JTable : exemple

JTable : ajout dans un conteneur

- Pour un JScrollPane

```
JScrollPane scrollPane = new JScrollPane(table);
table.setFillsViewportHeight(true);
```

- Pour un autre conteneur

```
container.setLayout(new BorderLayout());
container.add(table.getTableHeader(), BorderLayout.PAGE_START);
container.add(table, BorderLayout.CENTER);
```

- Ainsi, les titres de colonne resteront toujours visible

JTable : cas de base

- Contraintes d'utilisation du cas de base
  - Tout est éditable sous la forme de chaîne de caractère
- Constructeurs :
  - JTable(Object[][] données, Object[] nomsColonnes)
  - JTable(Vector données, Vector nomsSolonnes)

    Données est un vecteur de vecteurs d'objets

JTable : exemple du cas de base

```
import java.util.Vector;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTable;

public class Base extends JFrame {
    public Base() {
        super("table de base");
        // Intitulés des colonnes
        Vector<String> colonnes = new Vector<String>();
        colonnes.add("Nom"); colonnes.add("Prénom"); colonnes.add("Age");
        // Contenu de la table
        Vector<Vector<String>> donnees = new Vector<Vector<String>>();
        Vector<String> donnee1 = new Vector<String>();
        donnee1.add("fournier"); donnee1.add("domique"); donnee1.add("35"); donnees.add(donnee1);
        Vector<String> donnee2 = new Vector<String>();
        donnee2.add("amanton");donnee2.add("laurent"); donnee2.add("20"); donnees.add(donnee2);
        // Création de la table
        JTable table = new JTable(donnees, colonnes);
        JScrollPane panneau; panneau = new JScrollPane(table);
        table.setFillsViewportHeight(true);
        add(panneau);
        setDefaultCloseOperation(EXIT_ON_CLOSE); pack(); setVisible(true);
    }
    public static void main(String [] args) {
        Base fenetre = new Base();
    }}
```

JTable : modèles sous-jacents

- Modèle des colonnes (TableColumnModel)
  - Interface TableColumnModel
  - Implantation de base : DefaultTableColumnModel
- Modèle des données (TableModel)
  - Interface TableModel
  - Implantation de base : DefaultTableModel (sous-classe de AbstractTableModel)
- Modèle de sélection (ListSelectionModel)

JTable : interface TableColumnModel

- Modification des colonnes
  - void addColumn(TableColumn col)
  - void removeColumn(TableColumn col)
  - void moveColumn(int columnIndex, int newIndex)
  - void setColumnMargin(int newMargin)
- Infos sur les colonnes
  - int getColumnMargin()
  - Enumeration<TableColumn> getColumns()
  - int getColumnCount()
  - int getColumnIndex(Object columnIdentifier)
  - int getColumnIndexAtX(int xPosition)
  - int getTotalColumnWidth()
  - TableColumn getColumn(int columnIndex)
- Sélection
  - boolean getColumnSelectionAllowed()
  - int getSelectedColumnCount()
  - int[] getSelectedColumns()
  - ListSelectionModel getSelectionModel()
  - void setColumnSelectionAllowed(boolean flag)
  - void setSelectionModel(ListSelectionModel newModel)
- Ecouteurs
  - void addColumnModelListener(TableColumnModelListener x)
  - void removeColumnModelListener(TableColumnModelListener x)

JTable : la classe TableColumn (1)

- Gère ce qui est propre à une colonne :
  - Sa largeur
  - Son intitulé
  - Les afficheurs (*renderer*) et éditeurs (*editor*) de ses données
- Constructeurs
  - TableColumn()
  - TableColumn(int modelIndex)

    m*odelIndex* : indice du champ des données associé à la colonne (quelle que soit la position de la colonne affichée)
  - TableColumn(int modelIndex, int width)
  - TableColumn(int modelIndex, int width, TableCellRenderer cellRenderer, TableCellEditor cellEditor)

JTable : la classe TableColumn (2)

Méthodes

- Void addPropertyChangeListener(PropertyChangeListener listener)
- TableCellEditor getCellEditor()
- TableCellRenderer getCellRenderer()
- TableCellRenderer getHeaderRenderer()
- Object getHeaderValue() / void setHeaderValue(Object headerValue)
- Object getIdentifier() / void setIdentifier(Object identifier)
- int getMaxWidth() / void setMaxWidth(int maxWidth)
- int getMinWidth() / void setMinWidth(int minWidth)
- int getModelIndex() / void setModelIndex(int modelIndex)
- int getPreferredWidth() / void setPreferredWidth(int preferredWidth)
- PropertyChangeListener[] getPropertyChangeListeners()
- boolean getResizable()
- int getWidth() / void setResizable(boolean isResizable)
- void removePropertyChangeListener(PropertyChangeListener listener)
- void setCellEditor(TableCellEditor cellEditor)
- void setCellRenderer(TableCellRenderer cellRenderer)
- void setHeaderRenderer(TableCellRenderer headerRenderer)
- void sizeWidthToFit()

JTable : la classe TableColumnModelListener

- 5 méthodes
  - void columnAdded(TableColumnModelEvent e)
  - void columnMarginChanged(ChangeEvent e)
  - void columnMoved(TableColumnModelEvent e)
  - void columnRemoved(TableColumnModelEvent e)
  - void columnSelectionChanged(ListSelectionEvent e)

- La classe TableColumnModelEvent
  - void columnAdded(TableColumnModelEvent e)
  - void columnMarginChanged(ChangeEvent e)
  - void columnMoved(TableColumnModelEvent e)
  - void columnRemoved(TableColumnModelEvent e)
  - void columnSelectionChanged(ListSelectionEvent e)

JTable : modèle des données
L'interface TableModel

- void addTableModelListener(TableModelListener l)
- Class<?> getColumnClass(int columnIndex)
- int getColumnCount()
- String getColumnName(int columnIndex)
- int getRowCount()
- Object getValueAt(int rowIndex, int columnIndex)
- boolean isCellEditable(int rowIndex, int columnIndex)
- void removeTableModelListener(TableModelListener l)
- void setValueAt(Object aValue, int rowIndex, int columnIndex)

JTable : la classe DefaultTableModel (1)

- Constructeurs
  - DefaultTableModel() / DefaultTableModel(int rowCount, int columnCount)
  - DefaultTableModel(Object[][] data, Object[] columnNames)
  - DefaultTableModel(Object[] columnNames, int rowCount)
  - DefaultTableModel(Vector columnNames, int rowCount)
  - DefaultTableModel(Vector data, Vector columnNames)
- Méthodes de la classe AbstractTableModel
  - void addTableModelListener(TableModelListener l)
  - int findColumn(String columnName)
  - void fireTableCellUpdated(int row, int column)
  - void fireTableChanged(TableModelEvent e)
  - void fireTableDataChanged()
  - void fireTableRowsDeleted(int firstRow, int lastRow)
  - void fireTableRowsInserted(int firstRow, int lastRow)
  - void fireTableRowsUpdated(int firstRow, int lastRow)
  - void fireTableStructureChanged()
  - Class<?> getColumnClass(int columnIndex)
  - String getColumnName(int column)
  - <T extends EventListener> T[] getListeners(Class<T> listenerType)
  - TableModelListener[] getTableModelListeners()
  - void removeTableModelListener(TableModelListener l)
  - void setValueAt(Object aValue, int rowIndex, int columnIndex)

JTable : la classe DefaultTableModel (2)

- Méthodes
  - void addColumn(Object columnName) / void addColumn(Object columnName, Object[] columnData) / void addColumn(Object columnName, Vector columnData)
  - void addRow(Object[] rowData) / void addRow(Vector rowData)
  - void insertRow(int row, Object[] rowData) / void insertRow(int row, Vector rowData)
  - void removeRow(int row)
  - void moveRow(int start, int end, int to)
  - int getColumnCount() / void setColumnCount(int columnCount)
  - int getRowCount() / void setNumRows(int rowCount) / void setRowCount(int rowCount)
  - String getColumnName(int column)
  - Vector getDataVector() / void setDataVector(Object[][] dataVector, Object[] columnIdentifiers) / void setDataVector(Vector dataVector, Vector columnIdentifiers)
  - Object getValueAt(int row, int column) / void setValueAt(Object aValue, int row, int column)
  - boolean isCellEditable(int row, int column)
  - void newDataAvailable(TableModelEvent event) / void newRowsAdded(TableModelEvent e) / void rowsRemoved(TableModelEvent event)
  - void setColumnIdentifiers(Object[] newIdentifiers) / void setColumnIdentifiers(Vector columnIdentifiers)

JTable : la classe TableModelListener

- 1 méthode

  void tableChanged(TableModelEvent e)

- La classe TableModelEvent
  - int getColumn()
  - int getFirstRow()
  - int getLastRow()
  - int getType()

    type d'événement : INSERT, UPDATE and DELETE.

La classe JTable : Constructeurs

- JTable()
- JTable(int numRows, int numColumns)
- JTable(Object[][] rowData, Object[] columnNames)
- JTable(TableModel dm)
- JTable(TableModel dm, TableColumnModel cm)
- JTable(TableModel dm, TableColumnModel cm, ListSelectionModel sm)
- JTable(Vector rowData, Vector columnNames)

Méthodes de JTable : données

- Modèle des données
  - TableModel getModel()
  - void setModel(TableModel dataModel)
- Données
  - Object getValueAt(int row, int column)
  - void setValueAt(Object aValue, int row, int column)
  - int getRowCount()
- Colonnes
  - void removeColumn(TableColumn aColumn)
  - void setAutoCreateColumnsFromModel(boolean autoCreateColumnsFromModel)
  - void moveColumn(int column, int targetColumn)
  - void addColumn(TableColumn aColumn)
  - void createDefaultColumnsFromModel()
  - boolean getAutoCreateColumnsFromModel()
  - TableColumn getColumn(Object identifier)
  - int getColumnCount()
  - String getColumnName(int column)
  - Class<?> getColumnClass(int column)
- Modèle des colonnes
  - TableColumnModel getColumnModel()
  - void setColumnModel(TableColumnModel columnModel)

Méthodes de JTable : mise en forme (1)

- Conversion des indices affichage/données
  - int convertColumnIndexToModel(int viewColumnIndex)
  - int convertColumnIndexToView(int modelColumnIndex)
  - int convertRowIndexToModel(int viewRowIndex)
  - int convertRowIndexToView(int modelRowIndex)

- Grille
  - Color getGridColor() / void setGridColor(Color gridColor)
  - boolean getShowHorizontalLines() / void setShowHorizontalLines(bolean b)
  - boolean getShowVerticalLines() / void setShowVerticalLines(boolean b)
  - void setShowGrid(boolean showGrid)

- Tailles variées
  - int getRowHeight() / void setRowHeight(int rowHeight)
  - int getRowHeight(int row) / void setRowHeight(int row, int rowHeight)
  - int getRowMargin() / void setRowMargin(int rowMargin)
  - Dimension getIntercellSpacing()
  - void setIntercellSpacing(Dimension intercellSpacing)

Méthodes de JTable : mise en forme (2)

- Défilement
  - int getScrollableBlockIncrement(Rectangle visibleRect, int orientation, int direction)
  - boolean getScrollableTracksViewportHeight()
  - boolean getScrollableTracksViewportWidth()
  - Dimension getPreferredScrollableViewportSize()
  - void setPreferredScrollableViewportSize(Dimension size)
  - int getScrollableUnitIncrement(Rectangle visibleRect, int orientation, int direction)
- UI
  - TableUI getUI() / void setUI(TableUI ui) / void updateUI() / String getUIClassID()
- En-tête
  - JTableHeader getTableHeader() / void setTableHeader(JTableHeader tableHeader)
- Autre
  - int getAutoResizeMode() / void setAutoResizeMode(int mode)
  - boolean getFillsViewportHeight() / void setFillsViewportHeight(boolean b)
  - boolean getSurrendersFocusOnKeystroke()
  - void setSurrendersFocusOnKeystroke(boolean b)
  - String getToolTipText(MouseEvent event)

Méthodes de JTable : divers

- Edition
  - boolean editCellAt(int row, int column)
  - boolean editCellAt(int row, int column, EventObject e)
  - int getEditingColumn() / void setEditingColumn(int aColumn)
  - int getEditingRow() / void setEditingRow(int aRow)
  - boolean isCellEditable(int row, int column)
  - boolean isEditing()

- Tri
  - boolean getAutoCreateRowSorter()
  - RowSorter<? extends TableModel> getRowSorter()
  - void setAutoCreateRowSorter(boolean autoCreateRowSorter)
  - void setRowSorter(RowSorter<? extends TableModel> sorter)

- Coordonnées
  - Rectangle getCellRect(int row, int column, boolean includeSpacing)
  - int columnAtPoint(Point point) / int rowAtPoint(Point point)

Méthodes de JTable : éditeurs/afficheurs

- Editeurs
  - TableCellEditor getCellEditor()
  - TableCellEditor getCellEditor(int row, int column)
  - TableCellEditor getDefaultEditor(Class<?> columnClass)
  - Component prepareEditor(TableCellEditor editor, int row, int column)
  - void setDefaultEditor(Class<?> columnClass, TableCellEditor editor)
  - void setCellEditor(TableCellEditor anEditor)
  - void removeEditor()
  - Component getEditorComponent()

- Afficheurs
  - void createDefaultRenderers()
  - TableCellRenderer getCellRenderer(int row, int column)
  - TableCellRenderer getDefaultRenderer(Class<?> columnClass)
  - Component prepareRenderer(TableCellRenderer renderer, int row, int column)
  - void setDefaultRenderer(Class<?> columnClass, TableCellRenderer renderer)

Méthodes de JTable : sélection (1)

- Général
  - void clearSelection() / void selectAll()
  - boolean getCellSelectionEnabled() / void setCellSelectionEnabled(boolean b)
  - Color getSelectionBackground() / void setSelectionBackground(Color col)
  - Color getSelectionForeground() / void setSelectionForeground(Color col)
  - boolean isCellSelected(int row, int column)
  - void setSelectionMode(int selectionMode)

- Colonnes
  - void addColumnSelectionInterval(int index0, int index1)
  - boolean getColumnSelectionAllowed() / void setColumnSelectionAllowed(boolean b)
  - int getSelectedColumn() / int[] getSelectedColumns()
  - boolean isColumnSelected(int col)
  - int getSelectedColumnCount()
  - void removeColumnSelectionInterval(int index0, int index1)
  - void setColumnSelectionInterval(int index0, int index1)

Méthodes de JTable : sélection (2)

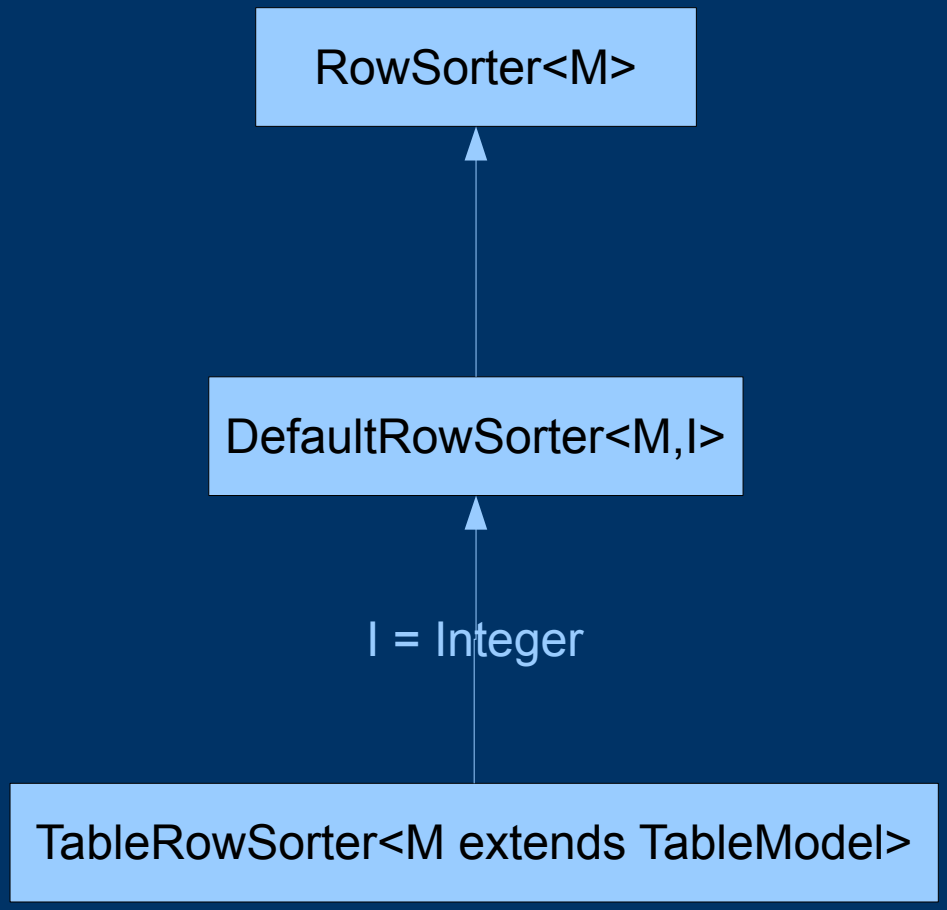
- Lignes
  - void addRowSelectionInterval(int index0, int index1)
  - void removeRowSelectionInterval(int index0, int index1)
  - void setRowSelectionInterval(int index0, int index1)
  - void changeSelection(int rowIndex, int columnIndex, boolean toggle, boolean extend)
  - int getSelectedRow() / int[] getSelectedRows()
  - boolean isRowSelected(int row)
  - int getSelectedRowCount()
  - ListSelectionModel getSelectionModel() /void setSelectionModel(ListSelectionModel M)
  - boolean getRowSelectionAllowed() / void setRowSelectionAllowed(boolean b)
  - void setUpdateSelectionOnSort(boolean b) / boolean getUpdateSelectionOnSort()

Méthodes de JTable : événements

- Interfaces notamment implantées par JTable :
  - TableModelListener
  - TableColumnModelListener
  - ListSelectionListener
  - CellEditorListener
  - RowSorterListener

- Méthodes associées :
  - void columnAdded(TableColumnModelEvent e)
  - void columnMarginChanged(ChangeEvent e)
  - void columnMoved(TableColumnModelEvent e)
  - void columnRemoved(TableColumnModelEvent e)
  - void columnSelectionChanged(ListSelectionEvent e)
  - void editingCanceled(ChangeEvent e)
  - void editingStopped(ChangeEvent e)
  - void tableChanged(TableModelEvent e)
  - void valueChanged(ListSelectionEvent e)
  - void sorterChanged(RowSorterEvent e)

Tri de lignes



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)

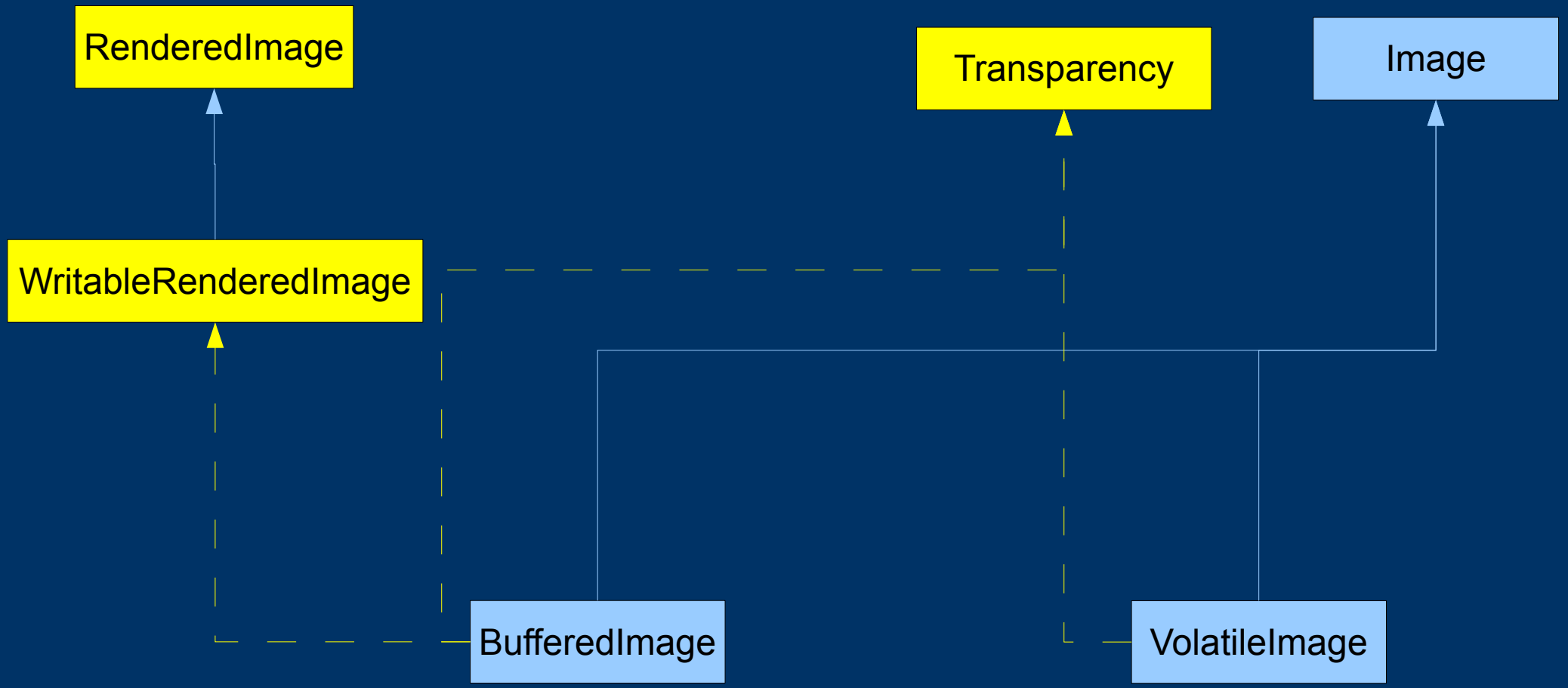
- ImageIcon(String filename)
- ImageIcon(String filename, String description)

- 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 "  "
- 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)

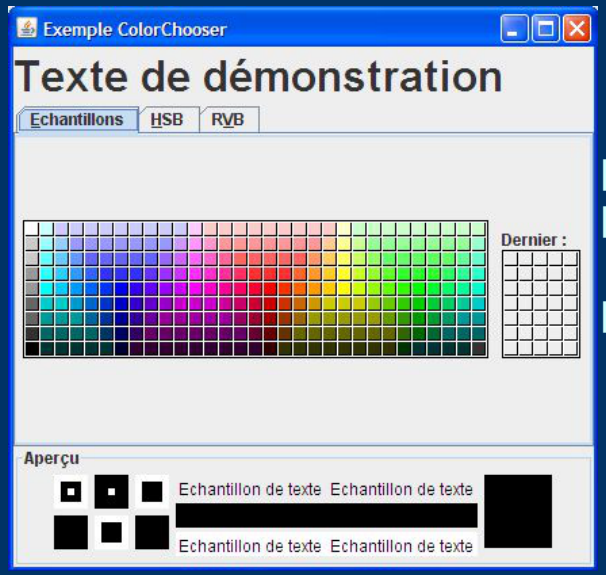
- static Color getHSB(float h, float s, float b)

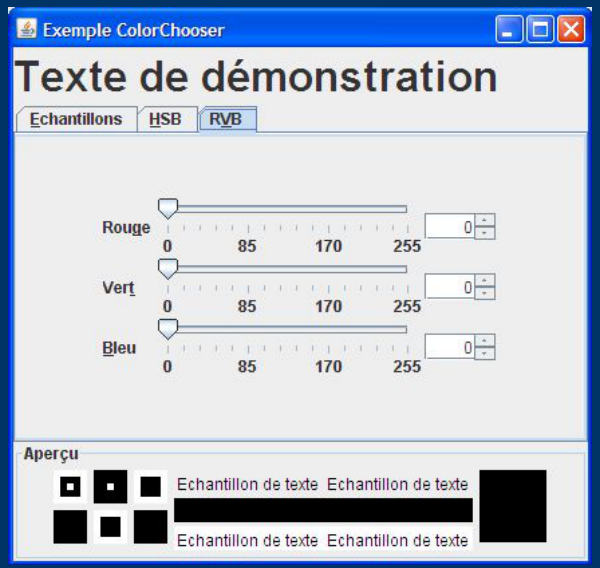
  Obtenir une couleur à partir de ses teinte, saturation, luminosité

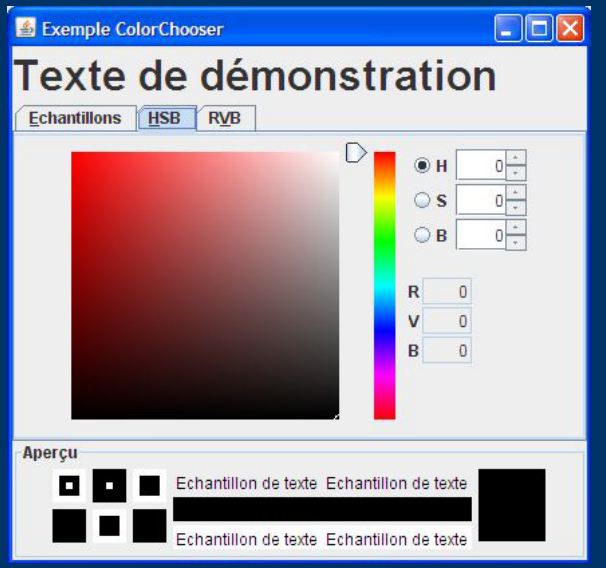
Classe JColorChooser
Introduction

- 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)

Accessibilité

Bruno Mermet
Septembre 2010

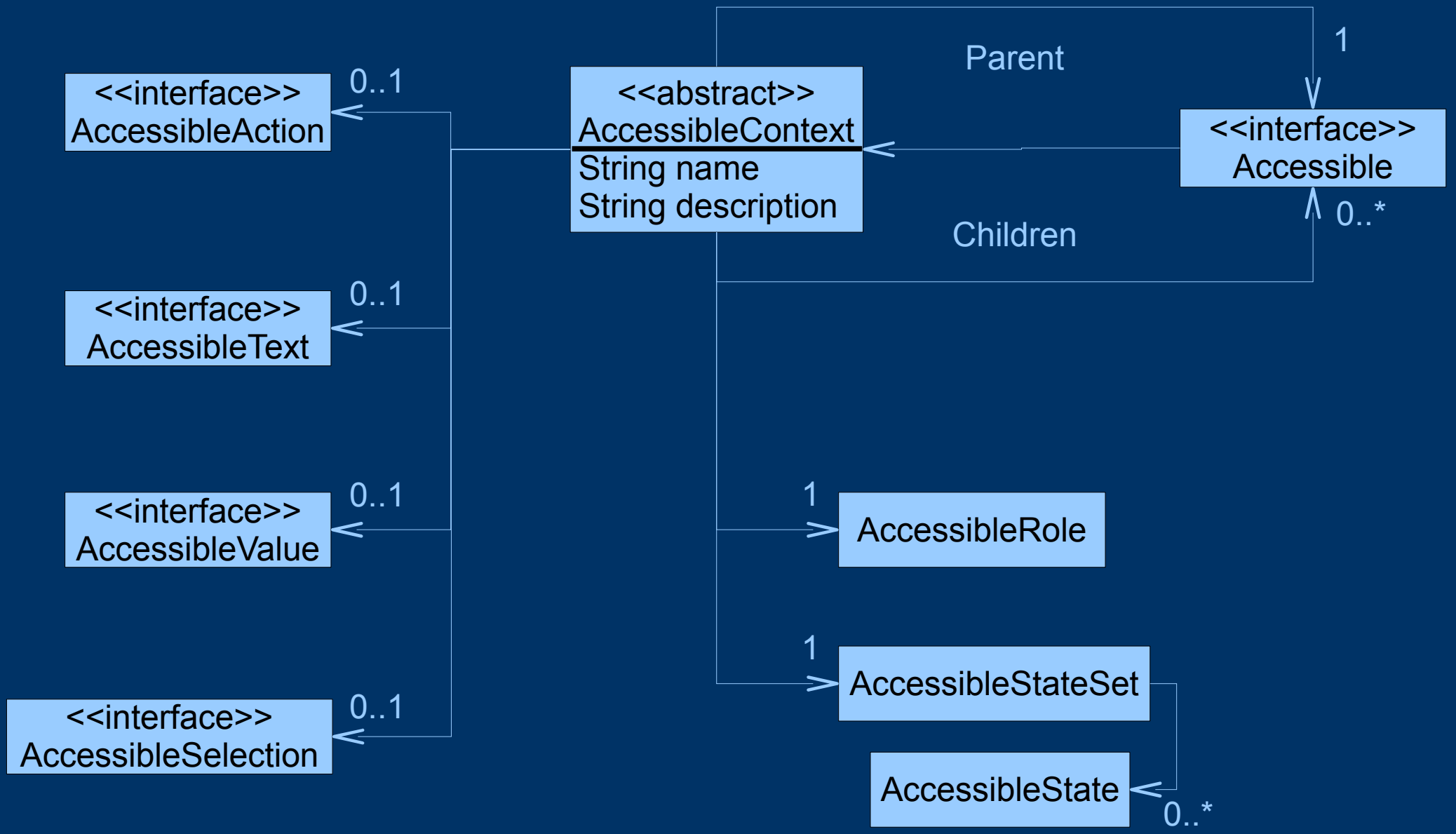
Introduction

- Motivation

  Un logiciel de qualité se doit d'être accessible au plus grand nombre, y compris les personnes souffrant de déficience physique, quelle qu'elle soit

- Principe général
  - Les JFC (Java Foundation Classes) incluent une API dédiées (javax.accessibility)
  - Les logiciels dédiés dans l'aide à tel ou tel type de handicap peuvent bénéficier de cette API, à condition que les interfaces importantes soient implantées par les composants

Principe simplifié illustré



Mise en oeuvre de base dans Swing

- La plupart des composants Swing implantent javax.Accessible
- Le nom est positionné automatiquement pour les composants où cela est significatif (JButton par exemple)
- La description est définie automatiquement lors de l'ajout d'une bulle d'aide (ToolTipText)

Effort élémentaire à faire

- Spécifier un nom et une description aux conteneurs significatifs (exemple : groupe de boutons radio)
- Associer les étiquettes aux composants qu'ils décrivent avec setLabelFor()
- Associer un nom et une description à tout ce qui ne représenter que par une image
- Définir des codes mnémoniques à tous les items de menus et tous les composants des boîtes de dialogue
- Définir des raccourcis à toutes les commandes fréquentes

Effort supplémentaire à fournir

- Rendre les couleurs, polices et tailles de caractères paramétrables
- Respecter les standards des IHM
- Implanter Accessible pour tous les composants « maison »
- Tester soi-même
- Faire tester par des personnes présentant différentes déficiences