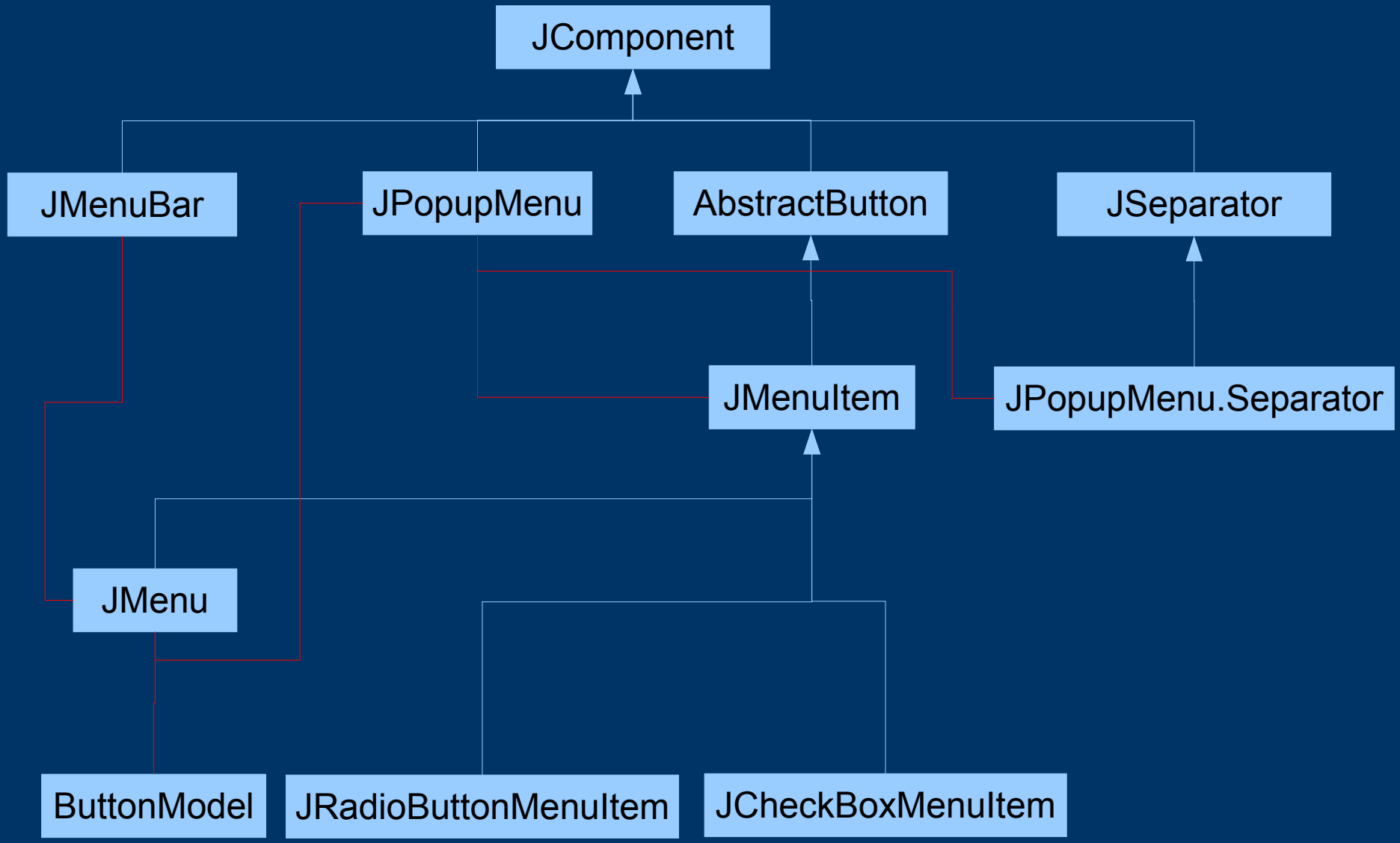
Les menus

Aperçu de l'API



Menus : conseils de présentation

Si possible :

- Pas plus de 10 items par menu
- Pas plus de 3 niveaux d'imbrication
- Limiter les items de type bouton radio ou case à cocher (peu intuitifs)
- Classer les items à l'intérieur des menus
- Utiliser les séparateurs

Menus
classe JMenuBar

- Constructeur : JMenuBar()
- Méthodes essentielles
  - void add(JMenu menu) / void setHelpMenu(JMenu m)
- Ajouter une barre de menu à une fenêtre
  - fenetre.setJMenuBar(JMenuBar maBarre)
- Ajouter un menu à une barre de menu
  - barre.add(JMenu menu)
- Supprimer un menu d'une barre de menu
  - barre.remove(Component menu)

Menus
Classe JMenu

- Rôle

  Créer un menu déroulant
  - Soit comme menu d'une barre (barre.add(JMenu menu))
  - Soit comme sous-menu d'un menu (menu.add(JMenuItem sousMenu)
- Constructeurs
  - JMenu(), JMenu(Action a)
  - JMenu(String s), JMenu(String s, boolean tear-off)
- Ajouter un article au menu
  - Component add(String s), Component add(Component c)
  - JMenuItem insert(String s, int pos), JMenuItem insert(JMenuItem m, int pos)
  - void addSeparator(), void insertSeparator()
- Supprimer un article de menu
  - remove(Component), remove(JMenuItem),
  - remove(int pos), removeAll()

Menus
Détection de la sélection

- Principe

  Ajouter un ActionListener au composant retourné par la méthode *add* de la classe JMenu

- Exemple

```
public class testMenuSuppression extends JFrame {
   private boolean edite;private JMenu fichier,editer,sousMenu; private JMenuBar barre;
   public testMenuSuppression() {
      super("suppression menu");
      barre = new JMenuBar();fichier = new JMenu("Fichier"); editer = new JMenu("Editer");
      sousMenu = new JMenu("Sous-Menu"); sousMenu.add("option 1");sousMenu.add("option 2");
      sousMenu.addSeparator(); JRadioButtonMenuItem b1 = new JRadioButtonMenuItem("Homme");
      JRadioButtonMenuItem b2 = new JRadioButtonMenuItem("Femme");
      ButtonGroup g = new ButtonGroup(); g.add(b1);g.add(b2);
      sousMenu.add(b1);sousMenu.add(b2) ;b1.setSelected(true); editer.add(sousMenu); edite = false;
      JMenuItem edition = new JMenuItem("edition");
      edition.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent ae) {barre.setVisible(false);
            if (edite) {edite = false;barre.remove(editer);} else {edite = true;barre.add(editer);}
            barre.setVisible(true);
      }});
      fichier.add(edition); barre.add(fichier); setJMenuBar(barre); pack(); setVisible(true);
}}
```

Menus
Les menus Pop-Up

- Principe

  Menu qui apparaît lors d'un certain clic sur un composant
- Mise en oeuvre
  - Créer le menu
  - Ajouter un écouteur de souris sur le composant
  - Pour les méthodes appropriées de l'interface MouseListener, tester si l'événement correspond bien à la demande d'un menu popup (méthode isPopupTrigger() de MouseEvent) en fonction du L&F
  - Si oui, afficher le menu au bon endroit

Menus
Les menus Pop-Up : exemple

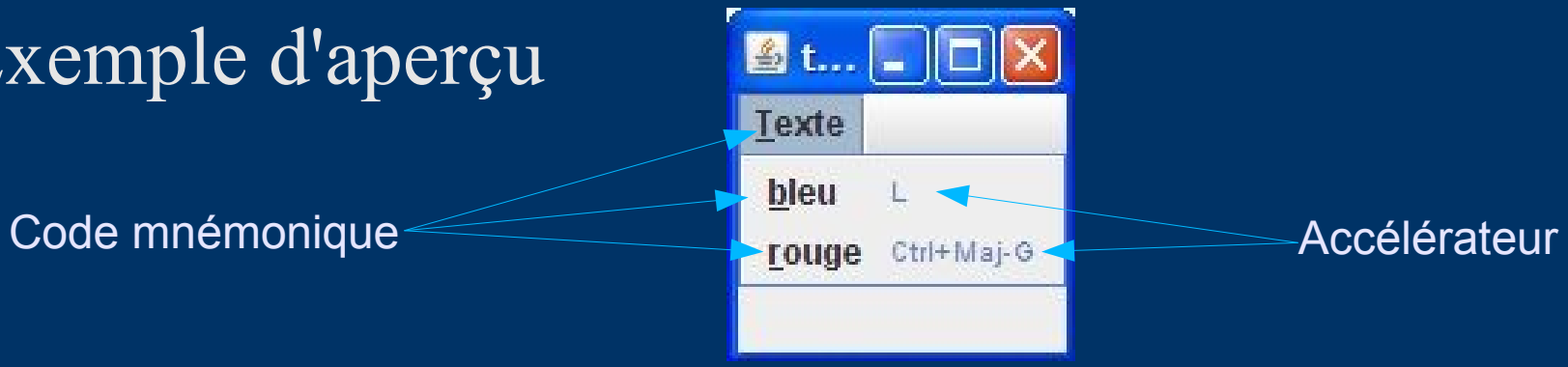
```
public class MenusPopup extends JFrame {
    private JPopupMenu popup;private JLabel etiquette;
    public MenusPopup() {
        super("Exemple pop-up");etiquette = new JLabel("Coucou");popup = new JPopupMenu();
        JMenuItem quitter = new JMenuItem("Quitter"); quitter.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {System.exit(0);}});
        JMenuItem opaque = new JMenuItem("Opaque");opaque.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {etiquette.setBackground(Color.YELLOW);
etiquette.setOpaque(true);}});
        JMenuItem trans = new JMenuItem("Transparent");trans.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {etiquette.setOpaque(false);etiquette.updateUI();}});
        popup.add(quitter);popup.add(opaque);popup.add(trans);
        etiquette.addMouseListener(new EcouteurPopUp());
        add(etiquette);
        pack();setDefaultCloseOperation(EXIT_ON_CLOSE);setVisible(true);
    }
    class EcouteurPopUp extends MouseAdapter {
        private void gerePopup(MouseEvent me) {
            if (me.isPopupTrigger()) {popup.show(etiquette, etiquette.getX(), etiquette.getY());}
        }
        public void mousePressed(MouseEvent me) {gerePopup(me);}
        public void mouseReleased(MouseEvent me) {gerePopup(me);}
    }
}
```

Raccourcis Claviers :

Codes mnémoniques
et
Accélérateurs

Codes mnémoniques et raccourcis
Introduction

- Code mnémonique
  Touche dont la combinaison avec une autre (alt en général) permettra d'activer une option de menu ou un bouton sans la souris lorsque le menu est affiché
- Accélérateur clavier
  Combinaison de touche permettant d'activer immédiatement une option d'un menu (ou le clic sur un bouton), même si aucun menu n'est déroulé
- Exemple d'aperçu



Utilisation des raccourcis claviers

- Souris/Clavier
  - Souris
    - Intuitif, découverte automatique
    - Exécution lente
  - Clavier
    - A priori peu intuitif, nécessite de faire travailler sa mémoire
    - Exécution rapide
- Utilisation typique d'un logiciel avec IHM graphique
  - Phase de découverte : 100% souris
  - Phase d'apprentissage : part souris diminue, part clavier augmente
  - Phase d'utilisation intensive : part clavier majoritaire
- Conclusion
  - Les deux modes souris/clavier doivent cohabiter
  - Les raccourcis claviers doivent être aussi simples et intuitifs que possible

Codes mnémoniques

- Composants visés
  - Boutons :
    - fait partie du modèle ButtonModel
    - Facilité d'accès depuis AbstractButton
  - Composants effectifs
    - JButton, JCheckBox, JRadioButton
    - JMenuItem, JMenu, JCheckBoxMenuItem, JRadioButtonMenuItem
- Méthodes de AbstractButton
  - public void setMnemonic(int mnemonic)
    souligne la première occurrence du caractère dans le nom du bouton
  - public void setDisplayedMnemonicIndex(int index)
    souligne la index^{ième} occurrence du caractère dans le nom du bouton

Codes mnémoniques code du caractère

Utilisation des constantes définies dans KeyEvent

Exemples

- Touche B : KeyEvent.VK_B (VK = Virtual Key)
- Touche € : KeyEvent.VK_EURO_SIGN
- Touche ↓ du clavier standard : KeyEvent.VK_DOWN
- Touche ↓ du pavé numérique : KeyEvent.VK_KP_DOUWN

Codes mnémoniques Exemple

```
public class TestRaccourcisClaviers extends JFrame {
        JLabel affichage;

    public TestRaccourcisClaviers() {
        super("test raccourcis claviers");
        JMenuBar barre = new JMenuBar();
        JMenu menu = new JMenu("Texte");
        menu.setMnemonic(KeyEvent.VK_T);
        JPanel cont = new JPanel();
        cont.setLayout(new BorderLayout());
        setContentPane(cont);
        affichage = new JLabel("rien");
        cont.add(affichage,BorderLayout.CENTER);
        JMenuItem bleu = new JMenuItem("bleu");
        bleu.setMnemonic(KeyEvent.VK_B);
        bleu.addActionListener(new
ChangeLabel("bleu",Color.BLUE));
        menu.add(bleu);
        JMenuItem rouge = new JMenuItem("rouge");
        rouge.setMnemonic(KeyEvent.VK_R);
        rouge.addActionListener(new ChangeLabel("rouge",
Color.RED));
        menu.add(rouge);
        barre.add(menu);
        setJMenuBar(barre);
        pack();
        setVisible(true);
    }
    class ChangeLabel implements ActionListener {
        private String nom;
        private Color coul;
        ChangeLabel(String nom, Color coul) {
            this.nom=nom;
            this.coul=coul;
        }
        public void actionPerformed(ActionEvent ae) {
            affichage.setForeground(coul);
            affichage.setText(nom);
        }
    }

    public static void main(String[] args) {
        JFrame fenetre = new TestRaccourcisClaviers();
        fenetre.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Accélérateurs

- Composants visés
  - Éléments de menus
    - JMenuItem
    - JCheckBoxMenuItem
    - JRadioButtonMenuItem
  - Mais pas les menus
    - Redéfini dans Jmenu pour renvoyer une erreur
- Méthode de JMenuItem

  public void setAccelerator(KeyStroke k)

  KeyStroke : la combinaison de touche servant d'accélérateur

Accélérateurs
Construction d'un objet KeyStroke

- Utiliser les méthodes de classe de la classe KeyStroke parmi lesquelles :
  - static KeyStroke getKeyStroke(int keycode, int modifiers)
  - static KeyStroke getKeyStroke(int keycode, int modifiers, boolean onKeyRelease)
- Arguments :
  - keycode : code du caractère (c.f. Codes mnémoniques)
  - onKeyRelease : lorsqu'on relâche la touche ou bien lorsqu'on presse la touche (par défaut = presse)
  - modifiers : autres touches enfoncées

Accélérateurs et KeyStroke
Paramètre modifiers

Modifiers : "ou" bit-à-bit entre les valeurs suivantes :

- InputEvent.SHIFT_DOWN_MASK
- InputEvent.CTRL_DOWN_MASK
- InputEvent.META_DOWN_MASK
- InputEvent.ALT_DOWN_MASK
- InputEvent.ALT_GRAPH_DOWN_MASK

Accélérateurs Exemple

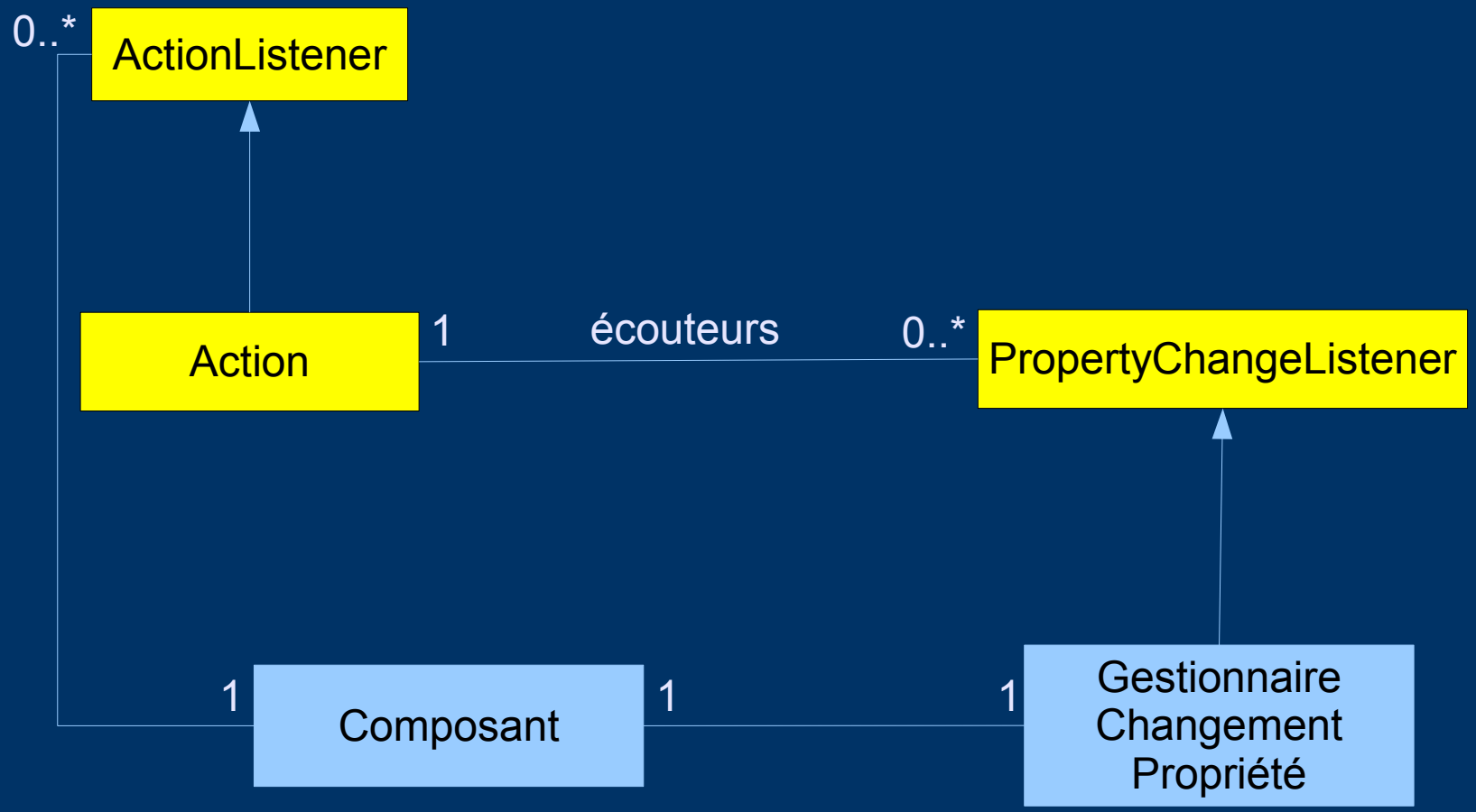
```
public class TestRaccourcisClaviers extends JFrame {
        JLabel affichage;

    public TestRaccourcisClaviers() {
        super("test raccourcis claviers");
        JMenuBar barre = new JMenuBar();
        JMenu menu = new Jmenu("Texte");
        JPanel cont = new JPanel();
        cont.setLayout(new BorderLayout());
        setContentPane(cont);
        affichage = new JLabel("rien");
        cont.add(affichage,BorderLayout.CENTER);
        JMenuItem bleu = new JMenuItem("bleu");
        bleu.setAccelerator( KeyStroke.getKeyStroke(
         KeyEvent.VK_L, 0));
        bleu.addActionListener(new
ChangeLabel("bleu",Color.BLUE));
        menu.add(bleu);
        JMenuItem rouge = new JMenuItem("rouge");
         rouge.setAccelerator(KeyStroke.getKeyStroke(
             KeyEvent.VK_G,
             InputEvent.CTRL_DOWN_MASK|
                 InputEvent.SHIFT_DOWN_MASK));
        rouge.addActionListener(new ChangeLabel("rouge",
Color.RED));
        menu.add(rouge);
        barre.add(menu);
        setJMenuBar(barre);
        pack();
        setVisible(true);
    }
    class ChangeLabel implements ActionListener {
        private String nom;
        private Color coul;
        ChangeLabel(String nom, Color coul) {
            this.nom=nom;
            this.coul=coul;
        }
        public void actionPerformed(ActionEvent ae) {
            affichage.setForeground(coul);
            affichage.setText(nom);
        }
    }

    public static void main(String[] args) {
        JFrame fenetre = new TestRaccourcisClaviers();
        fenetre.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

Les <<actions>>

Actions
Problématique

- Souvent plusieurs moyens de faire la même chose
  - Option de menu
  - Bouton d'une barre d'outils
  - Clic sur un bouton
  - Frappe Clavier
  - ...
- Comment valider/invalider d'un coup toutes ces commandes ?
- Solution : relier toutes ces commandes à un seul objet de type Action dont les commandes "observeront" les changements de propriété

Les actions : architecture



Principes

- L'Action réagit aux événements (actionPerformed) émanant du Composant
- Le Composant réagit aux changements de propriété de l'Action

Interface Action
Classe AbstractAction

- Liste des méthodes
  - void addPropertyChangeListener(PropertyChangeListener l)
  - void removePropertyChangeListener(PropertyChangeListener l)
  - void setEnabled(boolean b)
  - boolean IsEnabled()
  - void putValue(String key, Object Value)
  - Object getValue(String key)
- Classe AbstractAction
  - Implante Action
  - Définit toutes les méthodes de AbstractAction sauf actionPerformed
  - Définit également :
    - Des constructeurs
      - AbstractAction(String name)
      - AbstractAction(String name, Icon icone)
    - De nouvelles méthodes
      - quelques autres méthodes :
      - FirePropertyChange(...)
      - Object [] getKeys()
      - PropertyChangeListener[] getPropertyChangeListeners()

Propriétés prédéfinies de Action

nom de la propriété de AbstractAction	propriété cible du composant
Enabled (via isEnabled)	enabled
SHORT_DESCRIPTION	toolTipText
ACTION_COMMAND_KEY	actionCommand
MNEMONIC_KEY	mnemonic
NAME	text
LARGE_ICON_KEY	icon (sauf JmenuItem si SMALL_ICON non nul)
SMALL_ICON	icon (JmenuItem)
ACCELERATOR_KEY	accelerator
SELECTED_KEY	selected

Remarque

Si text et LARGE_ICON_KEY définis, un bouton affichera les deux. Pour qu'il n'affiche que l'icône, faire un setHideActionText(true) sur le bouton

Définir et utiliser une action

- Définir l'action
  - Créer une classe héritant de AbstractAction
  - Appeler le constructeur de la superclasse requis
  - Définir la méthode actionPerformed
- Utiliser l'action
  - Créer les boutons et items de menu à partir de l'action
  - Désactiver éventuellement l'affichage du texte pour les boutons

Actions et frappes clavier

Actions et Frappes Clavier

- Problème
  - Comment définir des sortes d'accélérateur sans item de menu correspondant

  Ou
  - Comment déclencher automatiquement certaines actions sur certaines entrées clavier ?
- Solutions
  - InputMap et ActionMap

Actions et Frappes Clavier
Principe général (1)

- Une *InputMap* associe un objet clé (*Object*) à une frappe clavier (*KeyStroke*)
- Une *ActionMap* associe une action (*Action*) à une clé (*Object*)

=> Frappe clavier --> Object --> Action

Remarque :

La clé peut être n'importe quel objet. En général, on choisit le nom de l'action

Actions et Frappes Clavier
Principe général (2)

- Tout composant Swing a
  - 3 InputMap
    - WHEN_FOCUSED

      Valable uniquement quand le composant a le focus
    - WHEN_ANCESTOR_OR_FOCUSED_COMPONENT

      Quand le composant ou un de ses descendants à le focus
    - WHEN_IN_FOCUSED_WINDOW

      Quand la fenêtre du composant a le focus
  - 1 ActionMap

Actions et Frappes Clavier Exemple

- Mettre un équivalent clavier (ctrl-b) à un bouton (*JButton*) créé à partir d'une action *actionBouton :* bouton = new JButton(actionBouton)
- Méthode 1 : Passer par la table WHEN_ANCESTOR_OR_FOCUSED_COMPONENT de la fenêtre

```
getRootPane().getInputMap(
        Jcomponent.WHEN_ANCESTOR_OF_FOCUSED_COMPONENT).
        put(KeyStroke.getKeyStroke(KeyEvent.VK_B,
        KeyEvent.CTRL_DOWN_MASK),"bouton");
getRootPane().getActionMap().put("bouton", actionBouton);
```

- Méthode 2 : Passer par la table WHEN_IN_FOCUSED_WINDOW du Jbutton

```
bouton.getInputMap(
      Jcomponent.WHEN_IN_FOCUSED_WINDOW).put(KeyStroke.getKeyStroke(
      KeyEvent.VK_B, KeyEvent.CTRL_DOWN_MASK),"bouton");
bouton.getActionMap().put("bouton", actionBouton);
```

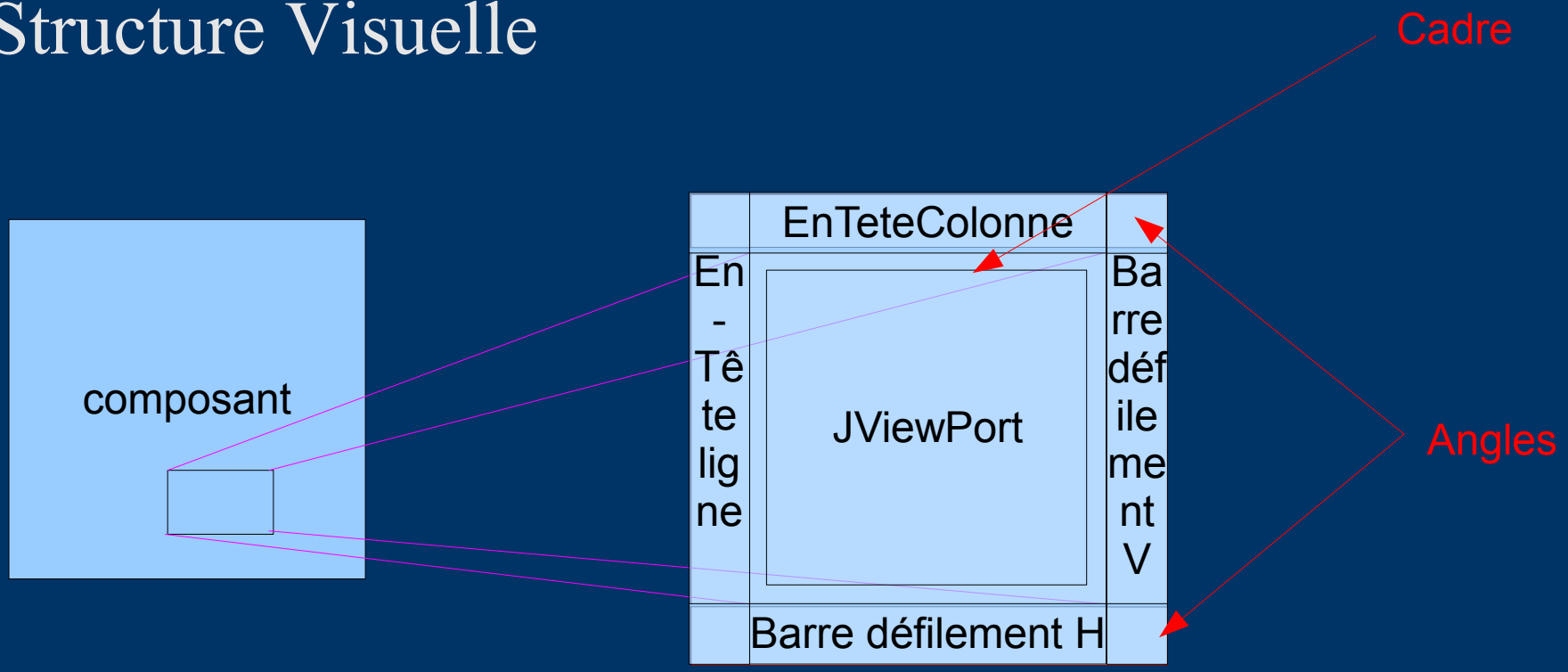
Conteneurs spécifiques

JScrollPane

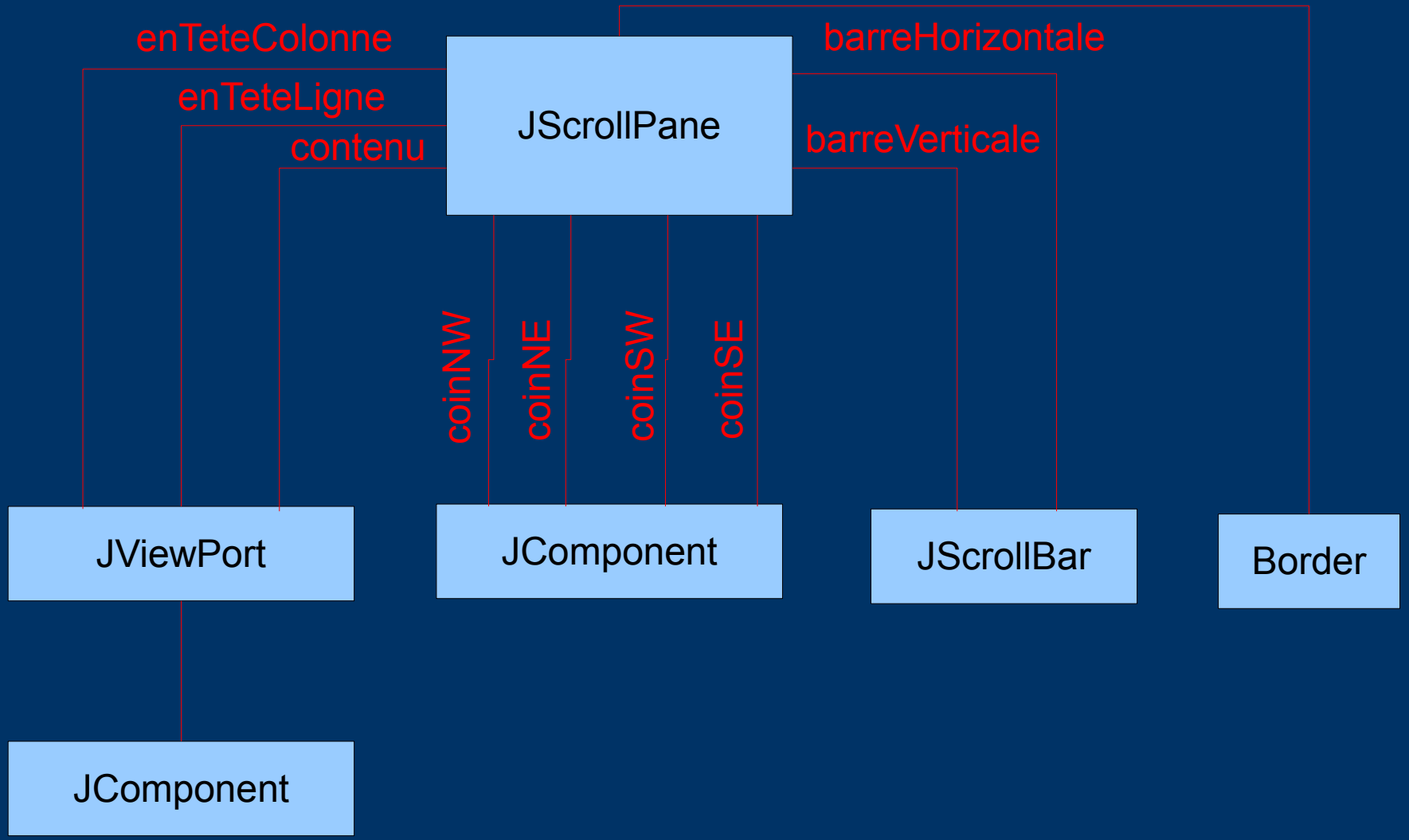
- Rôle

  Ajouter des ascenseurs à n'importe quel composant

- Structure Visuelle



JScrollPane
Structure interne



JScrollPane
Affichage du composant principal

- Taille du ViewPort
  - Déterminée par le JscrollPaneLayout
- Affichage des barres de défilement
  - Méthodes
    - Void setHorizontalScrollBarPolicy(int politique)
    - Void setVerticalScrollBarPolicy(int politique
  - Valeurs (*XXX* ∈ {HORIZONTAL, VERTICAL})
    - *XXX*_SCROLLBAR_AS_NEEDED
    - *XXX*_SCROLLBAR_NEVER
    - *XXX*_SCROLLBAR_ALWAYS

JScrollPane
Affichage d'autres composants

- En-têtes
  - void setColumnHeaderView(Component c)
  - void setRowHeaderView(Component c)
- Coins
  - void setCorner(String position, Component c)
  - Avec *position* pouvant valoir :
    - LOWER_LEFT_CORNER, LOWER_RIGHT_CORNER
    - UPPER_LEFT_CORNER, UPPER_RIGHT_CORNER
    - LOWER_LEADING_CORNER, LOWER_TRAILING_CORNER
    - UPPER_LEADING_CORNER, UPPER_TRAILING_CORNER

JScrollPane
Gestion du déplacement

- BlockIncrement
  - Déplacement lorsque l'on clique avant/après un ascenseur
- UnitIncrement
  - Déplacement lorsque l'on clique sur les flèches de l'ascenseur
- Molette de la souris
  - Géré par le système
  - Activé par setWhellScrollingEnabled(true)

JScrollPane
Détermination des incréments

- Si le composant n'implante pas Scrollable
  - Incréments déterminés par JscrollPane
- Si le composant implant Scrollable
  - Par le composant lui-même

JScrollPane
L'interface Scrollable

- Taille préférée du ViewPort

  Dimension getPreferredScrollableViewportSize()
- Détermination des incréments
  - Méthodes
    - Int getScrollableBlockIncrement(Rectangle recVisible, int orientation, int direction)
    - Int getScrollableUnitIncrement(Rectangle recVisible, int orientation, int direction)
  - Paramètres
    - Orientation : VERTICAL/HORIZONTAL
    - Direction : <0 → gauche/haut, >0 → droite/bas
- Forcer la taille du composant à être la taille du viewPort
  - Boolean getScrollableTrackViewportHeight()
  - Boolean getScrollableTrackViewportWidth()

JSplitPane

- But
  - Partager une fenêtre en 2 parties, verticalement ou horizontalement, avec barre de séparation déplaçable
- Aperçu

JSplitPane Constructeurs

- Liste
  - JSplitPane()
  - JSplitPane(int newOrientation)
  - JSplitPane(int newOrientation, boolean newContinuousLayout)
  - JSplitPane(int newOrientation, boolean newContinuousLayout, Component newLeftComponent, Component newRightComponent)
  - JSplitPane(int newOrientation, Component newLeftComponent, Component newRightComponent)
- Paramètres
  - NewOrientation : HORIZONTAL_SPLIT / VERTICAL_SPLIT
  - NewContinuousLayout : mise à jour en temps réel ou seulement après le positionnement
  - NewLeftComponent : composant de gauche/du haut
  - NewRightComponent : composant de droite/du bas

JSplitPane
Quelques méthodes

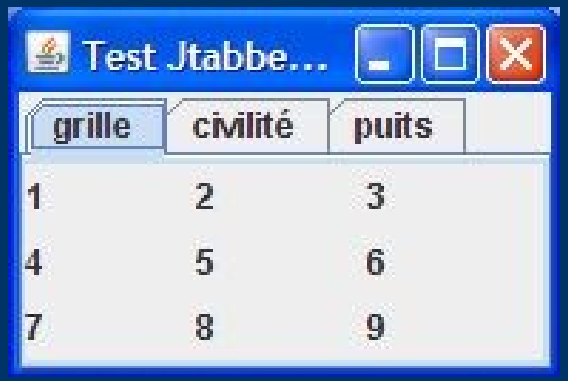
- Void setDividerLocation(double pos)
- Void setRightComponent(Component c) / void setBottomComponent
- Void setLeftComponent(Component c) / void setTopComponent
- Void setOrientation(int orientation)
- Void setOneTouchExpandable(boolean b)
- Void setResizeWeight(double val)
- Void setDividerSize(int taille)
- Void setContinuousLayout(boolean b)

JTabbedPane

- Rôle

  Gérer un affichage de composants sous forme d'onglets

- Aperçu



JTabbedPane
Affichage des onglets

- Position
  - Spécification
    - Constructeurs
    - setTabPlacement(int place)
  - Valeurs possibles
    - TOP, BOTTOM
    - LEFT, RIGHT
- Politique
  - Spécification
    - Constructeurs
    - setTabLayoutPolicy(int politique)
  - Valeurs possibles
    - WRAP_TAB_LAYOUT
    - SCROLL_TAB_LAYOUT
- Nombre de lignes/colonnes utilisées : int getTabRunCount()

JTabbedPane
Ajout/Suppression d'une page

- Ajout
  - Component add(String titre, Component c)
  - Component add(Component c) (titre = c.getName())
  - Component add(Component c, int index) (=insertTab)
  - void add(Component c, Object contrainte)
    Si contrainte : String ou Icon, utilisé pour le titre
  - void add(Component c, Object contrainte, int index)
  - void addTab(String nom, Component c)
  - void addTab(String nom, Icon icone, Component c)
  - void addTab(String nom, Icon icone, Component c, String tip)
  - void insertTab(String titre, Icon icone, Component c, String tip, int index)
- Suppression
  - Void remove(Component c)
  - Void remove(int index)
  - Void removeAll()
  - Void removeTabAt(int index)

JTabbedPane
Gestion de la sélection

- Modèle sous-jacent
  - SingleSelectionModel (getModel/setModel)
- Méthodes
  - Component GetSelectedComponent()
  - Int GetSelectedIndex()
  - Void setSelectedComponent(Component c)
  - Void setSelectedIndex(int i)
- Détection de changements de sélection
  - addChangeListener/removeChangeListener()

JTabbedPane
Paramètres des pages

- Icône/Titre
  - setIconAt(int index, Icon icône)/setDisabledIconAt...
  - setTitleAt(int index, String titre)
- Bulle d'aide
  - setToolTipTextAt(int index, String tip)
- Couleurs
  - setForegroundAt(int index, Color c)
  - setBackgroundAt(int index, Color c)
- Validation
  - setEnabledAt(int index, boolean b)
- Code mnémonique
  - setMnemonicAt(int index, int m)

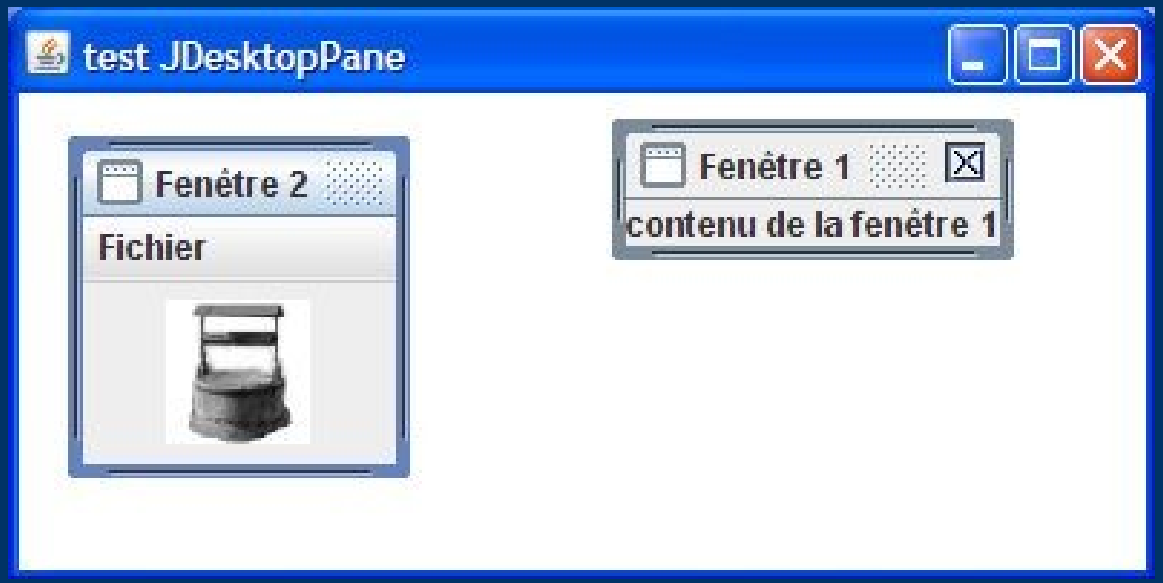
JTabbedPane
Conversion composant/index

- Méthodes
  - Component getComponentAt(int index)
  - Int indexOfComponent(Component c)
  - Int indexOfTab(Icon icône)
  - Int indexOfTab(String titre)
  - Int indexOfTabComponent(Component c)
- Attention !
  - Component getTabComponentAt(int index)

    Ne renvoie pas le composant mis dans la page, mais le composant affichant le titre !
  - Void setTabComponentAt(int index)
    - Spécifie le composant qui affichera le titre de la page

JDesktopPane et JInternalFrame

- But
  - Gérer des sous-fenêtres dans une fenêtre (bureau virtuel)
- Aperçu



JDesktopPane et JInternalFrame Exemple

```
public class TestJDesktopPane extends JFrame {
    public TestJDesktopPane() {
        super("test JDesktopPane");JDesktopPane contenu = new JDesktopPane(); setContentPane(contenu);
        JInternalFrame f1 = new JInternalFrame("Fenêtre 1");
        JLabel p1 = new JLabel("contenu de la fenêtre 1");
        f1.add(p1); f1.pack(); f1.setVisible(true);
        f1.setResizable(true); f1.setClosable(true);
        JInternalFrame f2 = new JInternalFrame("Fenêtre 2");
        JMenuBar barre = new JMenuBar();
        JMenu fichier = new JMenu("Fichier");
        fichier.add("Ouvrir");fichier.add("Sauver");fichier.add("Fermer"); barre.add(fichier);
        f2.setJMenuBar(barre);
        f2.add(new JLabel(new ImageIcon("puits.gif")));
        f2.pack(); f2.setVisible(true); f2.setResizable(true);
        contenu.add(f1); contenu.add(f2);
        setSize(400,200); setVisible(true);
    }
    public static void main(String[] args) {
        JFrame fenetre = new TestJDesktopPane();
        fenetre.setDefaultCloseOperation(EXIT_ON_CLOSE);
    }
}
```

JToolBar
Principe

- But :
  - Implanter des palettes d'outils, en général via des boutons
- Principe
  - Les composants sont disposés en utilisant un BoxLayout
  - La palette doit être disposée dans un composant dont la mise en page est géré par un BorderLayout
    - Le contenu du composant est dans la zone CENTER
    - La palette peut être déplacé entre les 4 zones périphériques, voire transformée en fenêtre

JToolBar Exemple


```
JLabel photo1 = new JLabel(new ImageIcon(url));
add(photo1,BorderLayout.CENTER);
JToolBar palette = new JToolBar("Palette");
JButton b1 =new JButton("b1");
JButton b2 = new JButton("b2");
palette.add(b1);
palette.add(b2);
add(palette,BorderLayout.NORTH);
```

JToolBar
Paramétrage

- setFloatable(false)

  la palette ne peut pas être déplacée
- setRollOver(false)

  le bouton sous la souris n'est pas mis en évidence
- AddSeparator()

  Ajoute un séparateur
- setBorderPainted(true)

  Si un cadre a été défini via setBorder(Border b), le dessine ;
  semble incompatible avec Floatable

Manipulation des icônes

Principe général

- Icon = interface
  - int getIconHeight()
  - int getIconWidth()
  - void paintIcon(Component c, Graphics g, int x, int y)
- Utilisation
  - Soit chargement à partir d'une image via la classe ImageIcon
  - Soit dessin manuel en écrivant sa propre classe

Classe ImageIcon
Constructeurs

- ImageIcon()

- ImageIcon(byte[] imageData)
- ImageIcon(byte[] imageData, String description)

- ImageIcon(Image image)
- ImageIcon(Image image, String description)

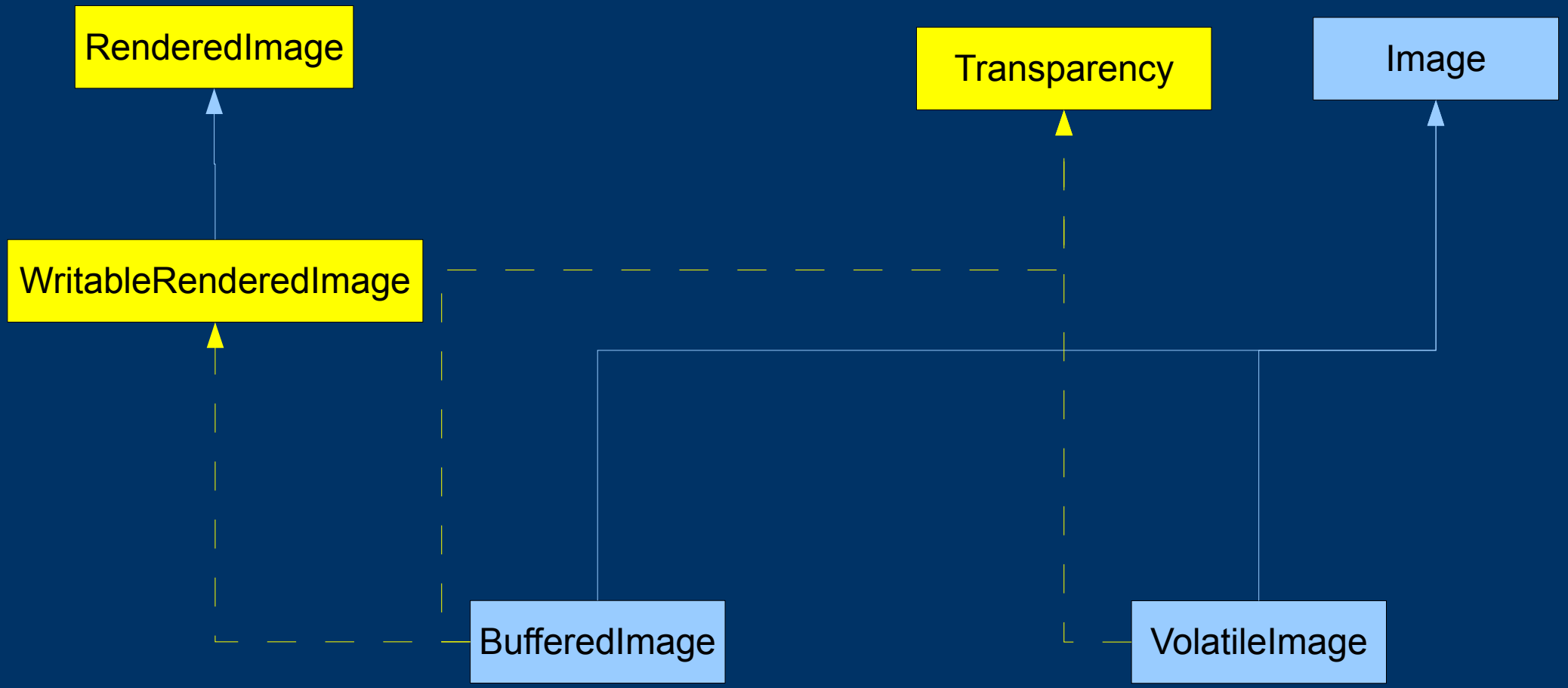
- ImageIcon(String filename)
- ImageIcon(String filename, String description)

- ImageIcon(URL location)
- ImageIcon(URL location, String description)

Classe ImageIcon
Méthodes

- String getDescription()
- int getIconHeight()
- int getIconWidth()
- Image getImage()
- int getImageLoadStatus()
- ImageObserver getImageObserver()
- void paintIcon(Component c, Graphics g, int x, int y)
- void setDescription(String description)
- void setImage(Image image).
- void setImageObserver(ImageObserver observer)
- String toString()

Gestion des Images
Introduction



Application
Mise à l'échelle d'une icône

```
ImageIcon feuille = new ImageIcon("feuille.gif");
```

On crée une première icône à partir d'un fichier

```
Image feuilleImg = feuille.getImage();
```

On récupère l'image associée à l'icône

```
Image feuillePetiteImg = feuilleImg.getScaledInstance(20, -1, Image.SCALE_SMOOTH);
```

On met l'image à la taille désirée :

20 : largeur

-1 : hauteur ; négatif pour préserver le rapport

Image.SCALE_SMOOTH : algo utilisé ; ici, on privilégie la qualité à la vitesse

```
ImageIcon petiteFeuille = new ImageIcon(feuillePetiteImg);
```

On recrée une icône à partir de la nouvelle image

Dessiner sa propre icône
Exemple (extrait de java.sun.com)

```
public class MissingIcon implements Icon{
    private int width = 32;
    private int height = 32;
    private BasicStroke stroke = new BasicStroke(4);

    public void paintIcon(Component c, Graphics g, int x, int y) {
        Graphics2D g2d = (Graphics2D) g.create();
        g2d.setColor(Color.WHITE); g2d.fillRect(x +1 ,y + 1,width -2 ,height -2);
        g2d.setColor(Color.BLACK); g2d.drawRect(x +1 ,y + 1,width -2 ,height -2);
        g2d.setColor(Color.RED); g2d.setStroke(stroke);
        g2d.drawLine(x +10, y + 10, x + width -10, y + height -10);
        g2d.drawLine(x +10, y + height -10, x + width -10, y + 10);
        g2d.dispose();
    }

    public int getIconWidth() {return width;}
    public int getIconHeight() {return height;}
}
```

Polices de caractères

Polices de caractères
Préambule

- Ne pas confondre caractère et glyphe
  - Souvent correspondance 1-1, mais pas tout le temps :
    - caractère "à" -> 2glyphes : "a" + "`"
    - Caractères "fi" -> 1 glyphe "ﬁ"
- Polices physiques vs polices logiques
  - Polices physiques
    - Celles réellement présentes sur le système (Helvetica, Times, etc.)
  - Polices logiques
    - Définies par Java et présentes dans toute machine virtuelle. Instanciées par une des polices physiques disponibles
    - Liste : Serif, SansSerif, Monospaced, Dialog DialogInput

Définition d'une police

- Récupération des polices disponibles

```
GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();
Font[] polices = ge.getAllFonts();
String[] familles= ge.getAvailableFontFamilyNames();
```

- Constructeurs
  - Font(String name, int style, int size)
    - Name : nom de la police ou de la famille de police
- A partir d'une police existante
  - deriveFont(float size)
  - deriveFont(int style)
    - PLAIN
    - BOLD, ITALIC, BOLD|ITALIC
  - deriveFont(AffineTransform t)
    - Applique une transformation affine à la police

Gestion des couleurs

Classe Color
Principes

- Classe permettant de décrire une couleur dans un *espace de couleur* (*ColorSpace*).
- Espace de couleur – Définition :
  - Un espace de couleur est un modèle pour représenter numériquement les couleurs avec au moins 3 coordonnées.
- *ColorSpace* par défaut = sRGB (RVB standard)
- Composantes d'un couleur
  - Rouge, Vert, Bleu
  - Alpha (transparence)

Classe Color
Constructeurs

- Color(ColorSpace cspace, float[] composantes, float alpha)
  - Pour un autre ColorSpace que sRGB
- Color(float r, float g, float b)
  - Pourcentages de la luminosité de chaque couleur
- Color(int r, int g, int b)
  - Niveau de chaque couleur de 0 à 255
- Color(int rgb)
  - Niveau de rouge dans les bits 16-23, vert dans les bits 8-15, bleu dans les bits 0-7
- Color(float r, float g, float b, float alpha)
- Color(int r, int g, int b, int alpha)
- Color(int rgba, boolean hasAlpha)

  Idem que les précédents mais avec paramètre alpha

Classe Color
Constantes

- BLACK
- BLUE
- CYAN
- DARK_GREY
- GRAY
- GREEN
- LIGHT_GREY
- MAGENTA
- ORANGE
- PINK
- RED
- WHITE
- YELLOW

Classe Color
Quelques méthodes

- Color brighter(), Color darker()

  Fournit une nouvelle couleur légèrement plus claire/plus foncée que la couleur courante. Peut être appliqué plusieurs fois

- int getAlpha(), int getBlue(), int getGreen(), int getRed(), int getRGB()

- float [] getColorComponents, Float[] getComponents()

  Renvoient les pourcentages des niveaux de chaque couleur (et d'alpha pour la deuxième méthode)

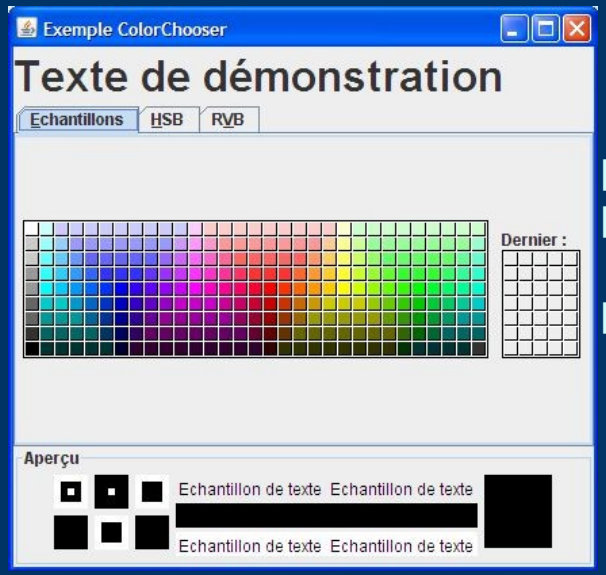
- static Color getHSB(float h, float s, float b)

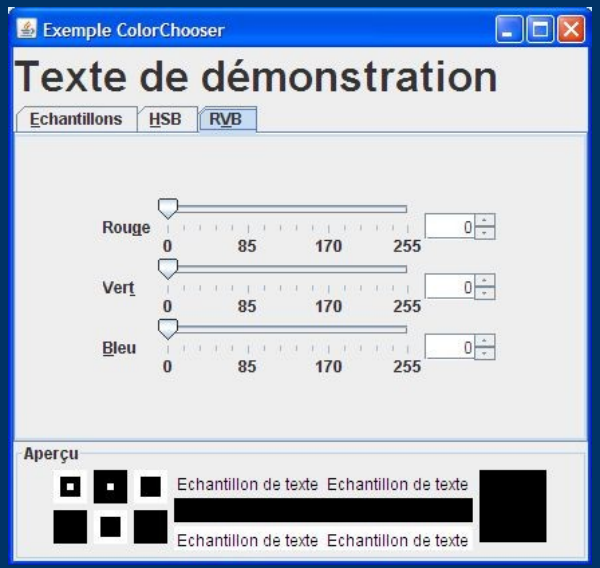
  Obtenir une couleur à partir de ses teinte, saturation, luminosité

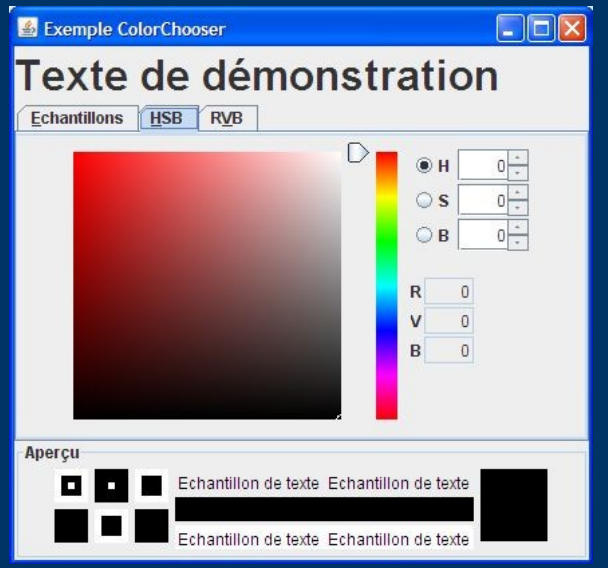
Classe JColorChooser
Introduction

- Buts :
  - Fournir une interface à l'utilisateur pour sélectionner une couleur
  - Proposer "de base" plusieurs moyens de choisir une couleur
  - Permettre d'intégrer ses propres "sélecteurs" de couleurs
- Utilisation de base
  - Soit comme composant
  - Soit via une boîte de dialogue
    - Color showDialog(Component parent, String titre, Color initiale)
    - JDialog createDialog(...), puis show sur le JDialog renvoyé
  - Réagir au changement de sélection
    - getSelectionModel().addChangeListener(...)

Classe JColorChooser Exemple








```
private JLabel demo;
private JColorChooser selecteur;

public TestColorChooser() {
    super("Exemple ColorChooser");
    demo = new JLabel("Texte de démonstration");
    demo.setFont(demo.getFont().deriveFont((float) 32));
    add(demo, BorderLayout.NORTH);

    selecteur = new JColorChooser(Color.BLACK);
    add(selecteur,BorderLayout.CENTER);

    selecteur.getSelectionModel().addChangeListener(new ChangeListener() {
        public void stateChanged(ChangeEvent e) {
            demo.setForeground(selecteur.getColor());
        }
    });
    pack();
    setDefaultCloseOperation(EXIT_ON_CLOSE);
    setVisible(true);
}
```

Classe JColorChooser
Utilisation avancée

- Rajouter/Supprimer des onglets
  - void addChooserPanel(AbstractColorChooserPanel panel)
  - void removeChooserPanel(AbstractColorChooserPanel panel)
  - void setChooserPanels(AbstractColorChooserPanel[] panels)
- Modifier la zone de prévisualisation
  - Void setPreviewPanel(JComponent preview)

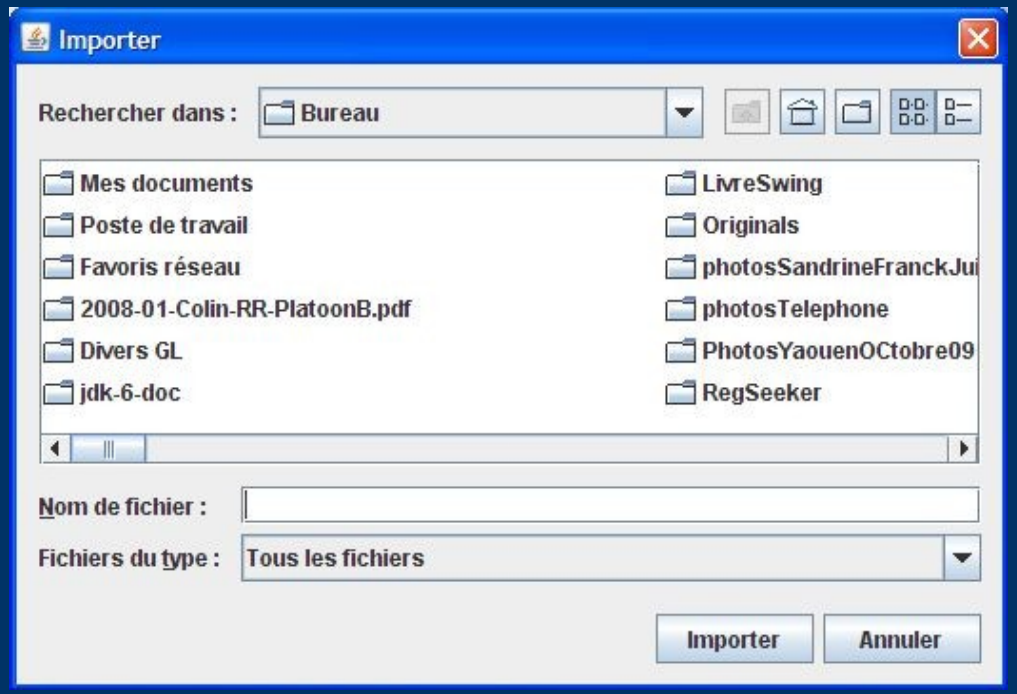
Boîtes de dialogue prédéfinies

Sélection de fichiers
Introduction

- 1 classe : JFileChooser avec différents constructeurs
  - JFileChooser()
  - JFileChooser(File répertoire)
  - JFileChooser(String répertoire)
- Plusieurs dialogues :
  - Ouverture : int showOpenDialog(Component parent)
  - Sauvegarde : int showSaveDialog(Component parent)
  - Autre : int showDialog(Component parent, String txt)
  - N.B. : retour : APPROVE_OPTION, CANCEL_OPTION, ERROR_OPTION

Sélection de fichiers Exemple

```
bouton.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JFileChooser jfc = new JFileChooser(

FileSystemView.getFileSystemView().getHomeDir
ectory());
        int retour =
jfc.showDialog(TestSelectionFichier.this,
"Importer");
        switch(retour) {
            case JFileChooser.APPROVE_OPTION:
                action.setText("OK");

fichier.setText(jfc.getSelectedFile().getName());
                break;
            case JFileChooser.CANCEL_OPTION:
                action.setText("Cancel");break;
            default:
                action.setText("erreur");
}}});
```



Sélection de fichiers
Paramètres

- Void rescanCurrentDirectory()
- setAccessory(JComponent accessoire)
- setFileHidingEnabled(boolean b)
- setMultiSelectionEnabled(boolean b)
- setFileSelectionMode(int mode)
  - FILES_ONLY, DIRECTORIES_ONLY, FILES_AND_DIRECTORIES

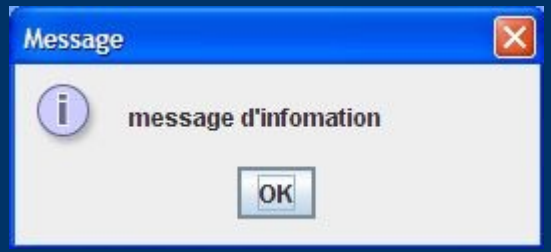
Sélection de fichiers
Filtres sur les fichiers

- Par défaut, un filtre : "tous les fichiers"
  - Désactivation : setAcceptAllFileFilterUsed(false)
- Ajout d'autres filtres :
  - addChoosableFileFilter(FileFilter filtre)
- Classe abstraite FileFilter : 2 méthodes
  - boolean accept(File f)
  - String getDescription()
- Classe concrète FileNameExtensionFilter
  - Constructeur : FileNameExentionFilter(String description, String... extensions)
  - Exemple d'utilisation
    - FileFilter ff = new FileNameExtensionFilter("images", "jpeg", "jpg", "gif");
    - jfc.addChoosableFileFilter(ff);

JOptionPane
Création de boîtes de dialogue simples

- Rôle
  - (Essentiellement) ensemble de méthodes de classes pour créer des boîtes de dialogues basiques
- Types de boîtes de dialogue
  - Message d'information (MessageDialog)
  - Demande de confirmation (ConfirmDialog)
  - Demande d'information avec choix fermé ou libre (InputDialog)
- Existe en version Dialogue standard et en version JInternalFrame

JOptionPane Exemples



JOptionPane MessageDialog : message d'info

- Static void showMessageDialog(Component parent, Object message)
  - Affiche le message
- Static void showMessageDialog(Component parent, Object message, String title, int messageType)
  - Title : titre de la fenêtre
  - MessageType : ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE
  - Le type de message détermine l'icone
- Static void showMessageDialog(Component parent, Object message, String title, int messageType, Icon icone)

JOptionPane MessageDialog : code des exemples

```
bMessage1 = new JButton("Message 1");
boutons.add(bMessage1);
bMessage1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(TestOptionPane.this, "message d'infomation");
    }
});


bMessage2 = new JButton("Message 2");
boutons.add(bMessage2);
bMessage2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        JOptionPane.showMessageDialog(TestOptionPane.this, "message d'alerte",
                "Ma boîte", JOptionPane.WARNING_MESSAGE);
    }
});
```

JOptionPane
ConfirmDialog : demande de confirmation

- Static int showConfirmDialog(Component parent, Object message)
  - Affiche le message avec trois boutons Oui/Non/Annuler
- Static int showConfirmDialog(Component parent, Object message, String title, int optionType)
  - Title : titre de la fenêtre
  - OptionType : YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION
- Static int showConfirmDialog(Component parent, Object message, String title, int optionType, int messageType)
  - MessageType : ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE
  - Le type de message détermine l'icone
- Static int showConfirmDialog(Component parent, Object message, String title, int optionType, int messageType, Icon icône)
- Retour : YES_OPTION, NO_OPTION, CANCEL_OPTION, OK_OPTION, CANCEL_OPTION

JOptionPane ConfirmDialog : code des exemples

```
bConfirm1 = new JButton("Confirmation 1");boutons.add(bConfirm1);
bConfirm1.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int choix = JOptionPane.showConfirmDialog(TestOptionPane.this, "message à
confirmer");
        switch(choix) {
            case JOptionPane.YES_OPTION:affichage.setText("Oui");break;
            case JOptionPane.NO_OPTION:affichage.setText("non");break;
            case JOptionPane.CANCEL_OPTION:affichage.setText("annuler");break;
            default:affichage.setText("autre");
}}});

bConfirm2 = new JButton("Confirmation 2"); boutons.add(bConfirm2);
bConfirm2.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        int choix = JOptionPane.showConfirmDialog(TestOptionPane.this,
                "message à confirmer", "Demande de confirmation",
                JOptionPane.OK_CANCEL_OPTION, JOptionPane.PLAIN_MESSAGE);
        switch(choix) {
            case JOptionPane.OK_OPTION:affichage.setText("OK");break;
            case JOptionPane.CANCEL_OPTION:affichage.setText("annuler");break;
            default:affichage.setText("autre");
}}});
```

JOptionPane
InputDialog : saisie d'un texte

- Static String showInputDialog(Component parent, Object message)
- Static String showInputDialog(Component parent, Object message, Object valeurInitiale)
- Static String showInputDialog(Component parent, Object message, String titre, int messageType)

- static Object showInputDialog(Component parent, Object message, String titre, int messageType, Icon icone, Object[] listePossibilités, Object valeurInitiale)
  - Choix limité via, par exemple, une JComboBox

JOptionPane
InputDialog : code des exemples

```
        bInput1 = new JButton("Saisie 1");boutons.add(bInput1);
        bInput1.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String saisie = JOptionPane.showInputDialog(TestOptionPane.this, "Entrez votre
texte");
                if (saisie != null) {affichage.setText(saisie);}
                else {affichage.setText("Saisie annulée");}
        }});

        bInput2 = new JButton("Saisie 2");boutons.add(bInput2);
        bInput2.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                String[] choix = {"ane", "chevre", "lapin", "mouton", "oie"};
                Object saisie = JOptionPane.showInputDialog(TestOptionPane.this,
                        "Sélectionnez :", "Choix d'une option",
                        JoptionPane.QUESTION_MESSAGE, null, choix, "mouton");
                if (saisie != null) {affichage.setText(saisie.toString());}
                else {affichage.setText("Saisie annulée");}
        }});
```

JoptionPane & JDesktopPane

- ShowInternalConfirmDialog
- ShowInternalInputDialog
- ShowInternalMessageDialog

A faire plus tard

- Drag-and-Drop
- Dessin 2D
- Impression
- Texte avec mise en forme