



Gui



# Eddig...

- JTextField
- JTextArea
- JList
- JButton
- JCheckBox
- JRadioButton
- JMenu
- JToggleButton



# Egyéb szükséges komponensek

- JTable
- JComboBox
- JScrollPane
- JSlider
- JPasswordField

# JTable

- Adatok táblázatos formában való megjelenítésére alkalmas.
- Opcionálisan editálható.
- A JTable nem tartalmazza a megjelenített adatokat, az adatoknak csak egy nézete.

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

# JTable Egyszerű példa

- ▶ Táblázat létrehozása az adatok és oszlopnevek megadásával:

```
String[] columnNames = {"column1", "column2",...};  
  
Object[][] data = {  
    {value1,value2,...},  
    {value1,value2,...}  
};  
  
JTable table = new JTable(data, columnNames);
```

- ▶ Hátrányai:
  - ▶ A táblázat minden cellája editálható.
  - ▶ Minden adattípus string-ként kezelt.
  - ▶ A tömböt össze kell állítani...

```
JScrollPane scrollPane = new JScrollPane(table);
```

```
table.setFillViewportHeight(true);
```

- JScrollPane létrehozása a táblázat konténerként, a táblázat automatikusan hozzáadásra kerül.
- setViewportHeight: ha igaz, a táblázat a konténer teljes magasságát felhasználja, akkor is, ha táblának nincs elegendő sora.
- A scroll pane a táblázat header-jét automatikusan a viewport tetejére helyezi, az oszlopnevek scrollozás közben is láthatóak maradnak.

### **Oszlopok szélessége**

- Alapértelmezetten minden oszlop egyforma széles, a táblázat teljes szélességét kitöltik.
- Egy oszlop szélességének megváltoztatása:

```
column = table.getColumnModel().getColumn(0);
```

```
column.setPreferredWidth(100);
```

# Kijelölések – Selection Mode

- ▶ Alapértelmezésként a táblázat minden sora kiválasztható.
- ▶ A `JTable.setSelectionMode` metódussal változható meg a táblázatban engedélyezett kijelölés módja.
- ▶ Ennek értéke a `javax.swing.ListSelectionModel` osztály konstansai lehetnek. (`MULTIPLE_INTERVAL_SELECTION`, `SINGLE_INTERVAL_SELECTION`, és `SINGLE_SELECTION`)

# Kijelölések – Selection Option

- ▶ `rowSelectionAllowed`: ha igaz (és a `columnSelectionAllowed` hamis) akkor a sorok kijelölhetőek.
- ▶ `columnSelectionAllowed` : ha igaz (és a `rowSelectionAllowed` hamis) akkor az oszlopok kijelölhetőek.
- ▶ `cellSelectionEnabled`: ha igaz, cellák kijelölhetőek.
- ▶ Kijelölés lekérdezése: `JTable.getSelectedRows` és `JTable.getSelectedColumns`. A kiválasztott indexek tömbjét adják meg.





# Table Model

- ▶ Minden táblázathoz tartozik, egy a tényleges adatokat tartalmazó `TableModel` objektum.
- ▶ Ennek az objektumnak implementálnia kell a `TableModel` interfészt.
- ▶ Ha nincs megadva a `JTable` automatikusan készít egy `DefaultTableModel` példányt az adatok tárolására.
- ▶ A table model legegyszerűbben az `AbstractTableModel` leszármazottjaként implementálható.



# TableModel részei

- ▶ `int getRowCount()`: Megadja a táblázat sorainak számát.
- ▶ `int getColumnCount()`: oszlopok száma.
- ▶ `getColumnName(int col)`: adott indexű oszlop neve.
- ▶ `Class getColumnClass(int col)`: adott indexű oszlop típusa.
- ▶ `boolean isCellEditable(int row, int col)`: editálható e, az indexekkel adott cella.
- ▶ `Object getValueAt(int row, int col)`: indexekkel adott cella értéke.
- ▶ `setValueAt(Object val, int row, int col)`: cella értékének módosítása

# Változások kezelése

- ▶ A model-ben történt adatváltozásokról a JTable-t értesíteni kell. Az `abstractTableModel` megfelelő metódusának hívásával.

## Method

`fireTableCellUpdated`

`fireTableRowsUpdated`

`fireTableDataChanged`

`fireTableRowsInserted`

`fireTableRowsDeleted`

`fireTableStructureChanged`

## Change

Update of specified cell.

Update of specified rows

Update of entire table (data only).

New rows inserted.

Existing rows Deleted

Invalidate entire table, both data and structure



# Renderers

- ▶ Az azonos típusú adatok megjelenítéshez ugyanaz a cell render komponens lesz felhasználva.
- ▶ Ha nincs explicit megadott renderer, a táblázat a getColumnClass alapján választ egy alapértelmezettet.
  - ▶ Boolean — rendered with a check box.
  - ▶ Number — rendered by a right-aligned label.
  - ▶ Date — rendered by a label
  - ▶ ImageIcon, Icon — rendered by a centered label.
  - ▶ Object — rendered by a label that displays the object's string value.

# Rendezés, szűrés

- ▶ A rendezhetőség megvalósításának legegyszerűbb módja a táblázat `autoCreateRowSorter` tulajdonságának `true`-ra állítása.
- ▶ Alternatívaként készíthető saját rendező objektum is:

```
TableRowSorter<TableModel> sorter  
    = new TableRowSorter<TableModel>(table.getModel());  
table.setRowSorter(sorter);
```

- ▶ A `TableRowSorter` egy `Comparator` objektumot használ a sorok rendezéséhez. `Comparator` kiválasztása egy oszlopra (az első megfelelő):
  1. `Comparator` meg van adva a `setComparator` metódussal.
  2. Ha az oszlop típusa `String`, `string comparator`.
  3. Ha a `getColumnClass` egy `Comparable` osztállyal tér vissza, a `comparator` annak a `compareTo` metódusát használja.
  4. Megadott `StringConverter` esetén a `comparator` a `string` reprezentációkkal.
  5. Minden más esetben a az oszlop értékeinek `toString` eredményeit használó `Comparator`.



# Rendezés sorrendje

- ▶ A rendezés sorrendjét és irányát a `setSortKeys` metódussal adhatjuk meg.

```
List <RowSorter.SortKey> sortKeys  
    = new ArrayList<RowSorter.SortKey>();  
sortKeys.add(new RowSorter.SortKey(1, SortOrder.ASCENDING));  
sortKeys.add(new RowSorter.SortKey(0, SortOrder.ASCENDING));  
sorter.setSortKeys(sortKeys);
```

# Sorok szűrése

- ▶ A sorok rendezése mellett sorter-el adható meg, mely sorok jelenjenek meg a táblázatban.
- ▶ A TableRowSorter a szűrést a javax.swing.RowFilter segítségével implementálja.

```
RowFilter<MyTableModel, Object> rf = RowFilter.regexFilter("regexp", 0);  
sorter.setRowFilter(rf);
```

- ▶ Szűrések és rendezések használatakor az adatok más sorrendben szerepelhetnek a megjelenített táblázatban, mint a modellben.
- ▶ Emiatt az indexeket konvertálni kell a megjelenítés és a table model között:
- ▶ JTable biztosít konvertáló metódusokat: `convertRowIndexToModel`, `convertColumnIndexToView`, stb....

# JComboBox

- ▶ A componens lehetőséget biztosít arra, hogy kiválasszunk egy elemet több lehetőség közül egy lenyíló lista segítségével.

```
String[] petStrings = { "Bird", "Cat", "Dog", "Rabbit", "Pig" };  
//Create the combo box, select item at index 4.  
//Indices start at 0, so 4 specifies the pig.  
JComboBox petList = new JComboBox(petStrings);  
petList.setSelectedIndex(4);  
petList.addActionListener(this)
```





# JSlider

- ▶ A JSlider komponens célja numerikus adatok megadása egy minimum és egy maximum érték között.

```
JSlider slider = new JSlider(JSlider.HORIZONTAL, MIN, MAX, INIT);  
slider.addChangeListener(this);  
slider.setMajorTickSpacing(10);  
slider.setMinorTickSpacing(1);  
slider.setPaintTicks(true);  
slider.setPaintLabels(true);
```

- ▶ A slider mutatójának mozgatása esetén a `ChangeListener StateChanged` metódusa hívódik meg.



# JSlider címkek módosítása

```
Hashtable labelTable = new Hashtable();  
labelTable.put( new Integer( 0 ), new JLabel("Stop") );  
labelTable.put( new Integer( FPS_MAX/10 ), new JLabel("Slow") );  
labelTable.put( new Integer( FPS_MAX ), new JLabel("Fast") );  
slider.setLabelTable( labelTable );  
slider.setPaintLabels(true);
```

# JPasswordField

- ▶ A JTextField komponens leszármazottja, jelszavak megadásához szükséges speciális beviteli mező.
- ▶ Biztonsági megfontásokból az értékét karakter tömbben tárolja String helyett.
- ▶ A komponens aléprtelmezsként egy „pont”-ot ír minden karakter helyére, megváltoztatása: `setEchoChar` metódussal.
- ▶ A begépet jelszó a `getPassword` metódussal érhető el. Ha az érték már nem szükséges, a visszakapott tömböt ki kell törölni.

```
char[] input = passwordField.getPassword();
```

```
...
```

```
Arrays.fill(input, '0');
```