

COURS – TD 7: TABLES

Objectifs d'un élément graphique de type Table

- Présenter une vision structurée d'un grand nombre de données et permettre leur manipulation
 - 2 dimensions
 - Synthèse
 - Classification

JTable

- Afficher un tableau
- Permettre l'édition de son contenu

The Header contains
Column labels

First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	<input type="checkbox"/>
John	Doe	Rowing	3	<input checked="" type="checkbox"/>
Sue	Black	Knitting	2	<input type="checkbox"/>
Jane	White	Speed reading	20	<input checked="" type="checkbox"/>
...

Each Cell displays
a data item

Each Column displays
one type of data

- Lignes
- Colonnes – En-têtes
- Cellules

Layout

- Souvent utile de placer une JTable dans un JScrollPane
 - Nombre important d'entrées
 - Taille de la zone d'affichage restreinte

Gestion d'une JTable

- Model Données à afficher
- Renderer Rendu visuel des cellules
- Editor Edition des données contenues dans les cellules

Données

- JTable doit implémenter l'interface TableModel
- Données transmises directement au constructeur
- Données transmises dans un modèle
 - Modèle simple: classe **DefaultTableModel**
 - Personnalisation du modèle: classe abstraite AbstractTableModel

Données transmises au constructeur

- JTable(Object[][] rowData, Object[] columnNames)
- JTable(Vector rowData, Vector columnNames)
rowData est un Vector de Vectors

Example

```
JTable(Object[][] rowData, Object[] columnNames)
```

```
String[] columnNames =
```

```
    {"First Name", "Last Name", "Sport", "# of Years",  
    "Vegetarian"};
```

```
Object[][] data = {
```

```
    {"Kathy", "Smith", "Snowboarding", new Integer(5), new  
    Boolean(false)},
```

```
    {"John", "Doe", "Rowing", new Integer(3), new  
    Boolean(true)},
```

```
    {"Sue", "Black", "Knitting", new Integer(2), new  
    Boolean(false)},
```

```
    {"Jane", "White", "Speed reading", new Integer(20), new  
    Boolean(true)},
```

```
    {"Joe", "Brown", "Pool", new Integer(10), new  
    Boolean(false)} };
```

```
JTable table = new JTable(data, columnNames);
```


Modèle simple - DefaultTableModel

- Instance créée par défaut à la création d'une Jtable

Exemple:

```
JTable myTable = new JTable ();
```

ou le constructeur

```
JTable(int numRows, int numColumns)
```

- Fournit des méthodes d'accès à des données enregistrées sous forme de vecteurs de vecteurs
- Toutes les cellules sont éditables par défaut
- Hérite de la classe AbstractTableModel

Personnalisation du modèle

- Définir une classe héritant de `AbstractTableModel`
- Exige de définir les méthodes d'accès aux données (au moins)
 - `public int getRowCount()`
 - `public int getColumnCount()`
 - `public Object getValueAt(int row, int column);`
- Edition des données
 - `void setValueAt(Object aValue, int rowIndex, int columnIndex)`

Surcharges intéressantes (1/2)

- Indiquer si la cellule est éditable

boolean isCellEditable(int rowIndex, int colIndex)

- Indiquer le nom des colonnes

String getColumnName(int colIndex)

- Indiquer la classe (le type) des colonnes

Class getColumnClass(int colIndex)

Surcharges intéressantes (2/2)

- Modifier une valeur dans le modèle

void setValueAt(Object val, int rowIndex, int colIndex)

Nécessite de « prévenir » des changements dans le modèle

fireTableCellUpdated: une cellule

fireTableRowsUpdated: une ligne

fireTableDataChanged: la table entière

fireTableRowsInserted: insertion de ligne

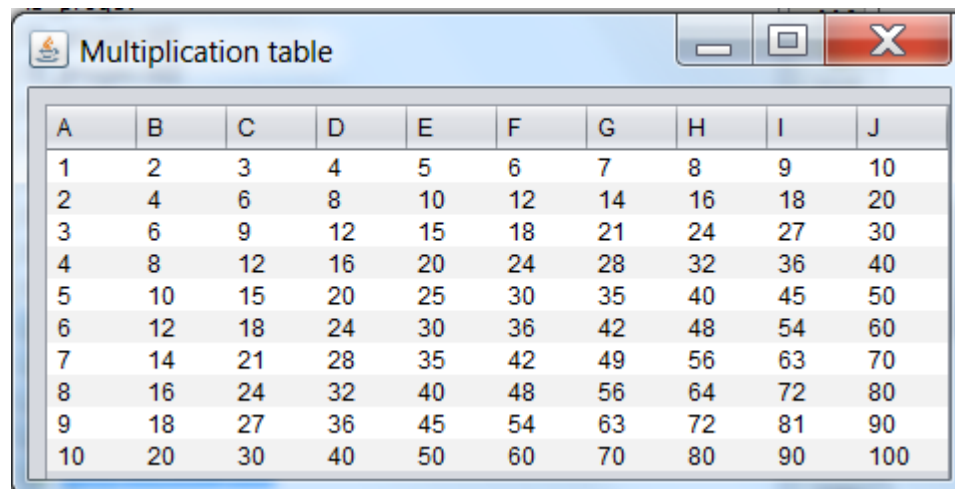
fireTableRowsDeleted: suppression de ligne

fireTableStructureChanged : structure de la table

Exercice 1

Table de multiplication

- Définir une classe de modèle de données pour une Jtable
 - Contenant une table de multiplication de 1 à 10
 - Les données ne sont pas éditables



A screenshot of a Java Swing window titled "Multiplication table". The window contains a 10x10 table with columns labeled A through J and rows labeled 1 through 10. The table displays the multiplication results for numbers 1 to 10. The table is read-only, as indicated by the lack of an edit cursor.

	A	B	C	D	E	F	G	H	I	J
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

Exercice 2

Répertoire

- Définir une classe de modèle de données pour la JTable suivante
 - Seule la colonne Adresse est éditable

Nom	Prénom	Adresse
Doe	John	24, rue du Chemin
Smith	Mary	2, rue du Faubourg
English	Johnny	5, rue du Pré

Rendu - Renderer

- Rendu visuel des cellules de la table
- Méthode à définir pour permettre de différencier les rendus

```
public Class getColumnClass(int c) {  
    return getValueAt(0, c).getClass();  
}
```

- Exemple: Type booléen dans la dernière colonne

Nom	Prénom	Adresse	Carte de fidélité
Doe	John	24, rue du Chemin	<input checked="" type="checkbox"/>
Smith	Mary	2, rue du Faubourg	<input type="checkbox"/>
English	Johny	5, rue du Pré	<input checked="" type="checkbox"/>

Rendus par défaut

classe `DefaultTableCellRenderer`

- Boolean — case à cocher
- Number — label aligné à droite
- Double, Float — label aligné à droite, conversion effectuée avec la classe [NumberFormat](#)
- Date — label, conversion avec la classe [DateFormat](#)
- ImageIcon, Icon — label centré
- Object — chaîne de caractère correspondante à l'objet

Personnalisation du rendu des cellules (1/2)

- Définir une classe qui met en oeuvre l'interface **TableCellRenderer** et étend un élément graphique (JComponent)

- Associer le nouveau renderer aux cellules des colonnes souhaitées:

- Rendu des colonnes contenant type de classe

JTable: public void setDefaultRenderer(Class<?> columnClass,
TableCellRenderer renderer)

- Rendu d'une colonne

Jtable:

public TableColumn getColumn(Object identifier)

TableColumn:

void setCellRenderer(TableCellRenderer cellRenderer)

Personnalisation du rendu des cellules (2/2)

- Implémenter la méthode

```
public Component getTableCellRendererComponent  
    (JTable table, Object value, boolean isSelected,  
    boolean hasFocus, int row, int column)
```

Exercice 3

Répertoire (rendu)

- Ecrire la portion de code permettant de personnaliser le rendu de la première colonne de la table de la manière suivante:

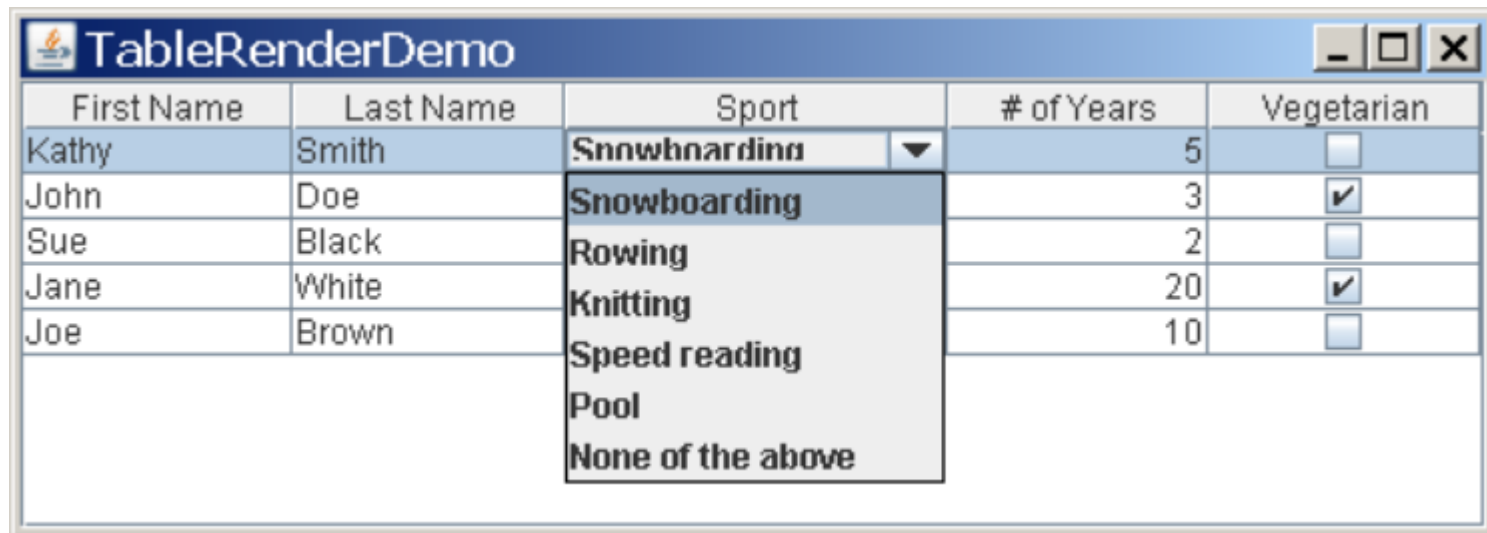
Nom	Prénom	Adresse	Carte de fidélité
Doe	John	24, rue du Chemin	<input checked="" type="checkbox"/>
Smith	Mary	2, rue du Faubourg	<input type="checkbox"/>
English	Johnny	5, rue du Pré	<input checked="" type="checkbox"/>

Editor - Editeur

- Différentes manières de saisir une donnée
 - Champ texte
 - Case à cocher
 - Liste déroulante
 - Color picker
 - ...

Editeur par défaut

- DefaultCellEditor
 - JTextField, JCheckBox, JComboBox



The screenshot shows a Java Swing window titled "TableRenderDemo". Inside the window is a table with five columns: "First Name", "Last Name", "Sport", "# of Years", and "Vegetarian". The table contains five rows of data. The "Sport" column has a dropdown menu open, showing a list of options: "Snowboarding", "Rowing", "Knitting", "Speed reading", "Pool", and "None of the above". The "Snowboarding" option is currently selected and highlighted. The "Vegetarian" column has checkboxes, with the second and fourth rows checked.

First Name	Last Name	Sport	# of Years	Vegetarian
Kathy	Smith	Snowboarding	5	<input type="checkbox"/>
John	Doe	Snowboarding	3	<input checked="" type="checkbox"/>
Sue	Black	Rowing	2	<input type="checkbox"/>
Jane	White	Knitting	20	<input checked="" type="checkbox"/>
Joe	Brown	Speed reading	10	<input type="checkbox"/>

Edition par liste déroulante - exemple

```
TableColumn sportColumn =  
table.getColumnModel().getColumn(2);
```

...

```
JComboBox comboBox = new JComboBox();  
comboBox.addItem("Snowboarding");  
comboBox.addItem("Rowing");  
comboBox.addItem("Chasing toddlers");  
comboBox.addItem("Speed reading");  
comboBox.addItem("Teaching high school");  
comboBox.addItem("None");  
sportColumn.setCellEditor(new DefaultCellEditor(comboBox));
```

Personnalisation de l'édition des cellules

- Définir une classe qui met en oeuvre l'interface **TableCellEditor** et étend la classe **AbstractCellEditor**

- Configurer l'éditeur

JTable:

```
void setDefaultEditor(Class<?> columnClass,  
    TableCellEditor editor)
```

TableColumn:

```
void setCellEditor(TableCellEditor cellEditor)
```

- Exemple (Color picker)

<http://docs.oracle.com/javase/tutorial/uiswing/components/table.html#editor>

Récap. model/rendu/edition

	Classe utilisée par défaut	Personnalisation
Modèle de données	Classe concrète DefaultTableModel	Extension de la classe abstraite AbstractTableModel
Rendu	Classe concrète DefaultCellRenderer	Extension d'un élément graphique (héritant de Jcomponent) + mise en oeuvre de l'interface TableCellRenderer
Edition	Classe concrète DefaultCellEditor	Extension de la classe abstraite AbstractCellEditor + mise en oeuvre de l'interface TableCellEditor

Remarque

- Les concepts associés au modèle de données sont applicables à d'autres éléments SWING
- Les concepts associés au rendu visuel sont applicables à d'autres éléments SWING
- Notion de Modèle-Vue-Contrôleur
(Design patterns, M1 IHM)
- Exemples: JList, JComboBox

Evènements

- Modification du modèle
- Edition d'une cellule
- Sélection de lignes/colonnes

Evènements: modification du modèle

- **TableModelListener**

`void tableChanged(TableModelEvent e)`

- **TableModelEvent**

`int getColumn()`

Colonne impactée par l'évènement.

`int getFirstRow()`

Première ligne ayant changé.

`int getLastRow()`

Dernière ligne ayant changé.

`int getType()`

Type de l'évènement: INSERT, UPDATE and DELETE.

Evènements: Edition d'une cellule

- **CellEditorListener**

`void editingCanceled(ChangeEvent e)`

Edition annulée

`void editingStopped(ChangeEvent e)`

Edition terminée

Evènements: sélection d'une ligne/colonne

- Evènement de sélection différent pour les lignes et pour les colonnes.

- Sélection d'une ligne (sur la JTable) :

Jtable: ListSelectionModel **getListSelectionModel()**

ListSelectionModel:

addListSelectionListener(ListSelectionListener l)

- Sélection d'une colonne (sur un TableColumn) :

Jtable: ListSelectionModel

getColumnModel().getSelectionModel()

ListSelectionModel:

addListSelectionListener(ListSelectionListener l)

Exercice 4

Répertoire (sélection)

- Ecrire la portion de code permettant d'associer un listener au modèle de sélection de la table, ainsi que l'événement handler déclenchant l'affichage suivant:

