

ANNEXE
Chapitre 1 : Les interfaces graphique en Java

Classe	Méthodes
JPanel	JPanel pan = new JPanel (); // constructeur sans layout JPanel pan = new JPanel (new LayoutManager()); // crée un JPanel avec un layout manager (BorderLayout, FlowLayout, ...) Container getContentPane(); permet d'obtenir le panneau d'une fenêtre. JLayeredPane getLayeredPane(); permet d'obtenir le LayeredPane d'une fenêtre pour l'ajout de panneaux an couches superposées. add(component) : ajoute un composant au panel
JToolBar	JToolBar(int orientation) : crée une barre d'outil dont l'orientation est spécifié par une constante de la classe JToolBar (.VERTICAL, .HORIZONTAL). Component add (Component) : ajoute un composant à la barre d'outils. void addSeparator() : ajoute un séparateur à la fin de la barre. void setFloatable(boolean) et boolean isFloatable() : fixe ou permet de savoir s'il est possible de sortir la barre dans une fenêtre séparée pour créer une palette flottante.
JScrollPane	JScrollPane() : crée un JScrollPane vide. JScrollPane(Component comp) : crée un JScrollPane avec un seul composant passé en paramètre. JScrollPane(int vertical, int horizontal) : crée un JScrollPane en précisant le comportement des ascenseurs. JScrollPane(Component comp, int vertical, int horizontal) : crée un JScrollPane avec un composant en précisant le comportement des ascenseurs. Le comportement des ascenseurs peut être une constante de la classe JScrollPane: .VERTICAL_SCROLLBAR_ASNEEDED : apparaît en cas de besoin .VERTICAL_SCROLLBAR_NEVER : toujours absent .VERTICAL_SCROLLBAR_ALWAYS : toujours présent Même chose pour HORIZONTAL
JSplitPane	JSplitPane(int orientation) : le paramètre est une constante de la classe JSplitPane : .VERTICAL_SPLIT ou .HORIZONTAL_SPLIT. JSplitPane(int orientation, Component comp, Component comp) : construit un JSplitPane avec une indication d'orientation et les deux composants à y placer. Component getXxxComponent() : retourne le composant gauche(Left), droite (Right), de haut (Top), ou d'en bas (Bottom). int getDividerSize() : retourne la taille en pixel de la séparation. void setDividerSize(int) : définit l'épaisseur de la barre de séparation. int getDividerLocation() : retourne la position de la séparation en pixels. void setDividerLocation(double) : définit la position du séparateur en la représentant par un pourcentage de 0.0 à 1.0
JTabbedPane	JTabbedPane(int) : crée l'objet en précisant la position des onglets. L'entier est une constante de la classe JTabbedPane (.TOP, .BOTTOM, .LEFT, .RIGHT) void addTab (String, Component) : ajoute un onglet avec un libellé de type String et y place un composant void addTab (String, ImageIcon, Component[, String]) : ajoute un onglet avec les paramètres dans l'ordre le libellé, l'image de l'icône, le composant à placer, une bulle d'aide.

ANNEXE
Chapitre 1 : Les interfaces graphique en Java

	void insertTab (String, ImageIcon, Component [, String], int) : même chose que addTab avec en plus la position d'insertion de l'onglet come dernier paramètre. void removeTabAt(int) : supprime l'onglet désigné par son numéro d'index. void setIconAt(int, ImageIcon) : associe une icône à l'endroit désigné. int getSelectedIndex() : retourne le numéro de l'onglet sélectionné. void setSelectedIndex(int) : sélectionne l'onglet dont le numéro est sélectionné int getTabCount() : retourne le nombre d'onglets. int indexOfTab(String) : retourne le premier onglet associé au nom passé en paramètre void setTitleAt(int, String) : définit le libellé à la position désignée. void setEnabledAt(int, boolean) : rend accessible ou pas l'onglet désigné.
JMenuBar	JMenuBar() : crée une nouvelle barre de menu. void add (JMenu) : ajoute une nouvelle rubrique dans la barre de menu. JMenu getMenu(int) : retourne la rubrique dont le numéro est spécifié.
JMenu	JMenu(String) : crée et définit le texte de la rubrique. JMenuItem add (JMenuItem) : ajoute une sous rubrique. Les sous rubriques sont ajoutés dans l'ordre de leurs insertions. La valeur retournée est le JMenuItem. JMenuItem add (String) : crée et ajoute une sous rubrique à partir d'une chaîne. void addSeparator() : place un séparateur entre sous rubriques. void insert (JMenuItem, int) : insère un JMenuItem à un rang spécifique. void insertSeparator(int) : insère un séparateur entre sous rubriques à un rang spécifique.
JMenuItem	JMenuItem(String) : crée et définit le texte de la sous-rubrique JMenuItem(ImageIcon) : crée une sous rubrique avec une icône JMenuItem(String, ImageIcon) : crée une sous rubrique avec définition du texte et d'une icône. void setEnabled (Boolean) : active ou désactiver une sous-rubrique. void setText(Stirng) : définit le texte du menu void setAccelerator(KeyStroke clé) : définit la combinaison de clé qui invoque le menu. void setMnemonic(int) : définit la lettre du texte du menu qui sera combiné avec la touche Alt pour activer le menu.
JLabel	JLabel() : construit une étiquette vide JLabel(Icon icone[, int alignement]) : crée un label contenant une image avec alignement. JLabel(String [Icon icone ,int alignement]) : construit un label avec son contenu textuel, une image et un alignement. La valeur de l'alignement est une constante statique : CENTER, LEFT, RIGHT. void setText(String) : définit le texte à afficher sur le label. String getText() : retourne le texte affiché sur le label.

ANNEXE
Chapitre 1 : Les interfaces graphique en Java

JButton	JButton(String[, ImageIcon]) : crée un bouton avec définition du texte et/ou d'une icône dans le bouton. String getText() : retourne le texte contenu dans le bouton. void setText(String) : définit le texte contenu dans le bouton.
JCheckBox	JCheckBox(String[, ImageIcon, boolean]) : construction d'une case à cocher avec du texte et éventuellement une icône. Le boolean permet de définir l'état de la case à cocher (sélectionnée ou pas). String getText() et void setText(String) : pour retourner le texte et définir le texte de la case à cocher. boolean isSelected() et void setSelected(boolean) : pour retourner et définir l'état de la case à cocher (cochée ou non).
JRadioButton	JRadioButton (String [, ImageIcon, boolean]) : construire un bouton radio, comportant éventuellement un texte et une icône. Le booléen permet de spécifier si la case radio est cochée ou pas. Par défaut elle ne l'est pas. boolean isSelected() et void setSelected(boolean) : pour retourner et définir l'état du bouton radio.
ButtonGroup	ButtonGroup() : crée un nouveau groupe de boutons mutuellement exclusifs. void add (AbstractButton) : ajoute un nouveau bouton / bouton-radio au groupe. void remove (AbstractButton) : supprime un bouton/ bouton-radio du groupe. ButtonGroup clearSelection() : remet l'état des boutons sélectionnés dans le groupe à l'état non sélectionné. Aucun bouton du groupe n'est sélectionné
JList	JList(Object[]) : construit une liste statique à partir d'un tableau d'objets (tel que String). void setListData(Object[]/Vector) : fixe le contenu de la liste à partir d'un tableau ou vecteur d'objets. void setVisibleRowCount(int) : définit le nombre d'éléments visibles sans ascenseurs. int getSelectedIndex() : retourne le numéro du premier élément sélectionné. int[] getSelectedIndices() : retourne les numéros des éléments sélectionnés. Object getSelectedValue() : retourne le premier objet sélectionné. L'objet retourné doit être converti car de type Object. Object[] getSelectedValues() : retourne les objets actuellement sélectionnés. void setSelectedIndex(int) : sélectionne l'élément désigné par son numéro. void setSelectedIndices(int[]) : sélectionne plusieurs éléments désignés par leurs numéro void clearSelection() : annule toutes les sélections. void setSelectionMode(int mode) : permet de fixer le mode de sélection des éléments dans la liste. La valeur du mode est une constante statique qui peut être : SINGLE_SELECTION pour une sélection unique, SINGLE_INTERVAL_SELECTION pour une sélection contiguë unique et MULTIPLE_INTERVAL_SELECTION pour des sélections quelconques. JList(ListModel dataModel) : crée une liste dynamique associé à un modèle qu'il faut instancier auparavant. On peut utiliser la méthode DefaultListModel() pour en définir un.

ANNEXE
Chapitre 1 : Les interfaces graphique en Java

JComboBox	JComboBox([Object[]]) : crée un combobox avec éventuellement une liste d'objets (par exemple des String). void addItem(Object) : ajoute une valeur possible de choix. void insertItemAt(Object objet, int index) : ajoute un élément à la fin de la liste des choix ou à une position spécifiée par l'index. int getSelectedIndex() : retourne le numéro de choix actuel. Object getSelectedItem() ou Object getItemAt(int index) : retourne l'objet associé au choix actuel ou à son index. Penser à la conversion de cet objet à sa classe d'origine. void setSelectedIndex(int choix) : sélectionne un élément définit par son numéro. int getItemCount() : Récupère le nombre d'éléments dans la liste de choix. void removeAllItems() ou void removeItemAt(int anIndex) ou void removeItem (Object anObject) : Permet de supprimer des items de la liste de choix. void setMaximumRowCount(int count) : fixe le nombre maximum d'éléments à afficher simultanément. Si le nombre total est supérieur à ceux autorisés, la liste est munie automatiquement d'ascenseurs. void setEditable (boolean) : permet de spécifier une liste de choix éditable ou pas. Lorsqu'elle est éditable, la liste se transforme en une zone de texte où l'utilisateur peut saisir son texte et a côté un petit bouton pour dérouler les choix possibles.
JTextField JPasswordField JTextArea	JTextField([String[,int]]) : crée un champ de texte avec éventuellement un contenu initial, et le nombre de colonnes. JPasswordField([String[,int]]) : crée un champ de saisie pour mot de passe. char[] getPassword() : retourne le mot de passe saisie par l'utilisateur. void setEchoChar(char c) : définit le caractère affiché à la place de celui entré par l'utilisateur. JTextArea(String[, int, int]) : crée une zone de texte avec un contenu initial et précise éventuellement le nombre de lignes et de colonnes de la zone. void append(String) : ajoute la chaîne à la fin du texte affiché. void insert (String, int) : insère la chaîne au texte affiché à la position du rang. void setTabSize(int) : définit la distance entre tabulation void setLineWrap(boolean) : détermine si les lignes longues doivent ou non être repliées. String getText() et void setText(String) : pour retourner et définir le texte dans la zone. void setEditable(boolean) : rend la zone éditable ou non. String getSelectedText() : retourne le texte sélectionné. void select(int, int) : sélectionne le texte compris entre deux positions données. void selectAll() : sélectionne tout le texte.