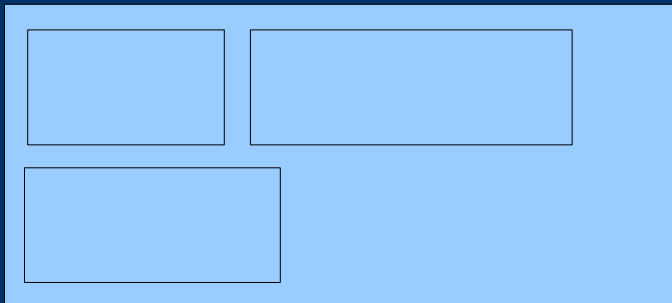
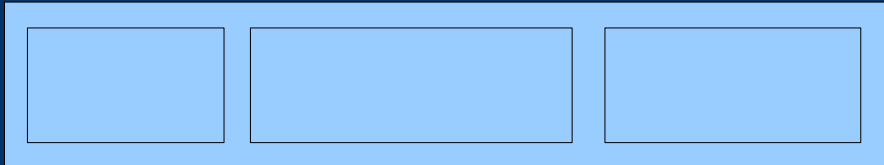


BoxLayout



- Permet de disposer des composants en colonne (Y_AXIS) ou en ligne (X_AXIS), les composants sont collés par défaut
- Constructeur
BoxLayout(Container c, int axis)
- Ajout
add(component)
- Séparation
 - Fixe
Box.createRigidArea(Dimension taille)
 - Variable
Box.createHorizontalGlue()
Box.createVerticalGlue()

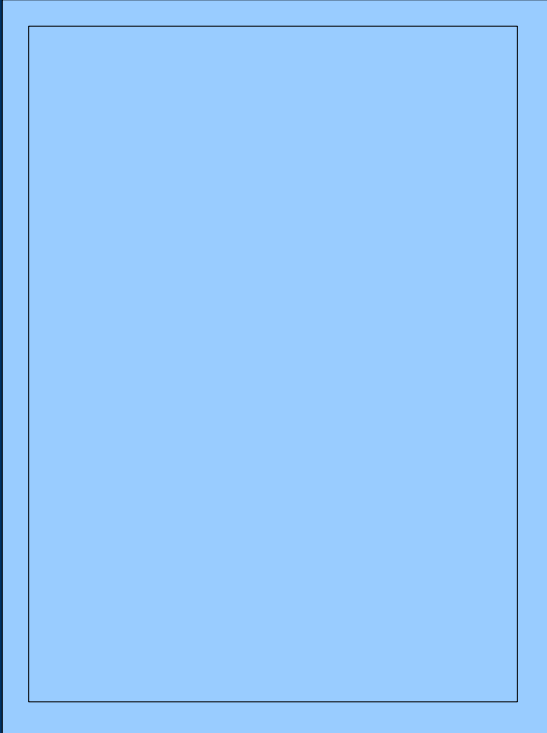
FlowLayout



- Permet de disposer des composants en ligne avec retour à la ligne si nécessaire (espacement par défaut de 5)
- Constructeurs
FlowLayout(), FlowLayout(int align)
- Ajout
add(component)

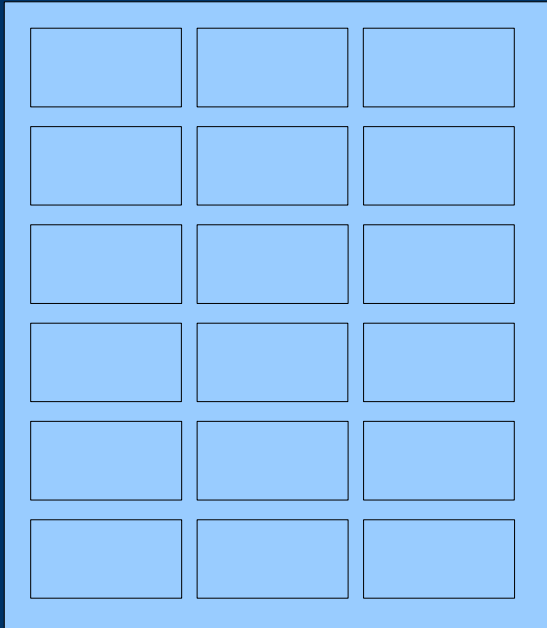
Alignements possibles :
FlowLayout.LEFT
FlowLayout.CENTER
FlowLayout.RIGHT
FlowLayout.LEADING
FlowLayout.TRAILING

CardLayout



- Traite les composants comme une pile de cartes : seul le composant du dessus est visible
- Constructeurs
`CardLayout()`
`CardLayout(hgap,vgap)`
- Ajout
`add(composant, String nom)`
- Affichage d'une carte
`show(parent, nom)`
`previous(parent),next(parent)`
`first(parent), last(parent)`

GridLayout



- Dispose les composants sur une grille dont toutes les cases ont la même taille

- Constructeurs

`GridLayout()`

`GridLayout(lignes,colonnes)`

`GridLayout(lignes,colonnes,hgap,vgap)`

- Ajout

`add(composant)`

Les composants sont ajoutés ligne après ligne

GridBagLayout

- Très complet
 - Repose sur une grille
 - Chaque composant est ajouté avec un objet *GridBagConstraints* associé
 - les attributs de l'objet *GridBagConstraints* sont copiés, donc le même objet peut être utilisé pour plusieurs composants, en modifiant ses attributs si nécessaire
 - Les attributs des objets *GridBagConstraints* sont publics et aucun accesseur n'est défini pour y accéder
-
-

Attributs de GridBagConstraints

- **int anchor** : si le composant est plus petit que sa "case", spécifie où le placer dans la case
 - **int fill** : si le composant est plus petit que sa "case", spécifie comment le redimensionner
 - **int gridheight, gridwidth** : nombre de lignes et colonnes de la grille occupées par le composant
 - **int gridx, gridy** : coordonnées de la première cellule (la plus en haut à gauche) de la grille occupée par le composant
 - **Insets insets** : l'espace autour du composant
 - **int ipadx, ipady** : espace interne rajouté au composant
 - **double weightx, weighty** : répartition de l'augmentation de taille des composants lors de la modification de taille du conteneur
-
-

Insets, ipadx, ipady : Précisions

Valeurs par défaut :

- Insets : `new Insets(0,0,0,0)`
- `ipadx, ipady` : 0

Anchor : détails

- Valeurs absolues
CENTER, NORTH, NORTHEAST, EAST, SOUTHEAST,
SOUTH, SOUTHWEST, WEST, NORTHWEST
 - Valeurs relatives à l'orientation
PAGE_START, PAGE_END, LINE_START, LINE_END,
FIRST_LINE_START, FIRST_LINE_END, LAST_LINE_START
LAST_LINE_END
 - Valeurs relatives à la "ligne de base"
BASELINE, BASELINE_LEADING, BASELINE_TRAILING
ABOVE_BASELINE, ABOVE_BASELINE_LEADING,
ABOVE_BASELINE_TRAILING, BELOW_BASELINE,
BELOW_BASELINE_LEADING,
BELOW_BASELINE_TRAILING
 - Valeur par défaut
CENTER
-
-

Fill : détails

- Valeurs possibles
 - NONE
 - HORIZONTAL
 - VERTICAL
 - BOTH
- Valeur par défaut
 - NONE



gridx, gridy, gridheight, gridwidth

- Lignes et colonnes numérotées à partir de 0
- La valeur spéciale RELATIVE peut être utilisée pour un positionnement par rapport aux autres composants

weightx, weighty : précisions

- Valeur par défaut : 0
 - Lorsque la taille du conteneur est augmentée, la part de l'augmentation de la taille affectée au i ème composant est proportionnelle à la part du `weightx` (resp. `weighty`) du i ème composant par rapport à la somme des poids des `weightx` (resp. `weighty`) des autres composants de la même ligne (resp. colonne)
 - Exemple
 - Une ligne avec 3 composants de `weightx` resp. (0, 1, 2)
 - Largeur du conteneur augmentée de 60 pixels
 - Augmentation resp. des 3 composants : (0, 20, 40)
-
-

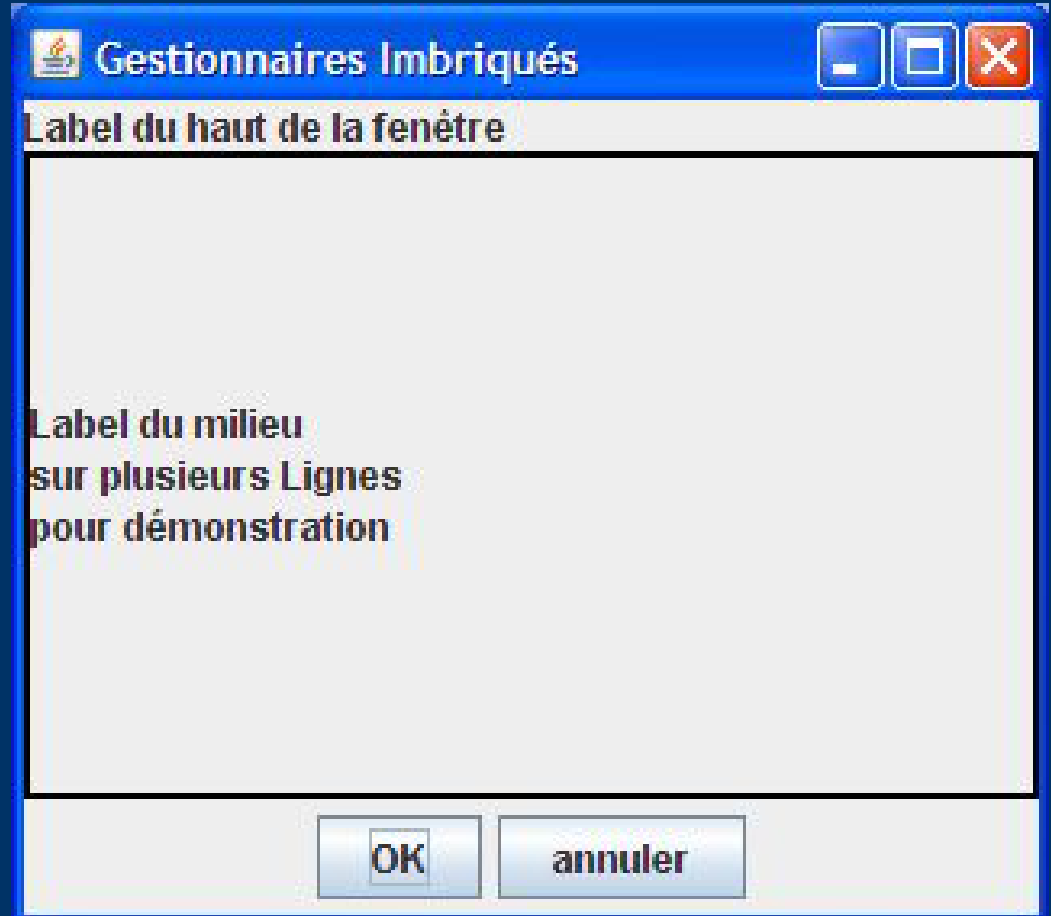
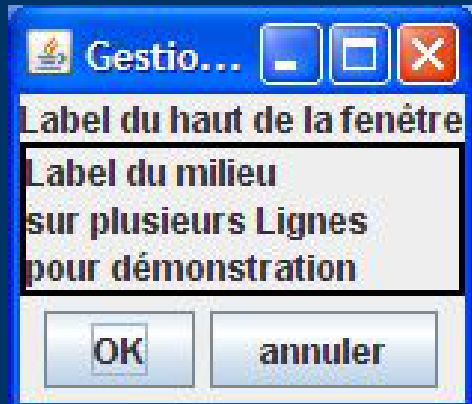
Gestionnaires de présentation :

En pratique

- Soit un gestionnaire prédéfini convient directement
- Soit mise en page plus compliquée
 - Utilisation de GridBagLayout
 - Utilisation d'imbrications de conteneurs :
 - Un conteneur de premier niveau, avec par exemple un BorderLayout
 - Dans une des parties du BorderLayout, insertion d'un JPanel (conteneur) auquel on affecte un nouveau gestionnaire de présentation (par exemple FlowLayout, ou à nouveau BorderLayout, etc.)

Imbrication de gestionnaires

Exemple : effet voulu

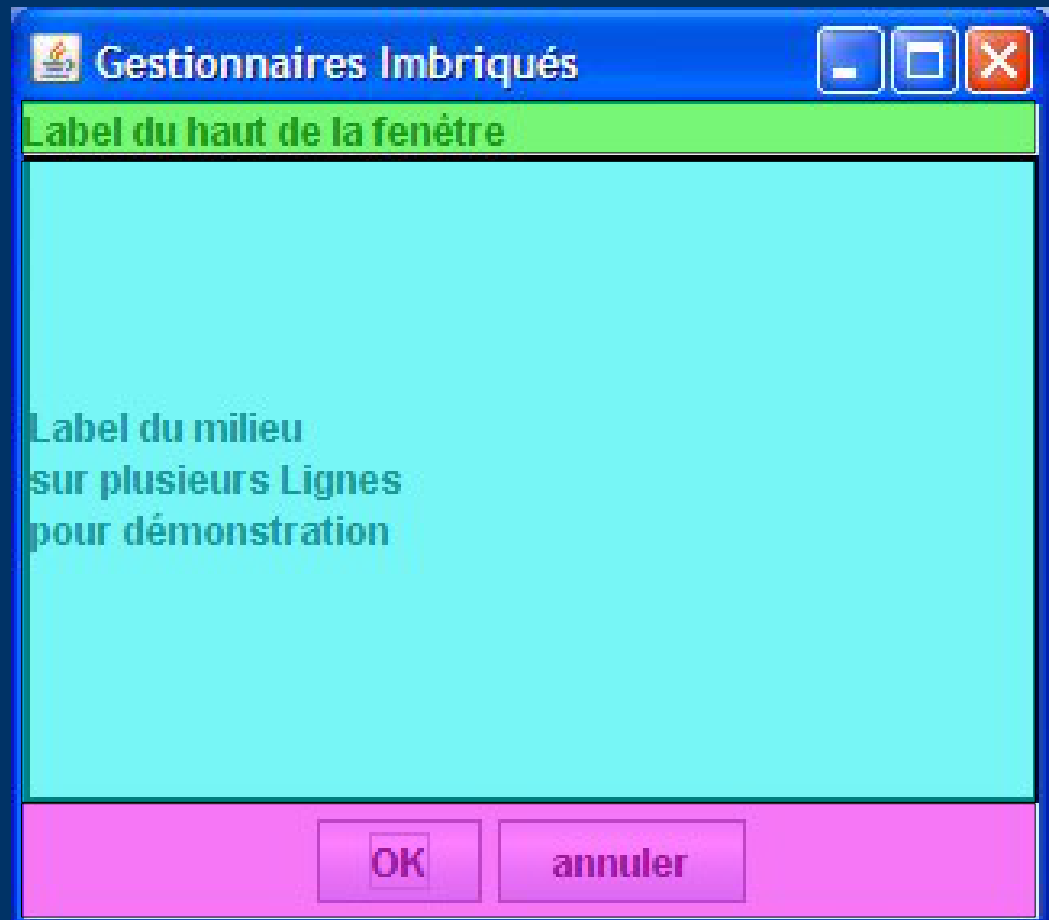


Les boutons restent en bas
Ils sont centrés
La zone du haut est de hauteur fixe
Seule la zone du milieu s'agrandit
verticalement

Imbrication de gestionnaires

Exemple : analyse

- Un BorderLayout dont on utilise les zones **NORTH**, **CENTER** et **SOUTH**
- La zone **SOUTH** est rempli par un JPanel avec un FlowLayout (alignement centré) auquel on ajoute les 2 boutons



Imbrication de gestionnaires

Exemple

```
public class ImbricationGestionnairesPresentation extends JFrame {
    public ImbricationGestionnairesPresentation() {
        super("Gestionnaires Imbriqués");
        Container contenu = getContentPane();
        contenu.setLayout(new BorderLayout());
        // Le haut de la fenêtre
        JLabel labelHaut = new JLabel("Label du haut de la fenêtre");
        contenu.add(labelHaut, BorderLayout.NORTH);
        // Le milieu de la fenêtre
        JLabel labelMilieu = new JLabel("<html>Label du milieu<BR/>"
+ "sur plusieurs Lignes<BR/>" + "pour démonstration</html>");
        labelMilieu.setBorder(new LineBorder(Color.BLACK,2));
        contenu.add(labelMilieu, BorderLayout.CENTER);
        // Les boutons du bas
        JPanel bas = new JPanel();
        bas.setLayout(new FlowLayout(FlowLayout.CENTER));
        JButton ok = new JButton("OK");
        bas.add(ok);
        JButton annul = new JButton("annuler");
        bas.add(annul);
        contenu.add(bas, BorderLayout.SOUTH);
        pack();
        setVisible(true);
    }
}
```

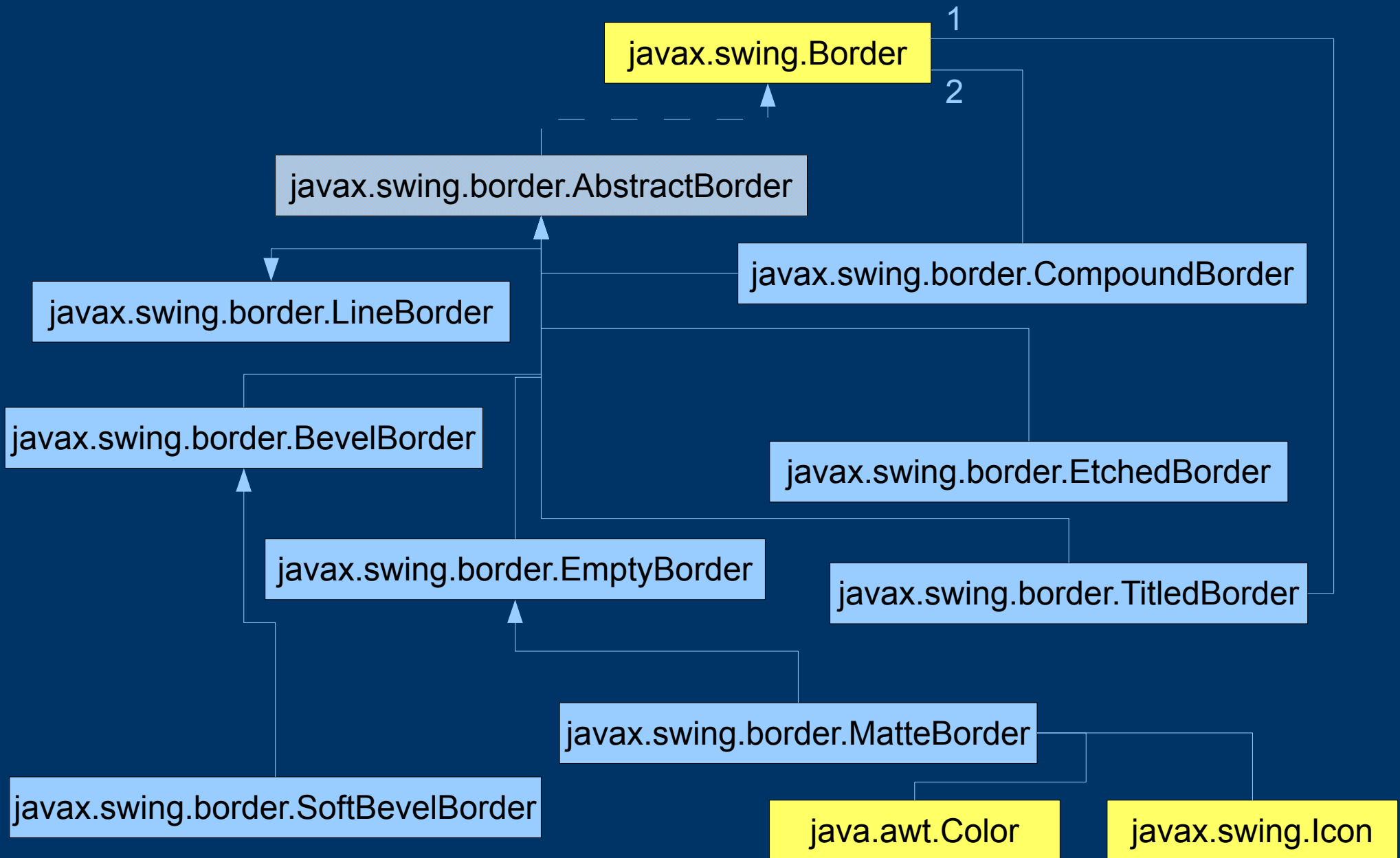
```
public static void main(String[] args) {
    JFrame fenetre = new
    ImbricationGestionnairesPresentation();

    fenetre.setDefaultCloseOperation(JFrame.
    EXIT_ON_CLOSE);
}
```

***Encadrements (classes implémentant
l'interface javax.swing.Border)***



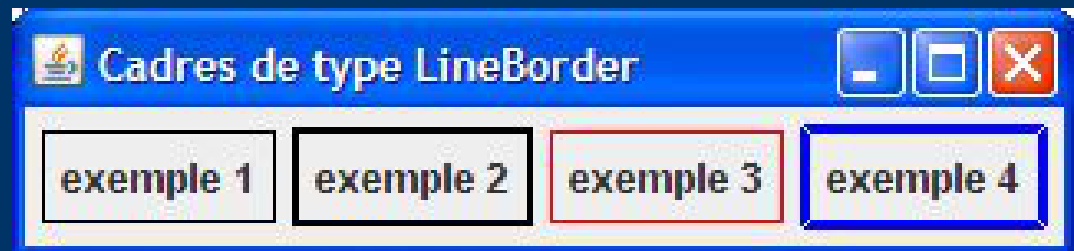
Les différents cadres



EmptyBorder

- Ajoute un cadre vide
 - Permet d'ajouter de l'espace autour d'un composants
- Constructeurs
 - `EmptyBorder(Insets insets)`
 - `EmptyBorder(int top, int left, int bottom, int right)`

LineBorder

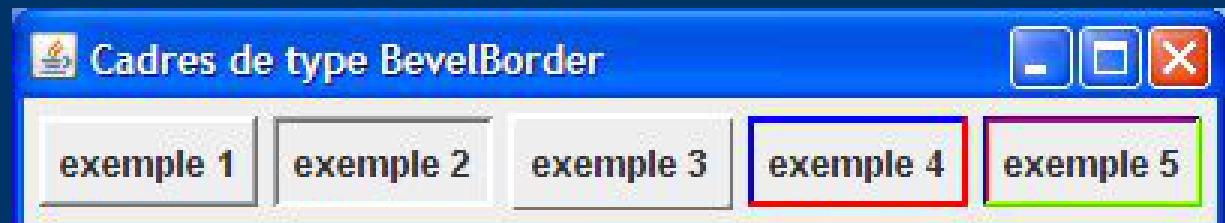


- Ajoute un cadre constitué d'une simple ligne
- Constructeurs
 - `LineBorder(Color coul)`
 - `LineBorder(Color coul, int epaisseur)`
 - `LineBorder(Color coul, int epaisseur, boolean coinsArrondis)`

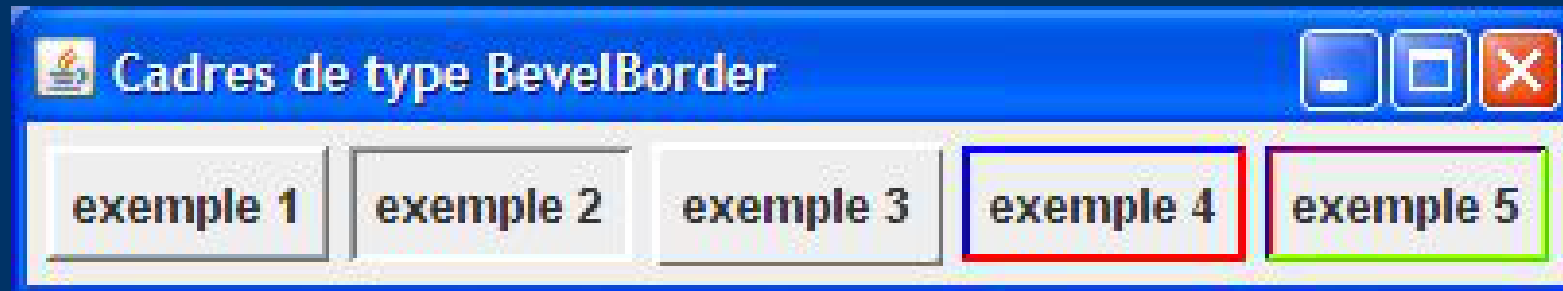
```
LineBorder lb1 = new LineBorder(Color.BLACK);  
LineBorder lb2 = new LineBorder(Color.BLACK, 2);  
LineBorder lb3 = new LineBorder(Color.RED);  
LineBorder lb4 = new  
LineBorder(Color.BLUE,3,true);
```

BevelBorder/SoftBevelBorder

- Ajoute un cadre avec effet marie-louise creux ou bosse (bevel = biseau)
- Constructeurs
 - BevelBorder(int type)
 - BevelBorder(int type, Color lumiere, Color ombre)
 - BevelBorder(int type, Color lumiereExterieur, Color lumiereInterieur, Color ombreExterieur, Color ombreInterieur)
- Variant Soft
 - Les coins sont adoucis
- Types : RAISED/LOWERED



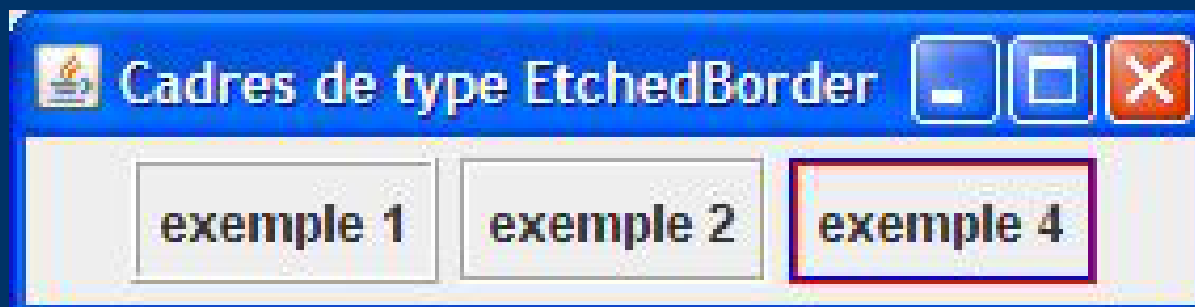
BevelBorder : code des exemples



```
Border lb1 = new BevelBorder(BevelBorder.RAISED);  
Border lb2 = new BevelBorder(BevelBorder.LOWERED);  
Border lb3 = new SoftBevelBorder(BevelBorder.RAISED);  
Border lb4 = new BevelBorder(BevelBorder.RAISED,Color.BLUE,Color.RED);  
Border lb5 = new  
BevelBorder(BevelBorder.RAISED,Color.BLUE,Color.RED,Color.GREEN,Color.YELLOW)  
;
```

EtchedBorder

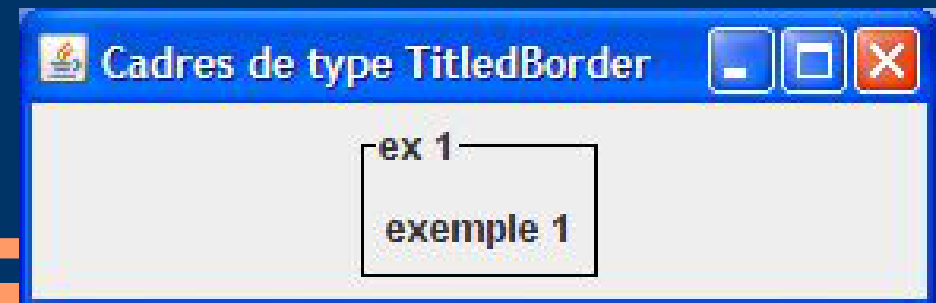
- Ajoute un cadre avec effet relief (creux ou bosse)
- Constructeurs
 - EtchedBorder()
 - EtchedBorder(Color lumiere, Color ombre)
 - EtchedBorder(int type)
 - EtchedBorder(int type, Color lumiere, Color ombre)
- Par défaut : en creux
- Types : RAISED/LOWERED



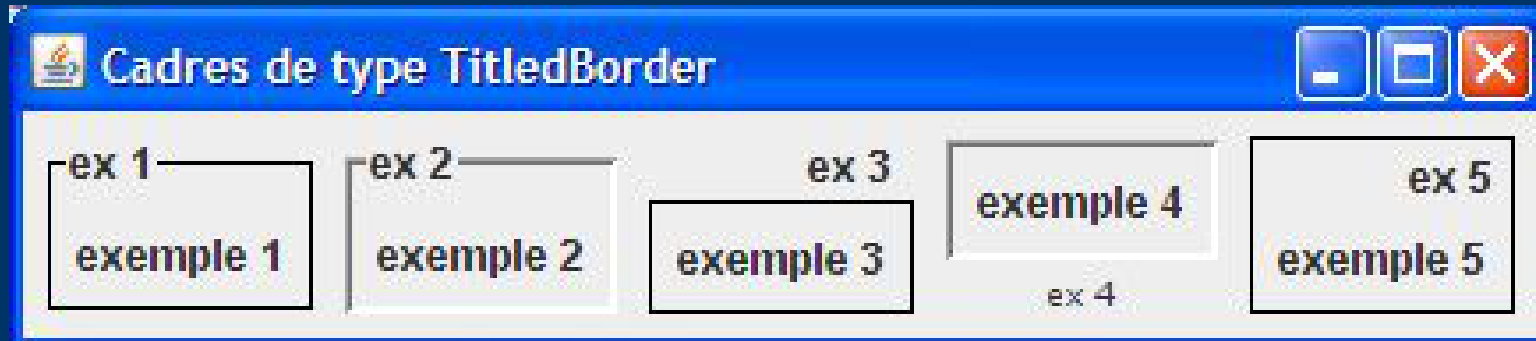
```
Border lb1 = new EtchedBorder(EtchedBorder.RAISED);  
Border lb2 = new EtchedBorder(EtchedBorder.LOWERED);  
Border lb4 = new  
EtchedBorder(EtchedBorder.RAISED, Color.BLUE, Color.RED);
```

TitledBorder

- Permet d'ajouter un titre à un autre cadre
- Constructeurs
 - TitledBorder(Border cadre)
 - TitledBorder(String titre)
 - TitledBorder(Border cadre, String titre)
 - TitledBorder(Border cadre, String titre, int titleJustification, int titlePosition)
 - TitledBorder(Border cadre, String titre, int titleJustification, int titlePosition, Font police)
 - TitledBorder(Border cadre, String titre, int titleJustification, int titlePosition, Font police, Color couleur)
- Positions possibles
 - ABOVE_TOP, TOP, BELOW_TOP
 - ABOVE_BOTTOM, BOTTOM, BELOW_BOTTOM
- Justifications possibles
 - LEADING, LEFT, CENTER
 - TRAILING, RIGHT



TitledBorder



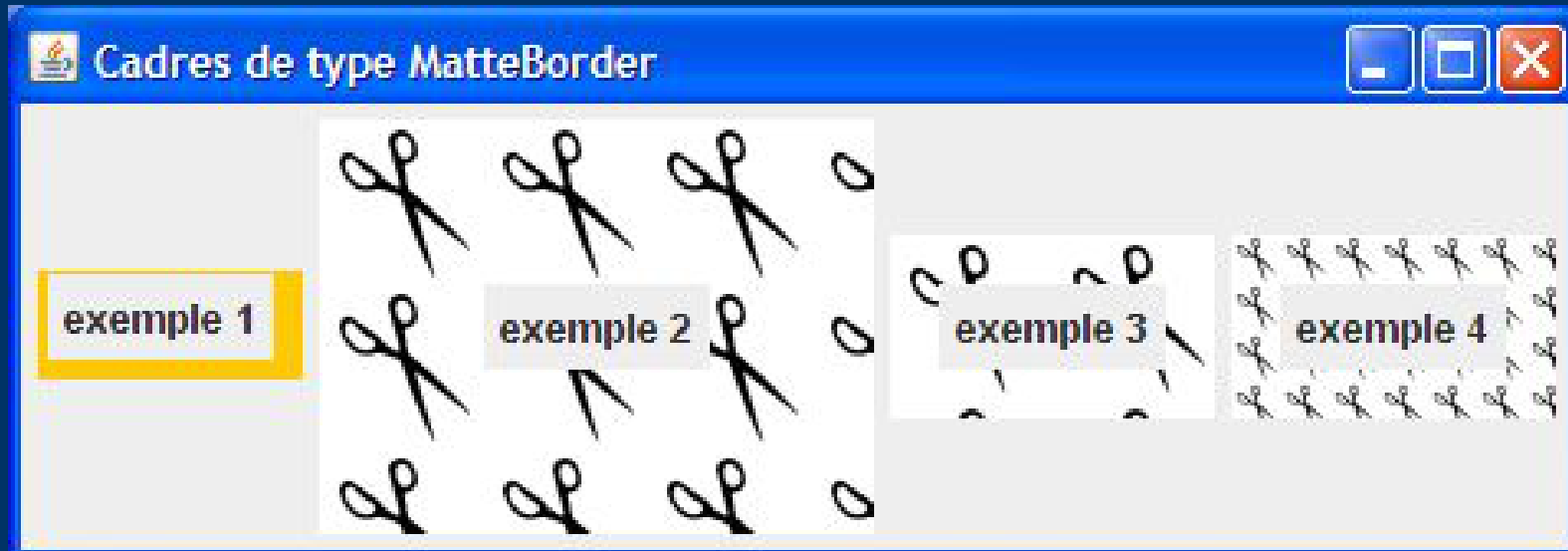
```
Border b1 = new LineBorder(Color.BLACK);
Border b2 = new BevelBorder(BevelBorder.LOWERED);
Border tb1 = new TitledBorder(b1, "ex 1");
Border tb2 = new TitledBorder(b2, "ex 2");
Border tb3 = new TitledBorder(b1, "ex 3",TitledBorder.RIGHT,TitledBorder.ABOVE_TOP);
Border tb4 = new TitledBorder(b2, "ex 4",
TitledBorder.CENTER,TitledBorder.BELOW_BOTTOM, new
Font("SansSerif",Font.PLAIN,10));
Border tb5 = new TitledBorder(b1, "ex 5",TitledBorder.RIGHT,TitledBorder.BELOW_TOP);
```

MatteBorder

- Permet d'ajouter un cadre constitué soit de la répétition d'une icône, soit d'une couleur
- Constructeurs
 - `MatteBorder(Icon icône)`
 - `MatteBorder(Insets insets, Icon icône)`
 - `MatteBorder(Insets insets, Color couleur)`
 - `MatteBorder(int top, int left, int bottom, int right, Icon icône)`
 - `MatteBorder(int top, int left, int bottom, int right, Icon icône)`



MatteBorder



```
Border lb1 = new MatteBorder(new Insets(1,3,6,9),Color.ORANGE);
Imagelcon im = new Imagelcon("c:\\Documents and
Settings\\Bruno\\Bureau\\ciseaux.gif");
Border lb2 = new MatteBorder(im);
Border lb3 = new MatteBorder(new Insets(15,15,15,15),im);
Imagelcon im2 = new Imagelcon(im.getImage().getScaledInstance(15, 15,
Image.SCALE_SMOOTH));
Border lb4 = new MatteBorder(new Insets(15,15,15,15),im2);
```


CompoundBorder

- Permet de créer un cadre formé de 2 autres cadres : un cadre intérieur et un cadre extérieur
 - N.B. : le résultat étant un cadre, il peut être de nouveau utilisé pour former un cadre composé
 - Constructeurs
 - `CompoundBorder()`
 - `CompoundBorder(Border externe, Border interne)`
 - Utilisations
 - Rajout de l'espace entre un composant et son cadre :
`CompoundBorder(new EmptyBorder(10,10,10,10), new LineBorder())`
 - Cadre à double ligne
 - Cadre à double relief
-
-

CompoundBorder : exemples



texte avec cadre Line



texte avec cadre Line + compound



texte avec double cadre



texte avec double relief

```
Border b2e = new LineBorder(Color.BLACK);
Border b2i = new EmptyBorder(10,10,10,10);
Border b2 = new CompoundBorder(b2e,b2i);
Border b3e = new LineBorder(Color.BLACK);
Border b3ie = new EmptyBorder(2,2,2,2);
Border b3ii = new LineBorder(Color.BLACK);
Border b3i = new CompoundBorder(b3ie,b3ii);
Border b3 = new CompoundBorder(b3e,b3i);
Border b4e = new
    EtchedBorder(EtchedBorder.RAISED);
Border b4i = new
    EtchedBorder(EtchedBorder.LOWERED);
Border b4 = new CompoundBorder(b4e,b4i);
```

Quelques remarques

- Eventuels problèmes avec les cadre

Si l'affectation d'un cadre à un composant pose problème, mettre le composant dans un JPanel et associer le cadre au JPanel en question

- Création de cadre

Pour minimiser les ressources, utiliser si possible les méthodes statiques de la classe BorderFactory qui essaieront de partager des instances des classes Border pour plusieurs composants



Bulles d'aide



Classe *javax.swing.ToolTip*

- Rôle
 - Définir une bulle d'aide qui apparaît sur un composant
- Utilisation de base
 - `composant.setToolTipText("texte de la bulle");`