Mise en œuvre d'une barre de menu

Pour mettre en œuvre une barre de menu, plusieurs classes de la librairie Swing sont nécessaire. Les principales classes sont les suivantes.

- **javax.swing.JMenuBar :**

  - ✓ Une instance de cette classe représente la barre de menu à proprement parler. Une barre de menu ne s'injecte pas dans le « content pane ».
  - ✓ Elle possède son propre emplacement privilégié au sein de la fenêtre, situé juste au-dessus du « content pane ».
  - ✓ Pour placer la barre de menu dans cet emplacement dédié, on utilise la méthode **setJMenuBar**

- **javax.swing.JMenu :**

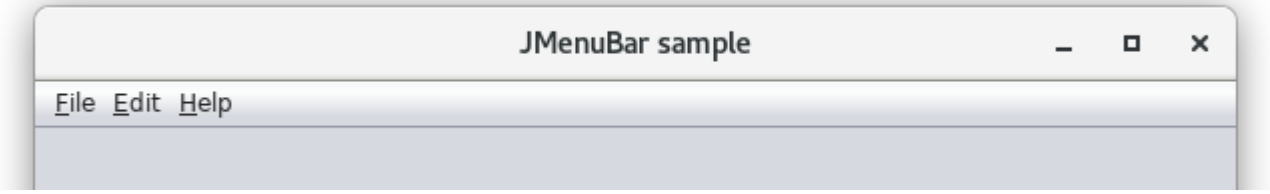


Figure 1: Simple JMenuBar

Elle représente un menu déroulant (« File » ou « Edit » par exemple). Un tel menu peut être placé soit la barre de menu principal, soit à l'intérieur d'un autre JMenu (pour obtenir des sous-menu déroulants).

On peut associer à un JMenu un mnémonique permettant l'ouverture du menu déroulant par le clavier en appuyant simultanément sur ALT et la touche correspondante à ce mnémonique. Une présence d'un mnémonique se repère grâce à la lettre surlignée dans le texte du menu (voir la capture d'écran ci-dessus ; Le F du menu File est son mnémonique).

- **javax.swing.JMenuItem :**

Elle représente un élément de menu (menu item, en anglais), donc un élément cliquable pouvant exécuter un traitement.

Un JMenuItem peut se voir associer plusieurs « décorations » comme un mnémonique (une lettre pouvant atteindre l'élément de menu quand il est affiché), un accélérateur permettant un accès direct à l'élément de menu (via la classe KeyStroke) et une icône graphique.

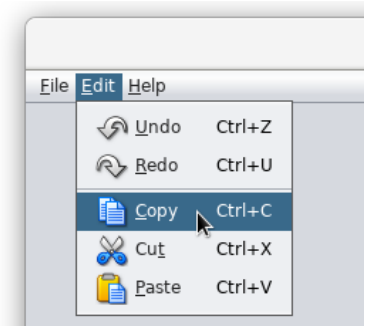


Figure 2: Eléments de menu

- **javax.swing.ImageIcon** :

Elle permet de définir une icône graphique associée à un JMenuItem.

- **javax.swing.KeyStroke** :

Elle permet de définir un accélérateur associé à un JMenuItem (par exemple CTRL+C pour lancer une sauvegarde d'un document en cours d'édition).

Décoration d'un JMenuItem

Il est donc possible de décorer un élément de menu, comme en atteste la capture d'écran suivante.

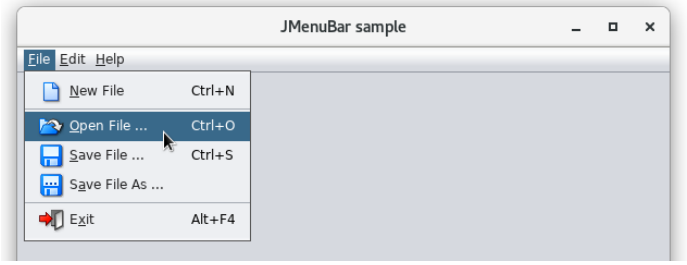


Figure 3: Décoration d'un menu

Utilisation d'un mnémonique

Pour ajouter un mnémonique, permettant l'ouverture du menu déroulant par le clavier en appuyant simultanément sur ALT et la touche correspondante à ce mnémonique, il vous faut invoquer la méthode setMnemonic. Elle prend en paramètre le caractère (et non pas une chaîne) associé au menu : ce caractère doit être présent dans la chaîne.

```
1  JMenuItem mnuNewFile = new JMenuItem( "New File" );
2  mnuNewFile.setMnemonic( 'N');
```

Pour activer cet élément de menu, vous pourrez taper **ALT+F** pour ouvrir le menu File (qui a son propre mnémonique) puis ensuite directement **N** pour activer l'élément de menu New File.

Utilisation d'un accélérateur (KeyStroke)

Un accélérateur est un moyen d'accès encore plus rapide que le mnémonique. En effet, il n'est plus nécessaire d'ouvrir le menu contenant notre élément de menu : la séquence de touche lance directement le traitement associé (le listener).

```
JMenuItem mnuNewFile = new JMenuItem ("New File" ) ;
mnuNewFile.setAccelerator(KeyStroke.getKeyStroke(KeyEvent.VK_N, KeyEvent.CTRL_DOWN_MASK)) ;
```

Association d'une icône

Pour rendre votre application esthétique, il est vivement conseillé d'utiliser un jeu d'icônes élégantes. Il y a de nombreux jeux d'icônes disponible sur Internet.

```
JMenuItem mnuNewFile = new JMenuItem( "New File" );
mnuNewFile.setIcon(new ImageIcon ( "icons/new.png" ) );
```

Association d'un gestionnaire d'événements

Un élément de menu (class **javax.swing.JMenuItem**) se comporte comme un bouton et c'est normal : la classe **JMenuItem** dérive de la classe **AbstractButton** (la classe mère de **JButton**).

On utilise donc un **ActionListener** pour réagir à un clic sur un élément de menu.

Pour l'implémentation du listener, c'est comme vous voulez : soit une classe anonyme, soit vous utilisez les nouvelles syntaxes Java 8.0 permettant de produire automatiquement cette classe anonyme (lambda ou référence sur méthode). Personnellement, je trouve que la syntaxe des références sur méthodes permet d'obtenir un code concis et clair quand-à la définition d'un ActionListener. Dans un premier temps, il faut définir une méthode acceptant un ActionEvent en paramètre. Voici un exemple.

Tableau 1:Définition d'une méthode pour traiter le clic sur un élément de menu

```
private void mnuNewListener( ActionEvent event ) {
    JOptionPane.showMessageDialog( this, "New File invoked" );
}
```

Il ne reste plus qu'à enregistrer l'écouteur qui va être produit grâce à la référence sur méthode (basée, bien entendu, sur la méthode précédemment définie).

```
JMenuItem mnuNewFile = new JMenuItem( "New File" );
// ...
mnuNewFile.addActionListener( this::mnuNewListener );
mnuFile.add(mnuNewFile);
```

Enregistrement du gestionnaire d'événements

L'exemple de code complet

```
import javax.swing.*;
import javax.swing.plaf.nimbus.NimbusLookAndFeel;
import java.awt.event.ActionEvent;
import java.awt.event.KeyEvent;

public class MenuBarSample extends JFrame {
   /* Construction de l'interface graphique */
   public MenuBarSample(){
      super( "JMenuBar sample" );
      this.setSize(600,400);
      this.setLocationRelativeTo( null );
      this.setDefaultCloseOperation( DISPOSE_ON_CLOSE );
      // Construction et injection de la barre de menu
      this.setJMenuBar( this.createMenuBar() );
```

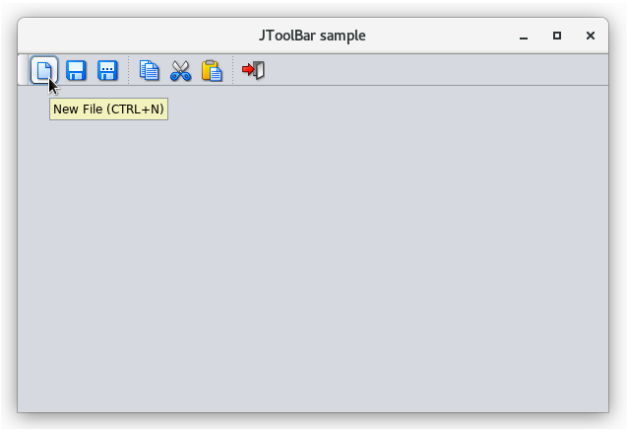
```
}
/* Methode de construction de la barre de menu */
private JMenuBar createMenuBar() {
    // La barre de menu à proprement parler
    JMenuBar menuBar = new JMenuBar();

    // Définition du menu déroulant "File" et de son contenu
    JMenu mnuFile = new JMenu( "File" );
    mnuFile.setMnemonic( 'F' );

    JMenuItem mnuNewFile = new JMenuItem( "New File" );
    mnuNewFile.setIcon( new ImageIcon("icons/new.png") );
    mnuNewFile.setMnemonic( 'N' );
    mnuNewFile.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_N, KeyEvent.CTRL_DOWN_MASK) );
    mnuNewFile.addActionListener( this::mnuNewListener );
    mnuFile.add(mnuNewFile);

    mnuFile.addSeparator();

    JMenuItem mnuOpenFile = new JMenuItem( "Open File ..." );
    mnuOpenFile.setIcon( new ImageIcon("icons/open.png") );
    mnuOpenFile.setMnemonic( 'O' );
    mnuOpenFile.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_O, KeyEvent.CTRL_DOWN_MASK) );
    mnuFile.add(mnuOpenFile);

    JMenuItem mnuSaveFile = new JMenuItem( "Save File ..." );
    mnuSaveFile.setIcon( new ImageIcon("icons/save.png") );
    mnuSaveFile.setMnemonic( 'S' );
    mnuSaveFile.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_S, KeyEvent.CTRL_DOWN_MASK) );
    mnuFile.add(mnuSaveFile);

    JMenuItem mnuSaveFileAs = new JMenuItem( "Save File As ..." );
    mnuSaveFileAs.setIcon( new ImageIcon("icons/save_as.png") );
    mnuSaveFileAs.setMnemonic( 'A' );
    mnuFile.add(mnuSaveFileAs);

    mnuFile.addSeparator();

    JMenuItem mnuExit = new JMenuItem( "Exit" );
    mnuExit.setIcon( new ImageIcon("icons/exit.png") );
    mnuExit.setMnemonic( 'x' );
    mnuExit.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_F4, KeyEvent.ALT_DOWN_MASK) );
    mnuFile.add(mnuExit);

    menuBar.add(mnuFile);

    // Définition du menu déroulant "Edit" et de son contenu
    JMenu mnuEdit = new JMenu( "Edit" );
    mnuEdit.setMnemonic( 'E' );

    JMenuItem mnuUndo = new JMenuItem( "Undo" );
    mnuUndo.setIcon( new ImageIcon("icons/undo.png") );
    mnuUndo.setMnemonic( 'U' );
    mnuUndo.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_Z, KeyEvent.CTRL_DOWN_MASK) );
    mnuEdit.add(mnuUndo);

    JMenuItem mnuRedo = new JMenuItem( "Redo" );
    mnuRedo.setIcon( new ImageIcon("icons/redo.png") );
    mnuRedo.setMnemonic( 'R' );
    mnuRedo.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_U, KeyEvent.CTRL_DOWN_MASK) );
    mnuEdit.add(mnuRedo);
```

```
        mnuEdit.addSeparator();

        JMenuItem mnuCopy = new JMenuItem( "Copy" );
        mnuCopy.setIcon( new ImageIcon("icons/copy.png") );
        mnuCopy.setMnemonic( 'C' );
        mnuCopy.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_C, KeyEvent.CTRL_DOWN_MASK) );
        mnuEdit.add(mnuCopy);

        JMenuItem mnuCut = new JMenuItem( "Cut" );
        mnuCut.setIcon( new ImageIcon("icons/cut.png") );
        mnuCut.setMnemonic( 't' );
        mnuCut.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_X, KeyEvent.CTRL_DOWN_MASK) );
        mnuEdit.add(mnuCut);

        JMenuItem mnuPaste = new JMenuItem( "Paste" );
        mnuPaste.setIcon( new ImageIcon("icons/paste.png") );
        mnuPaste.setMnemonic( 'P' );
        mnuPaste.setAccelerator( KeyStroke.getKeyStroke(KeyEvent.VK_V, KeyEvent.CTRL_DOWN_MASK) );
        mnuEdit.add(mnuPaste);

        menuBar.add(mnuEdit);

        // Définition du menu déroulant "Help" et de son contenu
        JMenu mnuHelp = new JMenu( "Help" );
        mnuHelp.setMnemonic( 'H' );

        menuBar.add( mnuHelp );
        return  menuBar;
    }

    private void mnuNewListener(ActionEvent actionEvent) {
        JOptionPane.showMessageDialog( this, "Button clicked !" );
    }

    public static void main(String[] args) throws Exception{
        UIManager.setLookAndFeel(new NimbusLookAndFeel());
       MenuBarSample fen = new MenuBarSample();
       fen.setVisible(true);
    }
}
```

Mise en œuvre d'une barre d'outils

Positionnement de votre barre d'outils

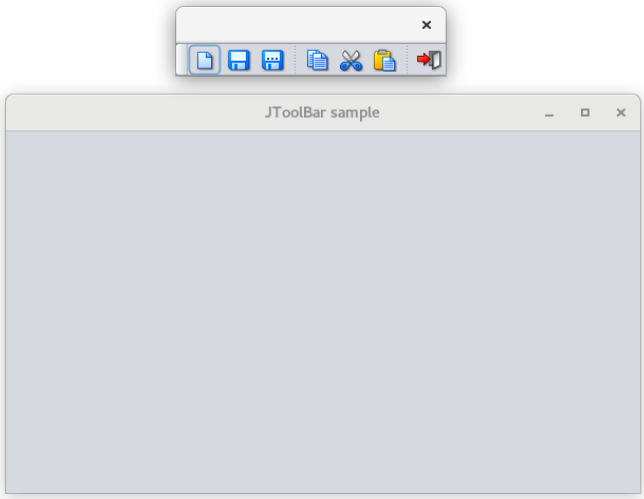
Une barre d'outils, une instance de la classe **javax.swing.ToolBar**, est un conteneur de composants graphiques. Traditionnellement, une barre d'outils est positionnée en haut de la fenêtre. Pour autant, une barre d'outils est, par défaut, « dockable » : c'est à dire qu'on peut la décrocher de la fenêtre ou la repositionner sur un côté quelconque.



En fait, une barre d'outils se positionne dans un conteneur Swing associé à une stratégie de positionnement de type **java.awt.BorderLayout**. Si une des quatre zones latérales est inoccupée, alors on pourra y placer la barre.

On peut, durant l'exécution de l'application, positionner la souris dessus le « gripper » de la barre d'outils et la déplacer d'une zone à une autre (à condition que cette zone soit inoccupée) par simple drag'n drop. Il est même possible de détacher la barre de sa fenêtre, comme en atteste la capture d'écran suivante.

N.B : En cliquant sur la croix de la fenêtre associée à la barre d'outils, on va la repositionner à l'intérieur de la fenêtre principale.



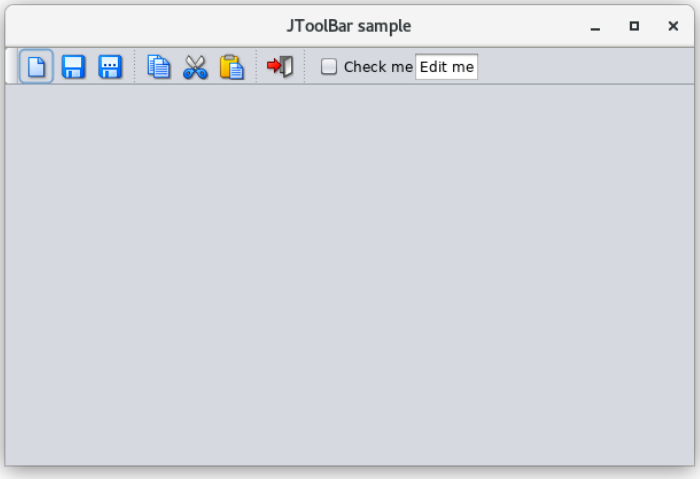
Une barre d'outils peut contenir divers composants

On peut placer autre chose que des boutons dans la barre d'outils. Par exemple, il est possible d'y ajouter une zone de saisie de texte, une liste déroulante, des cases à cocher, ... On peut par exemple imaginer l'extrait de code suivant.

```
JToolBar toolBAr = new JToolBar();
toolBar.add( new JButton( new ImageIcon( "icons/aFile.png" ) ) );
toolBar.add( new JCheckBox( "Check me" ) );
toolBar.add( new JTextBox( "Edit me" ) );
```

Ajout de composants graphiques dans votre barre d'outils

Et voici un visuel d'une barre d'outils contenant différents types de composants graphiques.



L'exemple de code

```
import javax.swing.*;
import javax.swing.plaf.nimbus.NimbusLookAndFeel;
import java.awt.*;
import java.awt.event.ActionEvent;

public class ToolBarSample extends JFrame {
   /* Construction de l'interface graphique */
   public ToolBarSample() {
      super( "JToolBar sample" );
      this.setSize(600,400);
      this.setLocationRelativeTo( null );
      this.setDefaultCloseOperation( DISPOSE_ON_CLOSE );

      // Construction et injection de la barre d'outils
      JPanel contentPane = (JPanel) getContentPane();
      contentPane.add( this.createToolBar(), BorderLayout.NORTH );
   }
   /* Méthode de construction de la barre d'outils */
   private Component createToolBar() {
      // La barre d'outils à proprement parler
      JToolBar toolBar = new JToolBar();

      JButton btnNew = new JButton( new ImageIcon( "icons/new.png") );
      btnNew.setToolTipText( "New File (CTRL+N)" );
      btnNew.addActionListener( this::btnNewListener );
      toolBar.add( btnNew );

      JButton btnSave = new JButton( new ImageIcon( "icons/save.png" ) );
      btnSave.setToolTipText( "Save (CTRL+S)" );
      toolBar.add( btnSave );

      JButton btnSaveAs = new JButton( new ImageIcon( "icons/save_as.png" ) );
      btnSaveAs.setToolTipText( "Save As..." );
      toolBar.add( btnSaveAs );

      toolBar.addSeparator();

      JButton btnCopy = new JButton( new ImageIcon( "icons/copy.png") );
      btnCopy.setToolTipText( "Copy (CTRL+C)" );
      toolBar.add( btnCopy );

      JButton btnCut = new JButton( new ImageIcon( "icons/cut.png") );
      btnCut.setToolTipText( "Cut (CTRL+X)" );
      toolBar.add( btnCut );

      JButton btnPaste = new JButton( new ImageIcon( "icons/paste.png") );
      btnPaste.setToolTipText( "Paste (CTRL+V)" );
      toolBar.add( btnPaste );

      toolBar.addSeparator();

      JButton btnExit = new JButton( new ImageIcon( "icons/exit.png") );
      btnExit.setToolTipText( "Exit (ALT+F4)" );
      toolBar.add( btnExit );

      toolBar.addSeparator();

      // Autres types de composants graphiques
      toolBar.add( new JButton( new ImageIcon( "icons/aFile.png" ) ) );
      toolBar.add( new JCheckBox( "Check me" ) );
      toolBar.add( new JTextField( "Edit me" ) );
```

```
        return toolBar;
    }

    private void btnNewListener(ActionEvent actionEvent) {
        JOptionPane.showMessageDialog( this, "Button clicked !" );
    }

    public static void main(String[] args) throws Exception{
        UIManager.setLookAndFeel( new NimbusLookAndFeel() );
        ToolBarSample frame = new ToolBarSample();
        frame.setVisible( true );
    }


}
```