Classes et Interfaces de SWING et AWT

class **JComponent**
superclasse de tous les composants swing excepté les top level
hérite de Container

apparence graphique :
void setBorder(Border) affecter une (nouvelle) bordure
void setForeground(Color) changer la couleur de premier plan
Color getForeground()
void setBackground(Color) changer la couleur de fond
Color getBackground()
void setOpaque(boolean) rendre le fond opaque ou transparent
boolean isOpaque()
void setCursor(Cursor) curseur lorsque la souris le survole
Cursor getCursor()
état :
void setToolTipText(String) active la bulle d'aide
boolean isShowing() détermine s'il est visible et affiché
void setEnabled(boolean) active/désactive son action (s'il en a)
boolean isEnabled()
void setVisible(boolean) rend le composant visible/invisible
boolean isVisible()
listeners :
void addMouseListener(MouseListener) écouteur d'événement souris
void removeMouseListener(MouseListener)
void addMouseMotionListener(MouseMotionListener) écouteur
d'événement mouvement de souris
void removeMouseMotionListener(MouseMotionListener)
void addKeyListener(KeyListener) écouteur d'événement du clavier
void removeKeyListener(KeyListener)
le repeindre :
Graphics getGraphics() donne son contexte graphique s'il est
visible, sinon null
void repaint() appel à le redessiner de façon asynchrone
void paintComponent(Graphics) méthode à surcharger, qui est
chargée de le redessiner
en tant que container (et s'il l'est) :
Component add(Component) lui ajoute un contenu
void remove(Component) l'enlève
sa géométrie :
void setLayout(LayoutManager) lui affecte un géométrie manager
void setPreferredSize(Dimension) spécifie une dimension souhaitée
Dimension getPreferredSize()
Dimension getSize() donne ses dimensions en pixel
int getWidth()
int getHeight()

abstract class **AbstractButton**
hérite de JComponent
sous-classes : JButton, JCheckBox, JMenuItem, JToggleButton,
JRadioButton

```
void setActionCommand(String)   affecte un nom de commande action
String getActionCommand()
void setText(String)            affecte le nom du bouton
String getText()
void setIcon(Icon)              affecte une image au bouton
void setPressedIcon(Icon)    image du bouton quand il est pressé
void setSelectedIcon(Icon)      image  quand il est sélectionné
void setRolloverIcon(Icon)      image quand il est survolé
boolean isSelected()       détermine si le bouton est sélectionné
void doClick()             réalise un clic par programmation
void addActionListener(ActionListener l) ajoute un écouteur des
          événements ActionEvent émis par le bouton
void removeActionListener(ActionListener l) retire l'écouteur de
          la liste des écouteurs du bouton
void addItemListener(ItemListener l) ajoute un écouteur des
           événements ItemEvent émis par le bouton
void removeActionListener(ActionListener l) le retire
```

class **JButton**
hérite de AbstractButton
JButton(String)   construit un bouton avec son texte qui est le
        nom de commande action par défaut
JButton(String, Icon)   ... avec une image

class **JCheckBox**
hérite de AbstractButton
JCheckBox(String)   construit une boite à cocher avec son texte
        qui est le nom de commande action par défaut
JCheckBox(String, boolean) le booléen détermine si la boite est
           cochée (sélectionnée)

class **JToggleButton**
hérite de AbstractButton
JToggleButton(Icon)   bouton à fonctionnement on/off

class **JRadioButton**
hérite de AbstractButton
s'utilise normalement avec un ButtonGroup
JRadioButton(String)   construit un bouton radio avec son texte
         qui est le nom de commande action par défaut

class **ButtonGroup**
ButtonGroup() construit un groupe pour boutons : la sélection des
      boutons se fait en exclusion
void add(AbstractButton) ajoute un bouton radio ou un
           JToggleButton au groupe

class JComboBox
hérite de JComponent

```
JComboBox()                    crée une liste de choix vide
JComboBox(Vector<?> items)     crée une liste à partir du Vector

void addItem(Object anObject) ajoute un choix
Object getItemAt(int index)   fournit le i-éme item de la liste
int  getItemCount()           donne le nombre d'items
int  getSelectedIndex()       donne l'index de l'item sélectionné
void setSelectedIndex(int anIndex)
Object getSelectedItem()
boolean isEditable()          spécifie si la liste choix est
            modifiable par l'utilisateur
void setActionCommand(String aCommand)      donne un nom d'action
String getActionCommand()
void setMaximumRowCount(int count) fixe le nombre max de choix vus

void addActionListener(ActionListener l)   ajoute un écouteur des
      événements ActionEvent émis par la liste de choix
void removeActionListener(ActionListener l) le retire de la liste
void addItemListener(ItemListener l)       ajoute un écouteur des
      événements ItemEvent émis par la liste de choix
void removeActionListener(ActionListener l) le retire de la liste
```

class **JTextField**
hérite de JTextComponent, JComponent

```
JTextField()              crée une zone de saisie de caractères
JTextField(String text)   avec une chaine de caractères donnée
JTextField(int columns)   avec une zone de n caractères visibles

void setText(String)      affecte le texte
String getText()          récupère le texte
void setColumns(int columns) fixe le nombre de caractères visibles
int getColumns()
void setEditable(boolean b)   modifiable par l'utilisateur ou non
boolean isEditable()
String getSelectedText() récupère de texte sélectionné à la souris
void replaceSelection(String content)   remplace le texte
                 sélectionné à la souris
void setCaretPosition(int position) modifie la position du curseur
               d'insertion
int  getCaretPosition()
void copy()       copie le texte sélectionné et le met dans le
       tampon Copier-coller du bureau
void cut()        idem mais couper
void paste()      coller ...

void addActionListener(ActionListener l) ajoute un écouteur des
  événements ActionEvent émis quand la touche "enter" est tapé
void removeActionListener(ActionListener l) le retire de la liste
```

abstract class **Graphics**
Graphics2D : en fait, on travaille sur cette sous-classe !

```
Color getColor()        donne la couleur
void setColor(Color c)  change la couleur
Font getFont()          donne la fonte, la police de caractère
void setFont(Font font) la change
Graphics create()       crée une copie
void drawString(String str, int x, int y) dessine un texte sur une
          ligne de base commençant au point donné
void drawRect(int x, int y, int width, int height) rectangle vide
void fillRect(int x, int y, int width, int height) rectangle plein
void drawLine(int x1, int y1, int x2, int y2)  dessine une ligne
void drawPolyline(int[] xPoints, int[] yPoints, int nPoints)
                dessine une ligne brisée
void drawPolygon(int[] xPoints, int[] yPoints, int nPoints)
                dessine un polygone
void fillPolygon(int[] xPoints, int[] yPoints, int nPoints)
                dessine un polygone plein
void drawOval(int x, int y, int width, int height) dessine un
          ovale dont le coin haut gauche est x y
void fillOval(int x, int y, int width, int height)  ovale plein
boolean drawImage(Image img, int x, int y, ImageObserver observer)
```

class **JFrame**
hérite de Container, de Window et Frame de AWT

```
JFrame()                construit une JFrame sans l'afficher
JFrame(String titre)    ... avec son titre

void setTitle(String titre)    affecte le titre
void setResizable(boolean resizable) modifiable par l'utilisateur
boolean isResizable()
void setLocation(int x, int y)   positionne le coin haut-gauche
             dans l'écran en x y
void toBack()            met la fenetre en arrière sur le bureau
void toFront()           en avant et reçoit le focus du clavier
void setContentPane(Container)  affecte un contenu son ContentPane
Container getContentPane()
void setJMenuBar(JMenuBar)  affecte une barre de menu à la fenetre
JMenuBar getJMenuBar()
void  setDefaultCloseOperation(int operation) affecte un
          comportement au bouton de fermeture
int getDefaultCloseOperation()
static int EXIT_ON_CLOSE    comportement arret de l'application
void pack()     calcule la forme optimale en commencant
      récursivement par les composants contenus
void dispose()  libère toutes les ressources de la fenêtre et de
      ses contenus
void addWindowListener(WindowListener l)  ajoute un écouteur des
  événements WindowEvent : iconifier, de-iconifier, fermer, ...
void removeWindowListener(WindowListener l) le retire de la liste
```

abstract class **Container**
classe de AWT héritant de Component
sous-classes : JComponent, JFrame

Component add(Component comp) ajoute le composant au container
Component add(Component comp, int index) ... à la position donnée
void add(Component comp, Object constraints) ... selon les contraintes données
void remove(Component comp) retire le composant du container
void remove(int index) retire le n-ième
boolean isAncestorOf(Component c) indique si le composant est contenu directement ou indirectement
int getComponentCount() retourne le nombre de composants contenus
Component getComponent(int n) retourne le n-ième composant contenu
Component[] getComponents() retourne un tableau des composants contenus par le container
Container getParent() retourne s container parent

void setLayout(LayoutManager layout) affecte un géométrie manager de réorganisation du contenu
LayoutManager getLayout()
void validate() provoque un calcul de sa géométrie et récursivement de ses contenus
void revalidate() provoque un appel validate() sur les parents du containers jusqu'à rencontrer un composant JScrollPane ou un composant top level comme JFrame ou JDialog

class **JPanel**
hérite de JComponent
JPanel() crée un JPanel avec une stratégie de rafraichissement par "bouble buffering" et un géométrie manager FlowLayout
JPanel(LayoutManager layout) ... en spécifiant le "layout"

class **JMenuBar**
hérite de JComponent
JMenuBar() crée une barre de menu
JMenu add(JMenu c) ajoute un menu à la barre

class **JMenu**
hérite de AbstractButton et JMenuItem
JMenu(String nom) contruit un menu au nom spécifié
JMenuItem add(JMenuItem menuItem) ajoute un menuItem au menu
void addSeparator() ajoute un séparateur graphique
Component add(Component c) ajoute un composant au menu

class **JMenuItem**
hérite de AbstractButton
JMenuItem(String nom) contruit un item de menu au nom spécifié

class **JCheckBoxMenuItem**
hérite de JMenuItem
JCheckBoxMenuItem(String nom) contruit un élément de menu cochable

class **JScrollPane**
hérite de JComponent
JScrollPane() crée une vue
JScrollPane(Component) crée une vue sur le composant
void setViewportView(Component view) affecte un composant vu

class **JSplitPane**
hérite de JComponent

JSplitPane(int orientation, Component, Component) crée un double volet sur les 2 composants selon l'orientation donnée
static int HORIZONTAL_SPLIT double volet orienté droite-gauche
static int VERTICAL_SPLIT double volet orienté haut-bas
void setContinuousLayout(boolean) indique si les 2 composants contenus sont redessinés pendant le déplacement de la barre de séparation de 2 volets
boolean isContinuousLayout()
void setDividerLocation(double) place la barre de séparation selon le pourcentage donné (double compris entre 0 et 1)
void setOneTouchExpandable(boolean) ajoute/enlève un bouton d'expansion d'un volet sur l'espace de l'autre
boolean isOneTouchExpandable()

class **JTabbedPane**
hérite de JComponent

JTabbedPane() crée un container à onglets
void addTab(String title, Component component) ajoute un composant : son onglet aura le titre indiqué
void setSelectedIndex(int index) affiche l'onglet d'index indiqué
int getSelectedIndex() donne l'index de l'onglet sélecté

class **FlowLayout**

FlowLayout() alignement CENTER par défaut et l'écartement vertical et horizontal entre contenu est de 5 pixels
FlowLayout(int align) spécifie l'alignement
FlowLayout(int align, int hgap, int vgap) ... et l'écartement vertical et horizontal
static int LEFT alignement à gauche
static int RIGHT, CENTER, ...

class **BorderLayout**

```
BorderLayout()                les contenus n'ont aucun écartement
BorderLayout(int hgap, int vgap)   avec écartement défini
static String NORTH           indique la zone Nord
static String SOUTH, EAST, CENTER, WEST, ....
```

class **GridLayout**

```
GridLayout(int l, int c) crée une grille de l lignes et c colonnes
GridLayout(int rows, int cols, int hgap, int vgap) avec un espace
                 vertical et un horizontal
```

class **Box**
hérite de JComponent
n'a que BoxLayout comme manager de placement

```
Box(int axis)   crée une Box avec une orientation axe des X ou Y
static Component createGlue() ajoute une zone vide auto-ajustable
static Component createHorizontalStrut(int width) ajoute une zone
                     de largeur fixe
static Component createVerticalStrut(int height)
static Component createRigidArea(Dimension d)  ajoute une zone de
                   dimension fixe
```

class **BoxLayout**

```
static int     X_AXIS placement horizontal de gauche à droite
static int     Y_AXIS, ...
```

class java.util.**EventObject**
sous-classes: java.awt.event.AWTEvent

```
Object getSource()  donne l'objet source de l'événement
```

class **ActionEvent**
hérite de AWTEvent

```
String getActionCommand()   retourne le nom de l'action commande
int getModifiers()    retourne l'état des touches modifiers
         alt, shift, ctrl, ...
```

class **KeyEvent**
hérite de AWTEvent

```
char getKeyChar()     donne le caractère associé à la frappe
int  getKeyCode()     retourne le code de la touche
static int VK_A       le code de la touche A
static int VK_0       le code de la touche 0
static int VK_CONTROL le code de la touche CTRL
static int VK_COMMA   le code de la touche ,
int getModifiers() état des touches modifiers alt shift ctrl ...
```

class **MouseEvent**
hérite de AWTEvent
int getX()   getY()   les positions
int getButton()   retourne les boutons enfoncés
static int BUTTON1   code du bouton1

interface **ActionListener**
void actionPerformed(ActionEvent) méthode appelée lors de l'action

interface **MouseListener**
void mouseEntered(MouseEvent) événement MouseEvent arrivée souris
void mouseExited(MouseEvent)   "sortie de souris"
void mousePressed(MouseEvent) "bouton pressé"
void mouseReleased(MouseEvent) "bouton relaché"
void mouseClicked(MouseEvent) "bouton pressé puis relaché dans
             sa zone graphique"
class **MouseAdapter** implements MouseListener

interface **MouseMotionListener**
void mouseMoved(MouseEvent)   événement MouseEvent "mouvement
            de souris"
void mouseDragged(MouseEvent) .... avec bouton enfoncé
class **MouseMotionAdapter** implements MouseMotionListener

interface **WindowListener**
void windowActivated(WindowEvent)
void windowClosed(WindowEvent)
void windowClosing(WindowEvent)
void windowDeactivated(WindowEvent)
void windowDeiconified(WindowEvent)
void windowIconified(WindowEvent)
void windowOpened(WindowEvent)
class **WindowAdapter** implements WindownListener

interface **KeyListener**
void keyPressed(KeyEvent)
void keyReleased(KeyEvent)
void keyTyped(KeyEvent)
class **KeyAdapter** implements KeyListener

javax.swing.S**wingUtilities**.invokeLater(Runnable)
   place la méthode run du Runnable dans la file d'attente
   du thread AWT-EventQueue

class **UIManager**
static void setLookAndFeel(String className) définit le rendu
  graphique de l'application : apparence et comportement