Liste et ComboBox

- JList et ListModel
- JComboBox et ComboBoxModel

Les Listes

- Composant affichant une séquence d'entrées.
- Utilise deux modèles différents :
  - **`ListModel`** pour les données,
    évènement : **`ListDataEvent`**
  - **`ListSelectionModel`** pour les données sélectionnées,
    évènement : **`ListSelectionEvent`**.
- La classe **`DefaultListSelectionModel`** correspond à l'implantation par défaut du modèle de sélection.
- L'interface **`ListCellRenderer`** permet d'indiquer le composant de rendu.

Les Listes (2)

- Constructeurs :

```
JList()                          // modèle vide
JList(ListModel dataModel)       // cas le plus général
JList(Object[] listData)         // tableau
JList(Vector listData)           // vecteur
```

- Création d'une liste à partir d'un tableau (liste non modifiable) :

```
String[] data = {"one", "two", "free", "four"};
JList liste = new JList(data);
```

- Accès au modèle, donc au contenu:

```
for(int i = 0; i < jList.getModel().getSize(); i++) {
  System.out.println(jList.getModel().getElementAt(i));
}
```

Evènement de données

- Un **`ListDataEvent`** contient notamment l'intervalle d'entrées consécutives concernées, spécifié par deux entiers inclus dans la sélection.

- L'interface **`ListDataListener`** définie 3 méthodes à implémenter
  - void **`contentsChanged`**(ListDataEvent e) indique que le contenue des données de l'intervalle a changé.
  - void **`intervalAdded`**(ListDataEvent e) indique l'insertion d'un intervalle.
  - void **`intervalRemoved`**(ListDataEvent e) indique la suppression d'un intervalle.

Le modèle de sélection

- Un **`ListSelectionModel`**est assez complexe.
- Il possède trois modes de sélection :
  - SINGLE_SELECTION, une seule donnée à la fois.
  - SINGLE_INTERVAL_SELECTION, un seul intervalle.
  - MULTIPLE_INTERVAL_SELECTION, plusieurs intervalles.
- Pour simplifier l'utilisation, quelques méthodes du modèle de sélection son présente sur **`JList`**.
  - int get/setSelectedIndex()
  - int[] get/setSelectedIndicies()
  - Object get/setSelectedValue()
  - Object[] get/setSelectedValues()

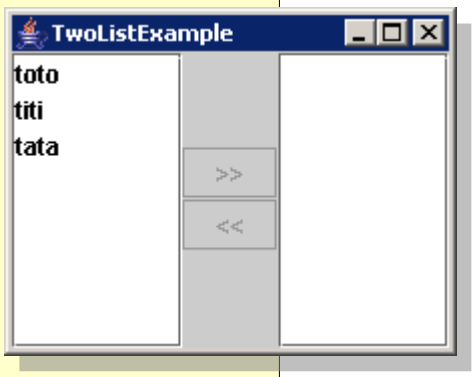
Evènement de sélection

- Un **`ListSelectionEvent`** contient notamment les indices de l'intervalle où une modification de sélection a eu lieu.

- L'interface **`ListSelectionListener`** possède une seule méthode à implémenter
  - void **`valueChanged`**(ListSelectionEvent e) exécutée quand la sélection a changée.
  - très bavard : quand on enfonce ***et*** quand on relâche la souris.

Exemple de sélection

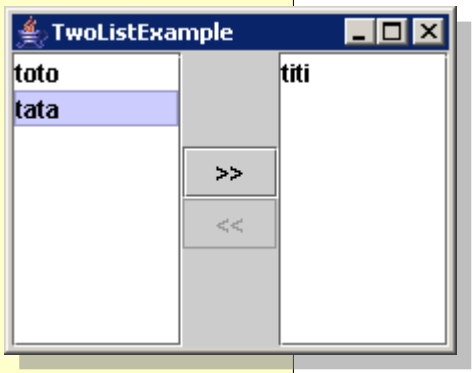
```
private static JButton createMoveButton(String text,
  final JList from,final JList to) {
  final JButton button=new JButton(text);
  button.setEnabled(false);
  button.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
      int index=from.getSelectedIndex();
      Object value=from.getSelectedValue();
      DefaultListModel fromModel=
        (DefaultListModel)from.getModel();
      fromModel.remove(index);
      DefaultListModel toModel=
        (DefaultListModel)to.getModel();
      toModel.addElement(value);
    }
  });
  from.getSelectionModel().addListSelectionListener(
    new ListSelectionListener() {
    public void valueChanged(ListSelectionEvent e) {
      button.setEnabled(from.getSelectedIndex()!=-1);
    }
  });
  return button;
}
```



Exemple de sélection -2

- On passe la liste en mode SINGLE_SELECTION.

```
private static JList createList() {
  DefaultListModel listModel=new DefaultListModel();
  JList list=new JList(listModel);
  list.setPrototypeCellValue("AAAAAAAAAA");
  list.setVisibleRowCount(8);
  list.getSelectionModel().
    setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
  return list;
}
public static void main(String[] args) {
  JList list1=createList();
  DefaultListModel listModel=
   (DefaultListModel)list1.getModel();
  for(String arg:args)
    listModel.addElement(arg);
  JList list2=createList();
  JButton button1=createMoveButton(">>",list1, list2);
  JButton button2=createMoveButton("<<",list2, list1);
  ...
```



Les Boites déroulante

- La classe **`JComboBox`** permet la sélection d'une entrée parmis une séquence.
- Le composant est constitué de deux parties :
  - un editeur qui affiche la sélection courrante et permet ou non d'entrée des valeurs et
  - une liste déroulante qui affiche les choix possibles.
- Deux modes de fonctionnement différents :
  - Non éditable : **`ComboBoxModel`**
  - Editable : **`MutableComboBoxModel`**
- Deux interfaces de modèle de donnée.
- Changement de mode (**`set/getEditable()`**).

Modèle de boite déroulante

- **`ComboBoxModel`** étend **`ListModel`**:
  - **`Object getSelectedItem()`**
  - **`void setSelectedItem(Object anItem)`**

- **`MutableComboBoxModel`** étend **`ComboBoxModel`**:
  - **`void addElement(Object obj)`**
  - **`void insertElementAt(Object obj, int index)`**
  - **`void removeElement(Object obj)`**
  - **`void removeElementAt(int index)`**

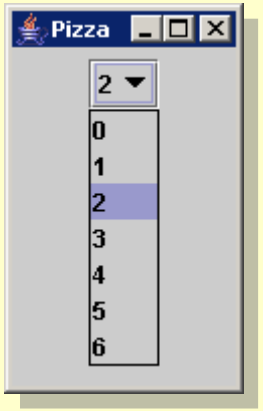
- Comme les deux modèles héritent de **`ListModel`**, la gestion des évènements de modification de données est la même que pour le **`ListModel`**.

Exemple de modèle - non mutable

- Un `ComboBoxModel` est un `ListModel` + une sélection

```
class PizzaModel extends AbstractListModel implements ComboBoxModel {
  public int getSize() {
    return icons.length;
  }
  public Integer getElementAt(int index) {
    return index;
  }
  public Integer getSelectedItem() {
    return item;
  }
  public void setSelectedItem(Object anItem) {
    item=(Integer)anItem;
  }
  private Integer item;
}

public JComboBox createCombo(ComboBoxModel model,
  ListCellRenderer renderer) {
  JComboBox combo=new JComboBox(model);
  combo.setRenderer(renderer);
  return combo;
}
```



Renderer et Editor

- Rendu graphique d'une boite déroulante :
  - un renderer (**`ListCellRenderer`**)
  - Un editor (**`ComboBoxEditor`**) si le composant est editable.

- Le renderer est le même que pour les listes mais le même composant de rendu est utilisé pour la liste déroulante et pour la partie sélectionné.
- Dans le cas de la partie sélectionnée, la valeur (**`value`**)est la l'objet sélectionné, l'index (**`index`**) vaux -1.
- La méthode **`setCellRenderer(ListCellRenderer)`** indique le composant de rendu.
- La méthode **`setComboBoxEditor(ComboBoxEditor)`** indique le composant d'édition.

Exemple de Renderer

- La méthode de rendu doit faire attention à la valeur sélectionnée.

```
private ListCellRenderer createRenderer() {
  return new DefaultListCellRenderer() {
    public Component getListCellRendererComponent(
      JList list, Object value, int index,
      boolean isSelected, boolean cellHasFocus) {

      super.getListCellRendererComponent(list,value,index,
        isSelected,cellHasFocus);

      // s'il n'y a pas de valeur sélectionnée
      if (value!=null) {
        index=(Integer)value; // unboxing
      }
      setIcon(icons[index]);
      setText("Pizza "+(index+1));

      return this;
    }
  };
}
```

Gestion des évènements

- La boîte déroulante gère deux types d'évènements :
- **`ActionListener`** pour obtenir l'objet sélectionné.
- **`ItemListener`** pour être appelé à chaque nouvelle selection dans la liste.
- L'**`ItemListener`** sert très rarement.