import javax.swing.*;(swing.event.*) ;
import java.awt.*;
import java.awt.event.*;

class Fenetre extends JFrame
class Ecouteur implements ActionListener

- public void actionPerformed(ActionEvent ev) ;
- ev.getSource (composant sur lequel le clic est effectué) ou getActionCommand()
- setVisible(boolean) ; setTitle(String);setSize(int, int),….
- getContentPane() → setLayout(new FlowLayout()) , add(composant,String direction)
- composant.addActionListener(Ecouteur)

Jpanel:

- public void paintComponent(Graphics g) {super.paintComponent(g)
- repaint() → appel à PaintComponent
- validate() → recalcul des positions lors de l'ajout d'un composant
- Graphics g = this.getGraphics() puis dispose() → lors d'un dessin à la volée, liberer la mémoire

Dimension
Toolkit tk = getDefaultToolkit() ; Dimension tk.getScreenSize(), .width et .height
Jbutton(String)
setPreferedSize(Dimension) ; revalidate() ;
JcheckBox(String(,boolean))

- boolean isSelected()
- public void setSelected() ;
- Implements ItemListener → public void itemStateChanged(ItemEvent ev)

JRadioButton(String color(,boolean))
ButtonGroup groupe = new ButtonGroup() ; groupe.add(new JradioButton(String color)))
Jlabel(String) ;
setText(String)
JtextField(int taille)

- setEditable(boolean) ; setColumns(int) → JtextField.revalidate() ;
- interface FocusListener → focusGained(FocusEvent ev) ;focusLost(FocusEvent ev);
- getText() et setText(String)
- interface DocumentListener → insert/remove/changeUpdate(DocumentEvent e) ;

Jlist(String[])

- setSelectionMode(SINGLE_SELECTION, SINGLE_INTERVAL_SELECTION, MULTIPLE_INTERVAL_SELECTION)
- JscrollPane(Jlist) → add container
- setVisibleRowCount(int) → nombre de valeurs affichées à l'ecran
- Objet[] valeurs = liste.getSelectedValues() → copie des valeurs selectionées
- int getSelectedIndex() → index premiere valeur selectionée
- int[] getSelectedIndices() → tableau index valeurs selectionéee
- interface ListSelectionListener, void valueChanged(ListSelectionEvent e) ;
  if(!e.getValueIsAdjusting())

JcomboBox

setEditable(boolean) ; setMaximumRowCount(4) ; setSelectedIndex(int)

getSelectedItem()/Index()

addItem(String), addItemAt(String, int), removeItem(String)

JTextArea

Prend une chaine de caractère en argument pour afficher des methodes toString();

Boites de dialogue

JoptionPane

-showMessage(fenetre(ou null), String) ;

-JOptionPane.ERROR/INFORMATION/WARNING/QUESTION/PLAIN_MESSAGE → Constante d'affichage

-showConfirmDialog(fenetre,String) ; (Yes : 0), (No : 1), (Cancel : 2), (Closed : -1)

-showInputDialog(fenetre,Texte, Titre, Icone) → fenetre avec champ texte

-showInputDialog(fenetre,texte,titre,icone,iconesupp,String[],String[index]) ; → fenetre avec boite combo

-showOptionDialog(fenetre, texte, titre, icône, iconesupp,String[],String[index]) ; → fenetre avec String.length boutons

-dispose() et setVisible(boolean) pour l'affichage et la libération de la boite

Dialogue personnalisés

methode LanceDialogue pour recuperer les infos en sortie de boite de dialogue

private boolean ok.

ok=false, setVisible(true) -> actionPerformed -> si bouton ok alors ok=true et setVisible(false) sinon setVisible(false) uniquement, puis dans lanceDialogue, if(ok) ajout dans infos des paramètres.

class fenetre -> actionPerformed -> nouveau dialogue et info. Paramètres à echanger à mettre dans info, appel lancedialogue, puis mise à jour des paramètres.

Class Info pour les informations à echanger avec la boite de dialogue

Transparence :

JFrame.setDefaultLookAndFeelDecorated(true) puis setOpacity ou setBackground(new Color(...));

Menus

JMenuBar, JMenu, JMenuItem

fen.setJMenuBar(JMenuBar) → ajout fenetre

JMenu couleur = new JMenu(« Couleur ») crée menu de nom Couleur

JMenuBar objet.add(couleur) → ajout à l'objet JMenu

JMenuItem rouge = new JMenuItem("Rouge") → crée option de nom Rouge

couleur.add(rouge); ajout au menu couleur

addSeparator() -> ajoute barre separatrice entre deux options

on peut ajouter des JCheckBoxMenuItem, JRadioButtonMenuItem (à ajouter dans des ButtonGroup)

JPopMenu(); comportement similaire aux JMenuBar

.show(fen,int x, int y) -> affiche le menu dans la fen au couple de coordonnée (x,y)

public void mouseReleased (MouseEvent ev){if (ev.isPopupTrigger()) couleur.show(fen,ev.getX(),ev.getY());

e.getComponent -> reference de la fenetre concernée dans une classe anonyme

---

setMnemonic(Char) -> Ajoute raccourcis clavier à un JMenu ou JMenuItem

setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VR_LETTRE,InputEvent.CTRL_MASK)

composant.setToolTipText(String) -> Ajoute une bulle d'aide au composant

---

On peut ajouter des JMenu à des JMenu ou des JPopMenu

setEnabled -> active/desactive des JMenuItem

AbstractAction

- Créer une classe personnalisée heritant de AbstractAction avec au moins appel super(String) ajout de l'action à un JMenu (pas necessaire d'ajouter un JMenuItem)
- Redefinir la méthode ActionPerformed pour un JMenuItem
- Pour un bouton -> JButton object = new JButton((String)actionObject.getValue(Action.NAME))
- object.addActionListener(actionObject)
- actionObject.putValue(NAME/SMALL_ICON/SHORT_DESCRIPTION/LONG_DESCRIPTION);

JToolBar();

- add(JButton)
- getContentPane().add(JToolBar)
- setFloatable(boolean)
- ImageIcon(String Path), JButton(ImageIcon(Str))

- actionObject.putValue(Action.SMALL_ICON,String Path);

Autre : JToggleButton et JProgressBar

Evenements de bas niveaux

Pour un MouseEvent :

- getClickCount() retourne le nombre de clic de souris sous un court délai ou sans deplacement souris.
- getModifiers & InputEvent.BUTTON1/2/3_MASK != 0 -> touche de souris utilisée
- MouseMotionListener -> mouseDragged et mouseMoved

Pour un KeyEvent (implements KeyListener) -> keyPressed, keyReleased, keyTyped
getKeyChar, getKeyCode :
VK_0 à 9, VK_A à Z
VK_NUMPAD0 à 9
VK_F1 à 24
ALT(_GRAPH)
CAPS_LOCK,CONTROL,DELETE,DOWN,END,ENTER,ESCAPE,HOME,INSERT,LEFT,RIGHT,NUM_LOCK,PAGE_DOWN/UP/PRINTSCREEN,SCROLL_LOCK,SHIFT,SPACE,TAB....
methodes is<nomTouche>Down (Alt,Control,Shift). getModifiers()&InputEvent.<Nom>_MASK pour le test dans les methodes d'ecouteur.
Capture par des actions :
JFrame.registerKeyboardAction(action,
KeyStroke.getKeyStroke(KeyEvent.VK_<NOM>,InputEvent.<NOM>_MASK (|un autre)),
JComponent.WHEN_FOCUSED(ou autre))

WindowListener:

```
addWindowListener(new WindowAdapter(){
                        public void windowClosing(WindowEvent e){System.exit(0);}});
compo.requestFocus() // force le focus sur le composant compo
```

Gestionnaires de mise en forme

On peut utiliser setLayout sur le conteneur d'une fenêtre ou sur un JPanel

BorderLayout : place suivant les bords du conteneur, les constantes sont North,South,West,East,Center Constructeur vide ou avec 2 entiers qui determine l'espace entre les composants ajoutés (pixels horizontaux et verticaux). Methodes setH/Vgap(int). taille des composants non respectée

FlowLayout : dispose selon une même ligne, taille des composants respectée et depend de la nature et le contenu du composant. Constructeur vide ou constantes LEFT,RIGHT,CENTER et entiers comme pour BorderLayout

CardLayout : dispose suivant une pile, seul le composant supérieur est visible à un moment donné.
conteneur.add(compo,String), string indispensable avec ce layout.
affichage composant : pile.next/previous/first/last(conteneur) et pile.show(conteneur,String de add)

GridLayout : dispose suivant une grille réguliere, chaque composant occupe une cellule
Constructeur à 2 ou 4 paramètres : GridLayout(NB_LIGNES,NB_COLONNES(,int espaceH, int espaceV));

BoxLayout : dispose suivant une seule ligne/colonne avec un conteneur Box
Box ligne = *Box.createHorizontalBox*(); box colonne = box.*createVerticalBox*();
add(Box.*createVert/HorizontalStrut(*int)) ou *add*(Box.*CreateGlue*()); -> espacement variable ou maximal

GridBagLayout : A l'instar de GridLayout il dispose suivant une grille mais avec possibilité pour les composants d'occuper plusieurs cases.
Constructeur sans argument. Methode :

1. tableaux de coordonnées x,y(coin superieur gauche du composant), largeur, hauteur, poids horizontal, poids vertical (gridx/ywidth/height,weightx/weighty).
2. fill : Manière dont le composant occupe l'espace disponible. -> Vert/horizontal/Both/None
3. GridBagConstraints c = new GridbagConstraints(); c.fill=new GridbagConstraints.BOTH(exemple)
4. for(i) {c.grid/weight = tab[i]; contenu.add(composant,c);

GroupLayout : Constructeur à un argument : le conteneur. GroupLayout ges = new GroupLayout(conteneur)
description ordonnale d'un groupe :
GroupLayout.SequentialGroup hg = ges.createSequentialGroup();
GroupLayout.ParallelGroup hv = ges.createSequentialGroup();.
ges.setVerticalGroup(hv)
ges.setHorizontalGroup(hg)
Ajouter un composant : hg.addComponent(composant)
IL faut donc un groupe horizontal et un vertical et decider de la disposition sequentielle ou parallele.

Texte et Graphiques

Position du texte : FontMetrics fm = g.getFontMetrics();
fm.stringWitdth("Bonjour") -> longueur en pixel; fm.getHeight(); -> hauteur en pixel
Couleur du texte : setForeground(Color)
Information de fonte : getAscent, getDescent et getLeading : Jambage et Interligne
Fontes logiques : Font.PLAIN/BOLD/ITALIC avec concaténation. (Sans)Serif, Monospaced, Dialog(Input)
new Font(nom, constantes, taille);
Fontes physiques : String [ ] fontes =
GraphicsEnvironmnet.getLocalGraphicEnvironmnent.getAvailableFontFamilyNames();
Couleurs : *brighter()* et *darker()*
Tracés de lignes : g.translate pour changer les coordonnées de l'origine du panneau

Pour le remplissage, on trace la forme puis la bordure ensuite

Graphics2D

Gestion de l'epaisseur du trait et du motif (traits discontinus); methode setStroke d'objet basicStroke tracés de courbes de Béziers.

Dans paintComponent : Graphics2D g2d = (Graphics2D) g;

Mode de dessin :

Superposition de deux formes -> g.setXORMode(getBackground())

Images

Chargement d'une image avec attente :

- ImageIcon imIc = new ImageIcon(path)
- Image im = ImIc.getImage(); g.drawImage(im,x,y,null)
- getIconHeight/Width() -> renvoi hauteur/largeur de l'image

Chargement d'une image sans attente :

- getImage de Toolkit (fichier local) et Applet(site distant).
- dans drawImage, le quatrième argument doit être this.

Applet

```
<HTML>
	<BODY>
		<APPLET CODE = "<NOM>.class"; WIDTH = ...  HEIGHT = ...>
		</APPLET>
	</BODY>
</HTML>
```

Une Applet peut-être vue comme une JFrame, on peut lui ajouter des panneaux grâce à *getContentPane*

Dans <APPLET, les balises <PARAM <NOM>, VALUE = "<NOM>" permettent de passer des paramètres en ligne de commande.

On les récupere dans la méthode init() en utilisant getParameter(PARAM) ou PARAM est le nom du paramètre. width et height sont egalements de paramètres.