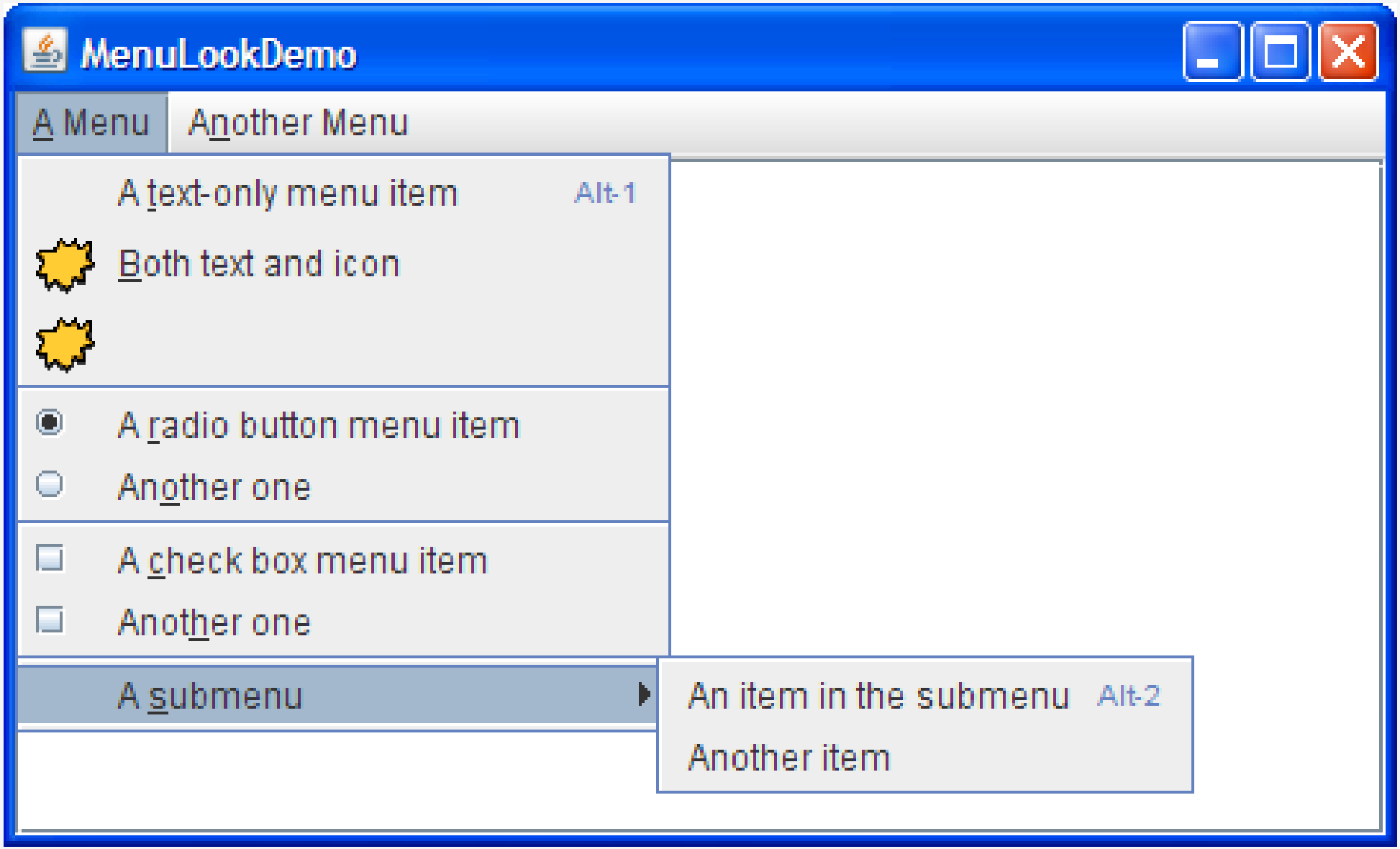
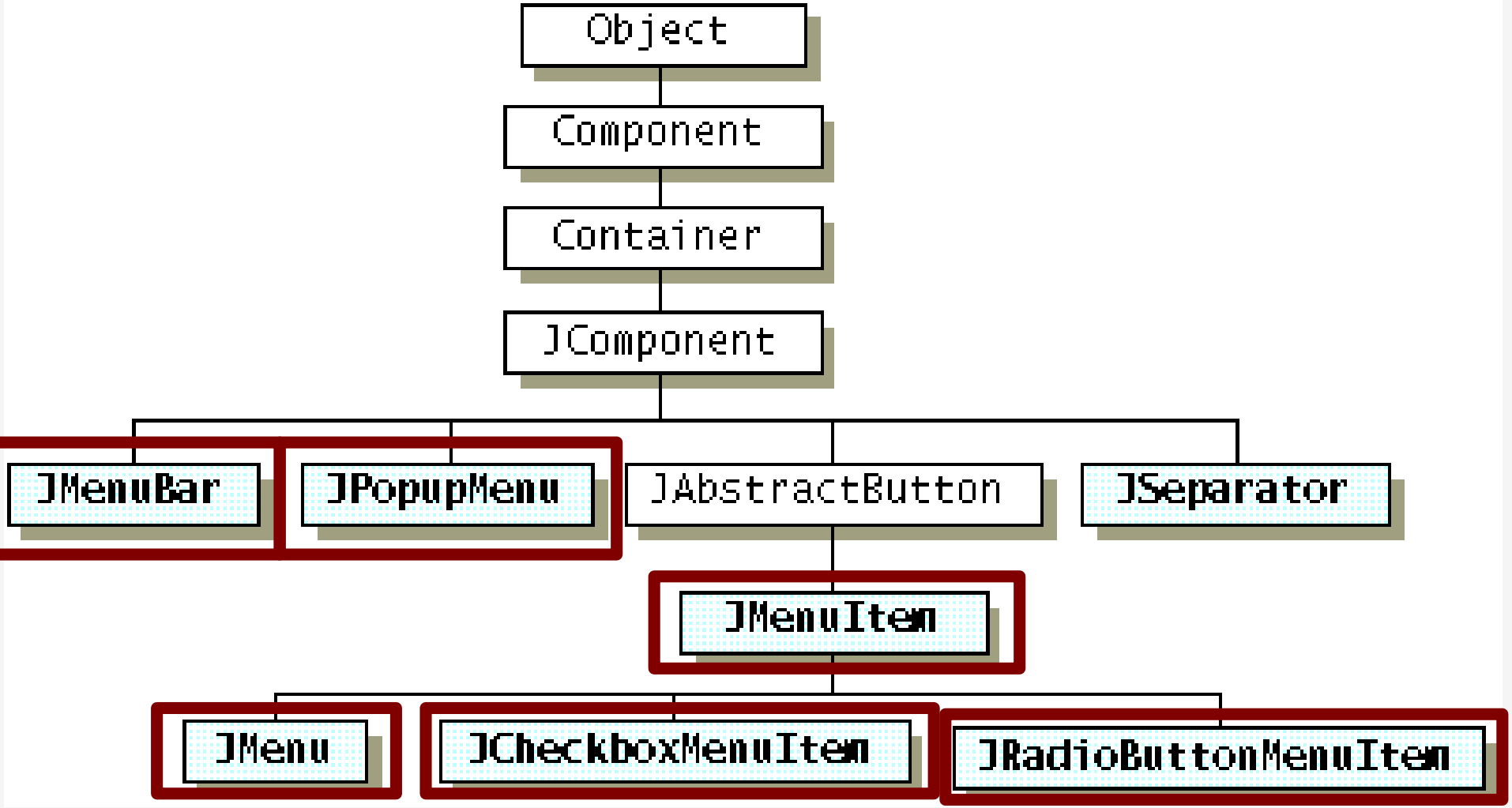
Menus, les bases
Utilisation des menus spéciaux

Menus


MenuLookDemo
A Menu
Another Menu
A text-only menu item
Alt-1
Both text and icon
A radio button menu item
Another one
A check box menu item
Another one
A submenu
An item in the submenu
Alt-2
Another item

Hiérarchie des composants



javax.swing. JMenuBar JMenu

```
public Appli(){

    JMenuBar menuBar;
    menuBar = new JMenuBar();

    JMenu menu;
    menu = new JMenu("File");
    //touche raccourci
    menu.setMnemonic(KeyEvent.VK_A);

    menuBar.add(menu);

    setJMenuBar(menuBar);
}
```

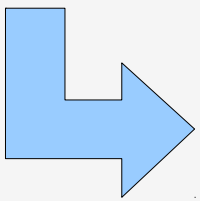
Barre de menu OBLIGATOIRE

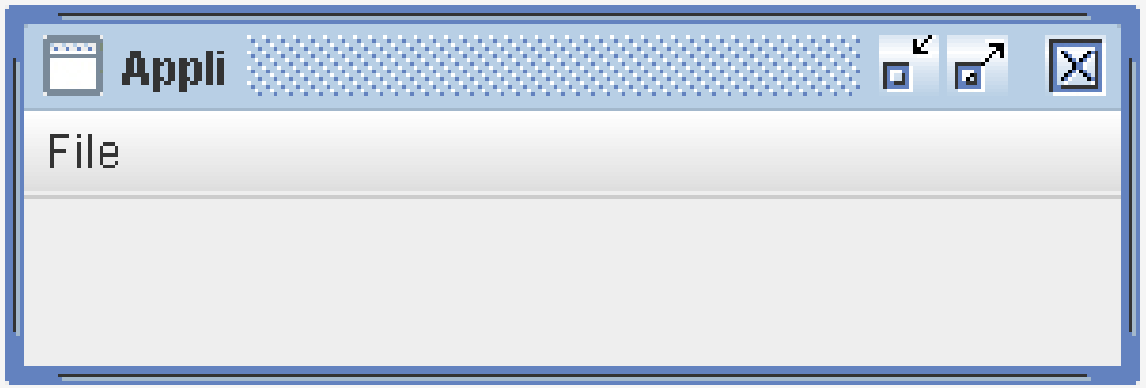
Un menu

raccourcis clavier pour le menu (alt)

ajout dans la barre

ajout de la barre dans la JFrame

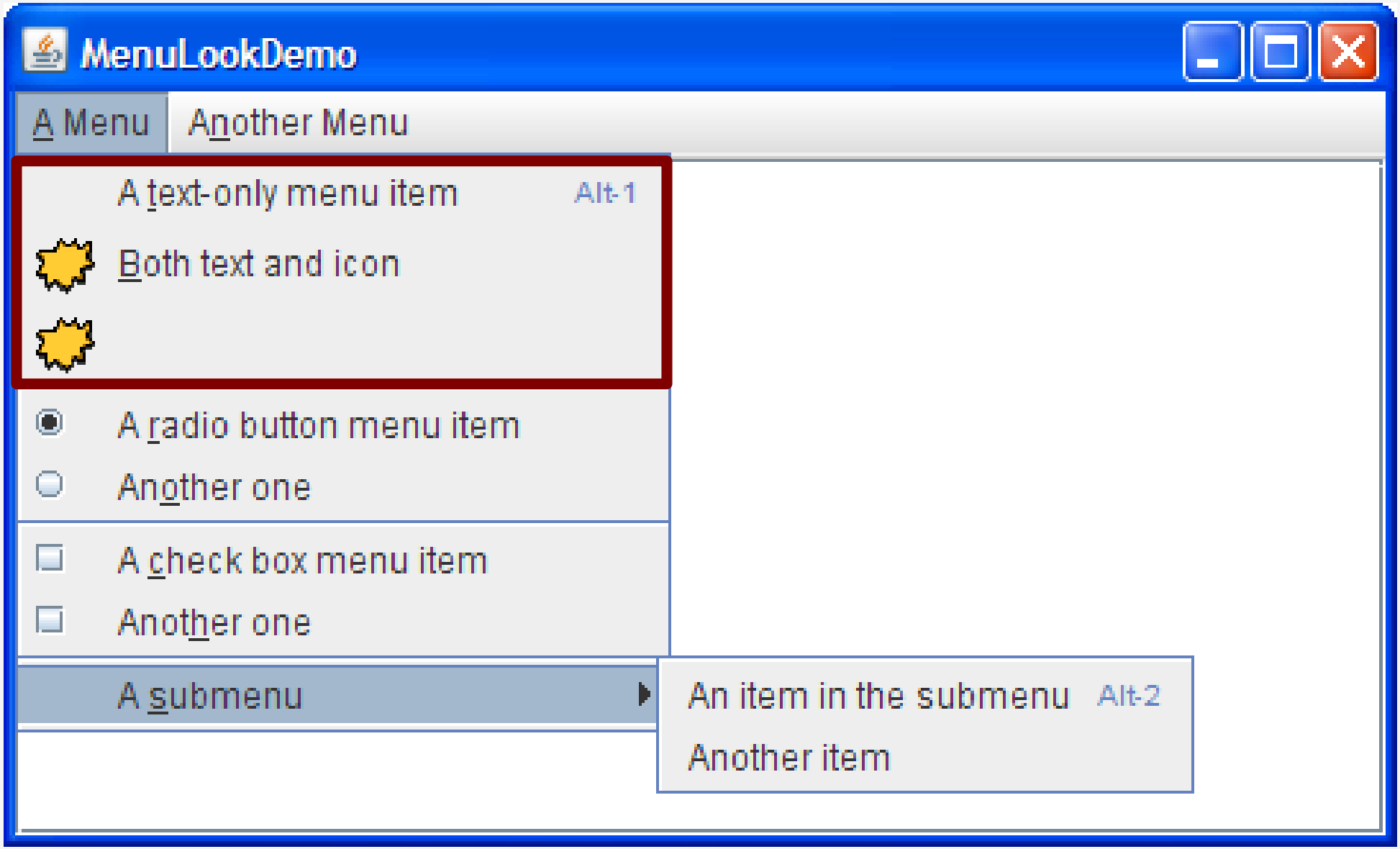




javax.swing.JMenuItem

- Il faut maintenant mettre des éléments dans le menu
- L'élément le plus courant est le ***Javax.swing.JMenuItem***
- Les ***JMenuItems*** sont des boutons (extends **javax.swing.AbstractButton**) !

JMenuItem


MenuLookDemo
A Menu
Another Menu
A text-only menu item
Alt-1
Both text and icon
A radio button menu item
Another one
A check box menu item
Another one
A submenu
An item in the submenu
Alt-2
Another item

javax.swing.JMenuItem

Constructor Summary

JMenuItem()

Creates a JMenuItem with no set text or icon.

JMenuItem(Action a)

Creates a menu item whose properties are taken from the specified Action.

JMenuItem(Icon icon)

Creates a JMenuItem with the specified icon.

JMenuItem(String text)

Creates a JMenuItem with the specified text.

JMenuItem(String text, Icon icon)

Creates a JMenuItem with the specified text and icon.

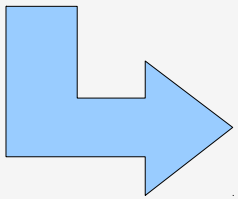
JMenuItem(String text, int mnemonic)

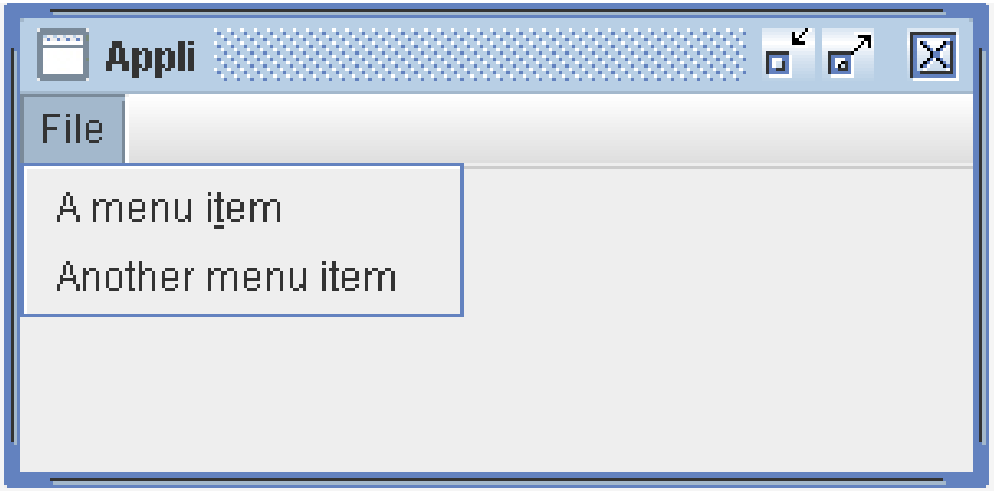
Creates a JMenuItem with the specified text and keyboard mnemonic.

javax.swing.JMenuItem

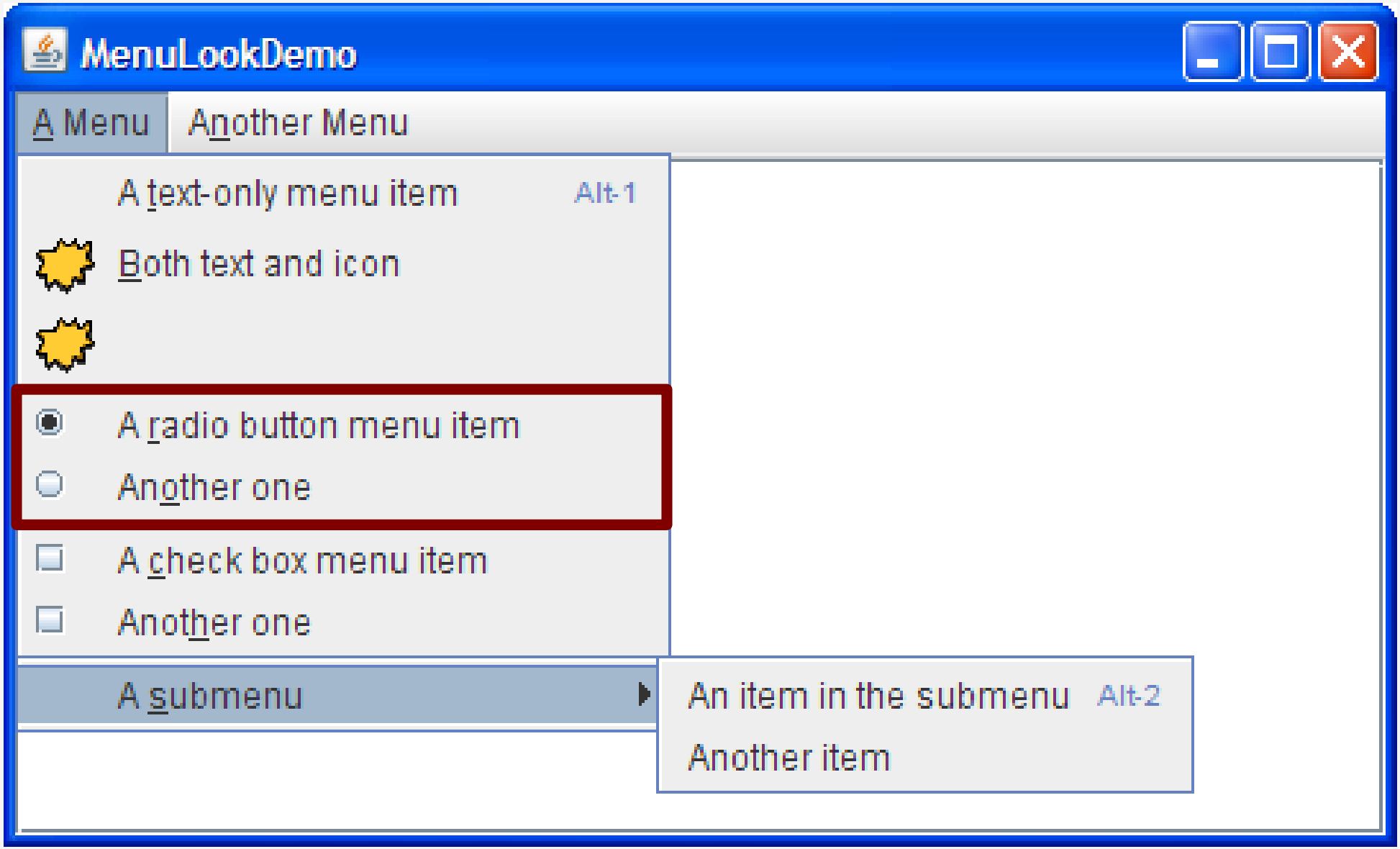
```
JMenuBar menuBar = new JMenuBar();

JMenu menu = new JMenu("File");

JMenuItem menuItem;
menuItem = new JMenuItem("A menu item",KeyEvent.VK_T);
menu.add(menuItem);

menuItem = new JMenuItem("Another menu item");
menu.add(menuItem);

menuBar.add(menu);
setJMenuBar(menuBar);
```

raccourcis clavier pour la navigation





JRadioButtonMenuItem



JRadioButtonMenuItem

javax.swing

Class JRadioButtonMenuItem

java.lang.Object
  └ java.awt.Component
    └ java.awt.Container
      └ javax.swing.JComponent
        └ javax.swing.AbstractButton
          └ javax.swing.JMenuItem
            └ **javax.swing.JRadioButtonMenuItem**

All Implemented Interfaces:

ImageObserver, ItemSelectable, MenuContainer, Serializable, Accessible, MenuElement, SwingConstants

---

```
public class JRadioButtonMenuItem
extends JMenuItem
implements Accessible
```

An implementation of a radio button menu item. A `JRadioButtonMenuItem` is a menu item that is part of a group of menu items in which only one item in the group can be selected. The selected item displays its selected state. Selecting it causes any other selected item to switch to the unselected state. To control the selected state of a group of radio button menu items, use a `ButtonGroup` object.

For further documentation and examples see How to Use Menus, a section in *The Java Tutorial*.

Warning: Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeans™ has been added to the `java.beans` package. Please see `XMLEncoder`.

JRadioButtonMenuItem

```
JMenuBar menuBar = new JMenuBar();

JMenu menu = new JMenu("File");

JMenuItem menuItem;
menuItem = new JMenuItem("A menu item",KeyEvent.VK_T);
menu.add(menuItem);

menu.addSeparator();

ButtonGroup group = new ButtonGroup();

JRadioButtonMenuItem rbMenuItem;
rbMenuItem = new JRadioButtonMenuItem("A radio button menu item");
rbMenuItem.setSelected(true);
group.add(rbMenuItem);
menu.add(rbMenuItem);

rbMenuItem = new JRadioButtonMenuItem("Another radio button");
group.add(rbMenuItem);
menu.add(rbMenuItem);

menuBar.add(menu);
setJMenuBar(menuBar);
```

Barre de séparation

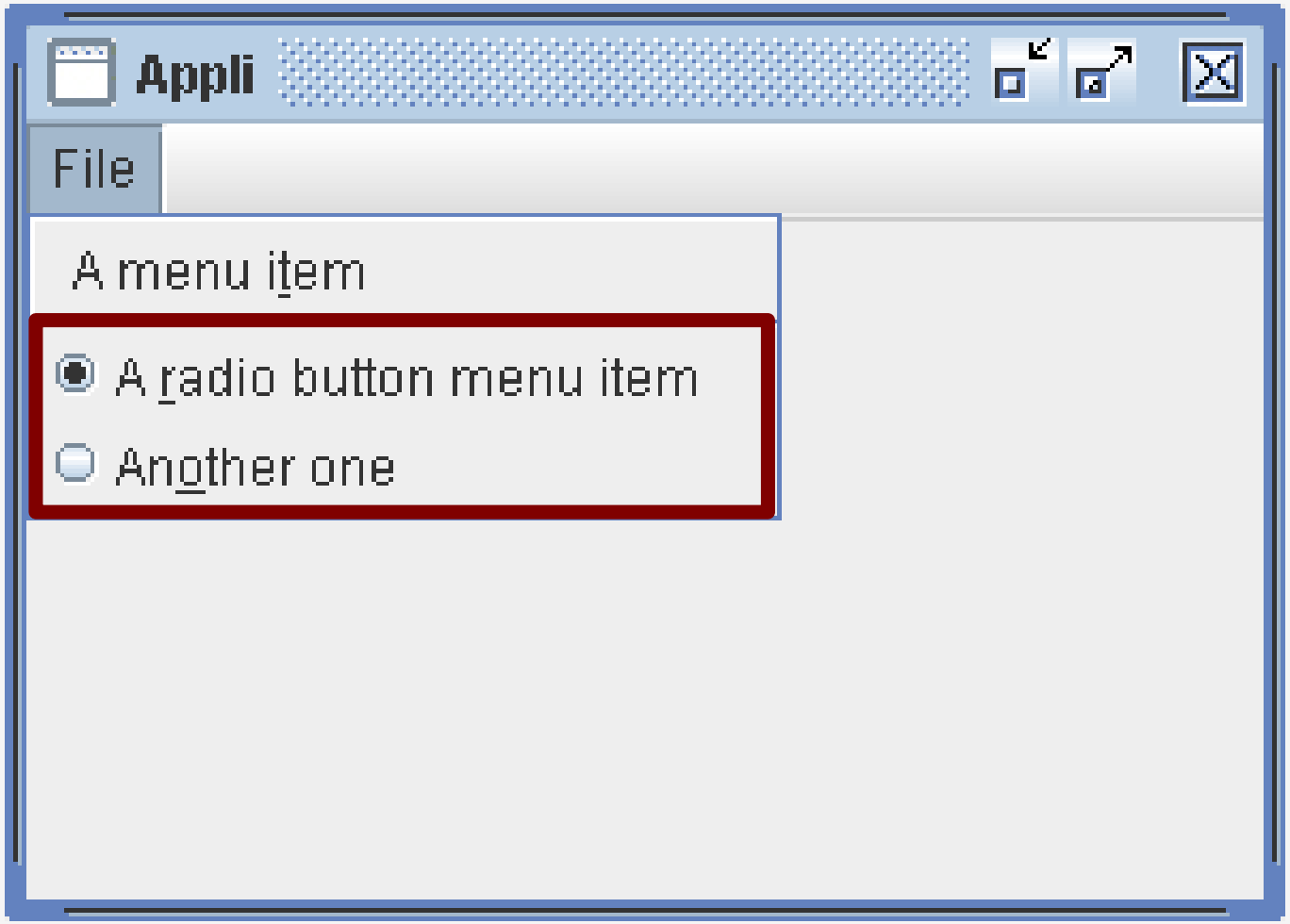
Groupe pour les radio boutons

Un radio bouton

à faux par défaut

Un 2ème radio bouton non sélectionné

JRadioButtonMenuItem



Écouter les JMenuItems

- Les JMenuItems sont des boutons !
- Pour les écouter, on applique donc le même principe que pour les boutons
- Il faut donc créer un ActionListener par JMenuItem

Écouter les JMenuItems

```
JMenuItem menuItem;
menuItem = new JMenuItem("A menu item",KeyEvent.VK_T);
menu.add(menuItem);
menuItem.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        System.out.println("le bouton menu item a été sélectionné");
    }
});
```

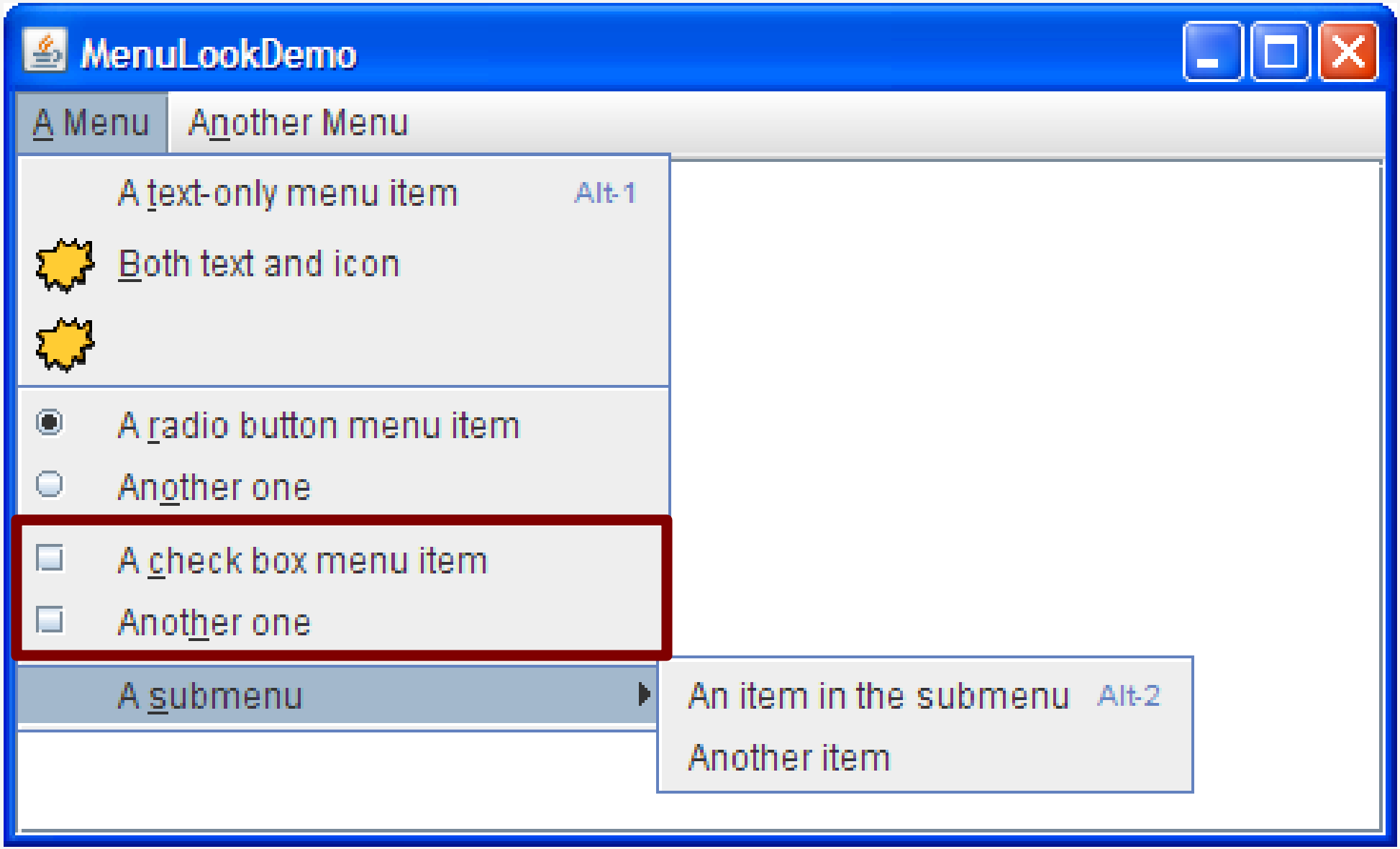
- Pour les autres composants, on utilise simplement l'interface écouteur qui correspond aux composants (cf. doc !)

Écouter un JRadioMenuItem

- Comme les ***MenuItems***, les ***JRadioMenuItems*** sont une sorte de boutton : on applique donc encore une fois le principe d'écoute par ***ActionListener :***

```
jRadioButtonMenuItem1 = new JRadioButtonMenuItem();
jRadioButtonMenuItem1.setText("3");
jRadioButtonMenuItem1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent e) {
        System.out.println("actionPerformed()");
    }
});
```

JCheckBoxMenuItem



JCheckBoxMenuItem

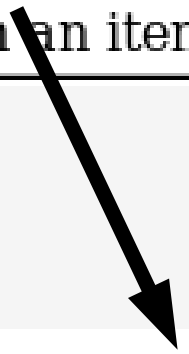
- La création et l'ajout de ce type de ***JMenuItem*** sont classiques
- L'écoute diffère cependant : il faut savoir s'il est sélectionné ou non !
- On utilise donc un ***ItemListener*** qui est ajouté à l'aide la fonction ***addItemListener***
- De plus un ***java.awt.event.ItemListener*** reçoit des ***java.awt.event.ItemEvent***

java.awt.event

Interface ItemListener

Method Summary

void	**itemStateChanged**(ItemEvent e) Invoked when an item has been selected or deselected by the user.



java.awt.event

Class ItemEvent

java.awt.event

Class ItemEvent

Method Summary

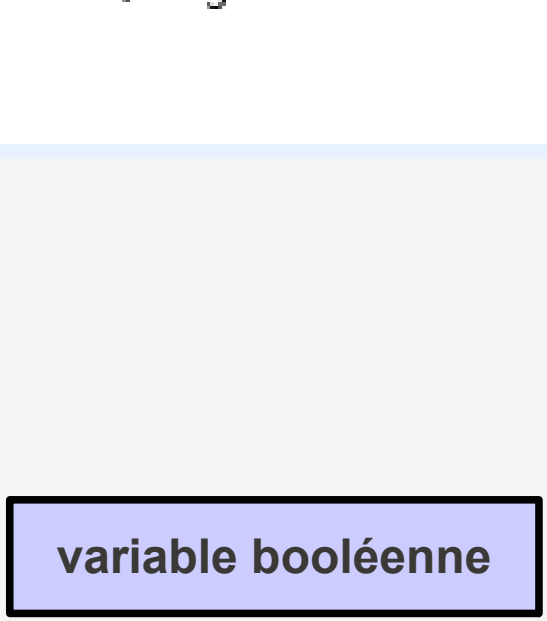
Object	**getItem**() Returns the item affected by the event.
ItemSelectable	**getItemSelectable**() Returns the originator of the event.
int	**getStateChange**() Returns the type of state change (selected or deselected).
String	**paramString**() Returns a parameter string identifying this item event.

Field Summary

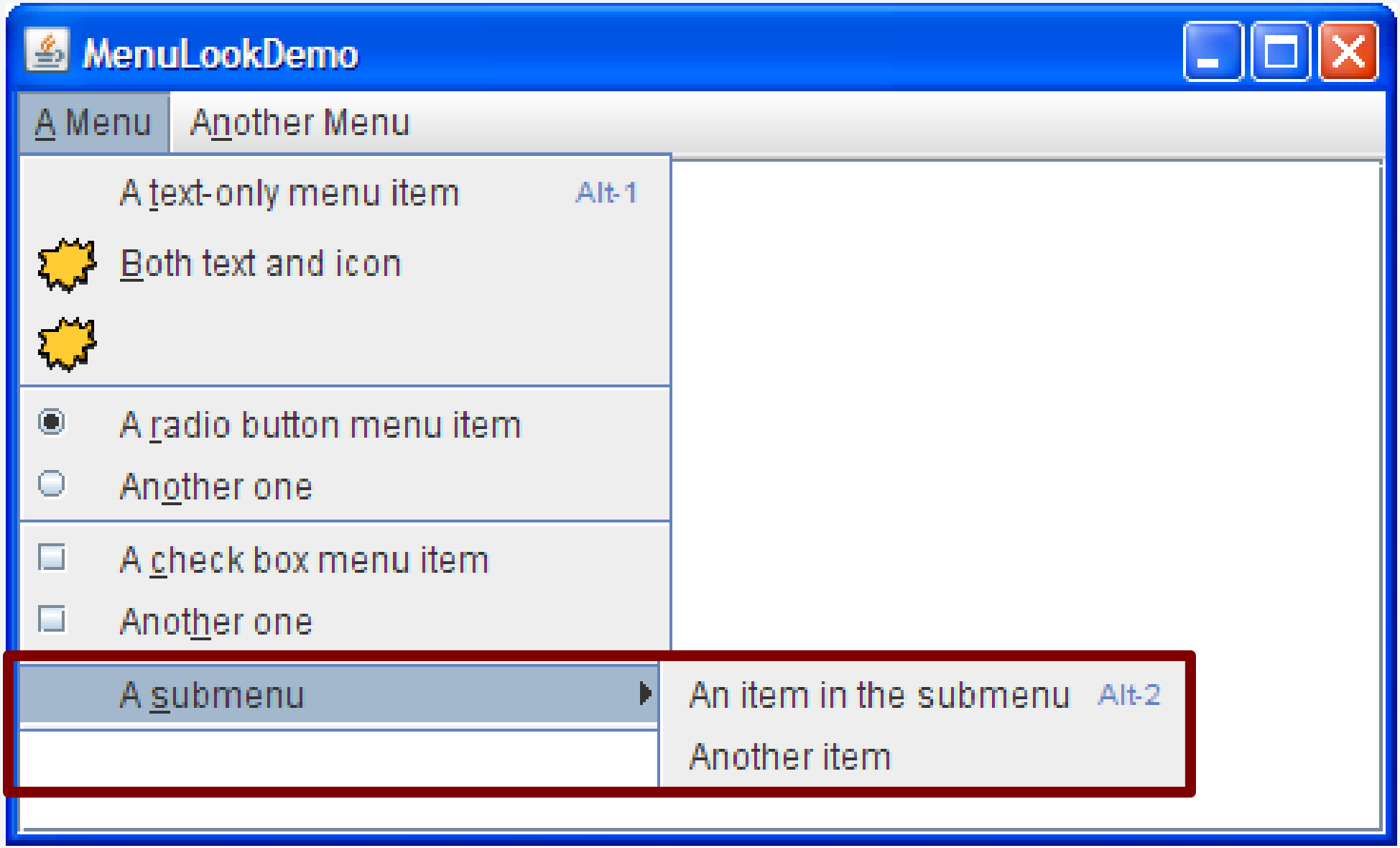
static int	**DESELECTED** This state-change-value indicates that a selected item was deselected.
static int	**ITEM_FIRST** The first number in the range of ids used for item events.
static int	**ITEM_LAST** The last number in the range of ids used for item events.
static int	**ITEM_STATE_CHANGED** This event id indicates that an item's state changed.
static int	**SELECTED** This state-change value indicates that an item was selected.

Ecoute d'un JCheckBoxMenuItem

```
jCheckBoxMenuItem = new JCheckBoxMenuItem();
jCheckBoxMenuItem.setSelected(true);
jCheckBoxMenuItem.setText("Génération automatique des noms");
jCheckBoxMenuItem.addItemListener(new java.awt.event.ItemListener() {
    public void itemStateChanged(java.awt.event.ItemEvent e) {
        noms_auto=(e.getStateChange() == ItemEvent.SELECTED);
    }
});
```



Création de sous menus



Création de sous menus

- Il suffit d'ajouter un menu à un menu !

```
//a submenu
menu.addSeparator();
submenu = new JMenu("A submenu");
submenu.setMnemonic(KeyEvent.VK_S);

menuItem = new JMenuItem("An item in the submenu");
menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_L, ActionEvent.ALT_MASK));
menuItem.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        System.out.println("Sélection du sous menu 1");
    }

});
submenu.add(menuItem);

menuItem = new JMenuItem("Another item");
submenu.add(menuItem);
menu.add(submenu);
```

raccourcis clavier avec combinaison de touches

javax.swing.JPopupMenu

- Menus qui apparaissent en fonction d'une action/condition définie par programme.
- Similaire à un ***JMenuItem*** normal pour le reste.
- Affiché grâce à la méthode suivante :

show

```
public void show(Component invoker,
                 int x,
                 int y)
```

Displays the popup menu at the position x,y in the coordinate space of the component invoker.

Parameters:

`invoker` - the component in whose space the popup menu is to appear
`x` - the x coordinate in invoker's coordinate space at which the popup menu is to be displayed
`y` - the y coordinate in invoker's coordinate space at which the popup menu is to be displayed

exemple

```
final JPopupMenu popup = new JPopupMenu();
menuItem = new JMenuItem("A popup menu item");
popup.add(menuItem);
menuItem = new JMenuItem("Another popup menu item");
popup.add(menuItem);

//a submenu
menu.addSeparator();
submenu = new JMenu("A submenu");
submenu.setMnemonic(KeyEvent.VK_S);

menuItem = new JMenuItem("An item in the submenu");
menuItem.addActionListener(new ActionListener(){
    public void actionPerformed(ActionEvent e){
        popup.show(getContentPane(),10,10);
    }

});
submenu.add(menuItem);
```

création comme pour un menu normal

invocation sur clique d'un JMenuItem

Ajouter des bulles d'aide

- Héritée de la classe ***javax.swing.JComponent :***

setToolTipText

```
public void setToolTipText(String text)
```

Registers the text to display in a tool tip. The text displays when the cursor lingers over the component.

See How to Use Tool Tips in *The Java Tutorial* for further documentation.

Parameters:

`text` - the string to display; if the text is `null`, the tool tip is turned off for this component

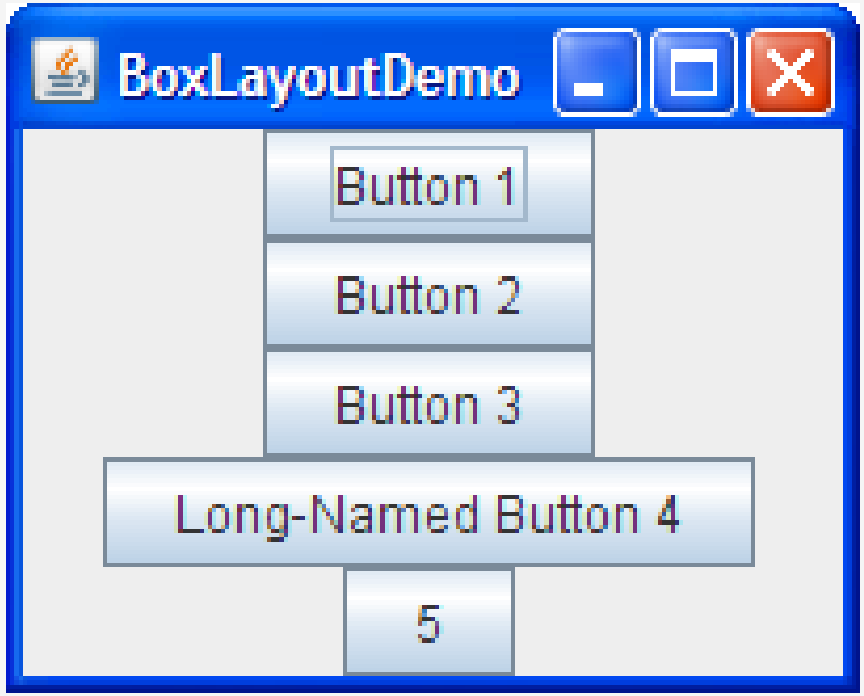
See Also:

`TOOL_TIP_TEXT_KEY`

```
menuItem.setToolTipText("This doesn't really do anything");
```

Mise en forme des menus

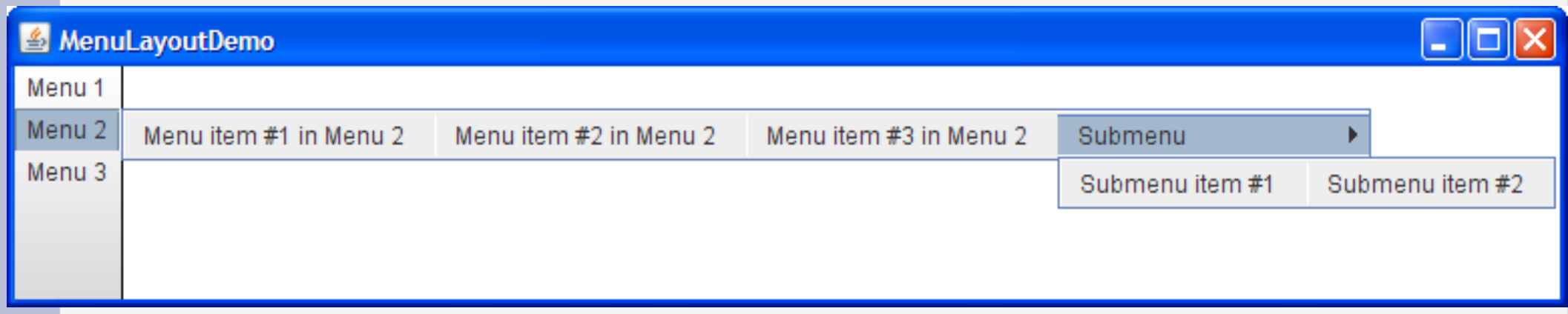
- Les menus sont gérés par défaut par un gestionnaire de type ***BoxLayout.*** (Un FlowLayout qui conserve l'alignement: vertical ou horizontal)



exemple

```
public JMenuBar createMenuBar() {
    JMenuBar menuBar = new JMenuBar();
    menuBar.setLayout(new BoxLayout(menuBar, BoxLayout.PAGE_AXIS));
    menuBar.add(createMenu("Menu 1"));
    menuBar.add(createMenu("Menu 2"));
    menuBar.add(createMenu("Menu 3"));

    menuBar.setBorder(BorderFactory.createMatteBorder(0,0,0,1,
                                                      Color.BLACK));
    return menuBar;
}
```

```
class HorizontalMenu extends JMenu {
    HorizontalMenu(String label) {
        super(label);
        JPopupMenu pm = getPopupMenu();
        pm.setLayout(new BoxLayout(pm, BoxLayout.LINE_AXIS));
    }
```



javax.swing.Box

- ***javax.swing.Box*** :classe composée de méthode statiques destinées à créer des composants particuliers (invisibles) qui permettent de combler un espace par exemple.
- menuBar.add(Box.createHorizontalGlue());
- //...
- menuBar.add(rightMenu);