Gui

Eddig...

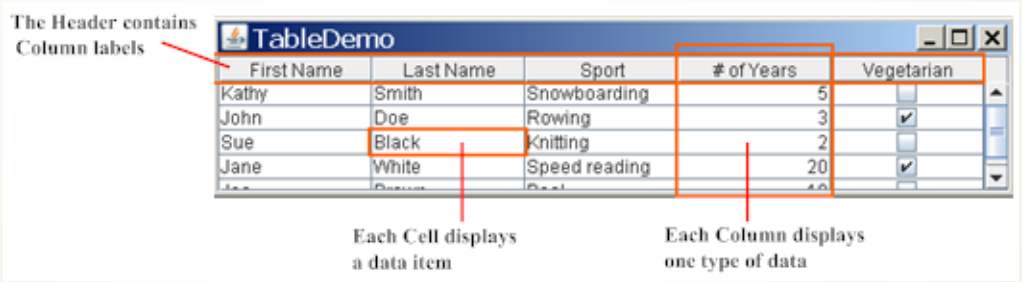
- JTextField
- JTextArea
- JList
- Jbutton
- JCheckBox
- JRadioButton
- JMenu
- JToggleButton

Egyéb szükséges komponensek

- JTable
- JComboBox
- JScrollPanel
- JSlider
- JPasswordField

JTable

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



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.setFillsViewportHeight(true);
```

- ScrollPane létrehozása a táblázat konténereként, a táblázat autómatikusan 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 autómatikusan a viewport tetejére helyezi, az oszlopnevek scrollozás közben is láthatóak maradnak.

Oszlopok szélsessége

- 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` autómatikusan 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	Change
`fireTableCellUpdated`	Update of specified cell.
`fireTableRowsUpdated`	Update of specified rows
`fireTableDataChanged`	Update of entire table (data only).
`fireTableRowsInserted`	New rows inserted.
`fireTableRowsDeleted`	Existing rows Deleted
`fireTableStructureChanged`	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 megvalsításának legegysezrűbb 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. Coparator kiválasztása egy oszlopra (az első megfelelő):

1. Comparator meg van adva a setComparator metódussal.
2. Ha az oszlop típusa String, string comparatora.
3. Ha a getcolumnClass egy Comperable osztállyal tér véssza, a comparator annak a comperTo metódusát használja.
4. Megadott StringConverter esetén a comperator a string reprezentációkkal.
5. Minden más esetben a az oszlop értekeinek 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 cimkék 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 megfontálsokból az értékét karakter tömbben tárolja String helyett.
- A komponens aléprtelmezsként egy „pont”-ot ír minden karakter helyére, megváltoztatása: `setEchoChar` metódussal.
- A begépelt 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');
```

Concurrency in Swing

- A szálkezelés a swing alkalmazásokban is fontos.
- Cél egy olyan felhasználói felület készítése, amely soha nem fagy, mindig válaszol a felhasználói interakciókra, bármit is csináljon éppen.
- Swing 3 féle szállal dolgozik:
  - Kezdeti szálak (initail Threads): az alkalmazást futtató kezdeti szálal.
  - Event dispatch thread: ahol minden eseménykezelő kód és minden swing-el kapcsolatos interakció fut.
  - Háttér szálak (worker - threads): időigényes műveletek háttérben futtatására.
- A szálakat nem szükséges explicit létrehozni, ezeket a Swing kezeli helyettünk.

Initial Threads

- Minden alkalmazáshoz tartozik néhány szál, ahonnan az alkalmazás elindul.
- Általában ez a main thread.
- Swing alkalmazásokban a kezdeti szál nem végez lát el sok feladatot.
- A legfontosabb feladatai
  - egy Runnable objektum létrehozása, amely inicializálja a GUI-t
  - A Runnable objektum ütemezése az event dispatch thread-re.
- A GUI elindítása után az alkalmazást többnyire az interfész eseményei vezérlik.
- Az események rövid taskok végrehajtását váltják ki az EDT-en.

- Az alkalmazás egyéb taskokat is tud az EDT-re vagy háttér szálra ütemezni.
- A kezdeti szálak a GUI létrehozásának ütemezése
  - `SwingUtilites.invokeLater`: Ütemezi a taskot és visszatér
  - `SwingUtilities.invokeAndWait`: Ütemezi a taskot és megvárja, hogy befejődjön.
- Általában a GUI létrehozás ütemezése az utolsó dolog, amit a main szál végez.
- **Minden Swing componenst használó kódnak az EDT-en kell futnia!**

```
SwingUtilities.invokeLater(new Runnable() {
    public void run() {
        createAndShowGUI();
    }
});
```

Event Dispatch Thread (EDT)

- A swing esemény kezelő kódja egy speciális szálon fut.
- A swing compnenseket használó, létrehozó kód is ezen a szálon fut.
- Ez szükséges, mert a legtöbb swing objektum nem szálbiztos.
- Ezeket figyelmen kívül hagyvan előre nem látható, nehezen kezelhető hibákba ütközhetünk
- Az EDT rövid taskok sorozataként fut:
  - A legtöbb, az eseménykezelő metódusok hívása, mint az `actionPerformed`
  - Egyéb: az alkalmazás által ütemezett taskok, az `invokeLater` és `invokeAndWait` metódusok használatával.
- Az EDT taskokknak rövidnek kell lennie.
- `SwingUtilities.isEventDispatchThread`.

Worker Threads – SwingWorker

- Swing alkalmazásból hosszú futásidejű feladatok végrehajtására használható szálak.
- Minden task egy `SwingWorker` példányként reprezentált.
- Ez egy absztrakt osztály, ennek egy leszármazottját kell definiálni.
- SwingWorker a java SE 6 óta.

SwingWorker

- Kommunikációs mechanizmusai
  - `done` metódus definiálható a `SwingWorker` implementációban, a háttér task lefutása után kerül meghívásra, autómatikusan az EDT-n.
  - A `SwingWorker` implementálja a `Future` interfészt így a háttértask adhat vissza értéket. Az interfész lehetővé teszi még a task leállítását (`cancel`) és a task állapotának lekérdezését (befejezedőött, leállított).
  - A háttér szál elérhetővé tud tenni részeredményeket a futás befejeződése előtt a `publish` metódus hívásával.
  - A részeredményeket a `process` metódus dolgozza fel, amely autómatikusan az EDT-n fut.

```
SwingWorker worker = new SwingWorker<ImageIcon[], Void>() {
    public ImageIcon[] doInBackground() {
        final ImageIcon[] innerImgs = new ImageIcon[nimgs];
        for (int i = 0; i < nimgs; i++) {
            innerImgs[i] = loadImage(i+1);
        }
        return innerImgs;
    }

    public void done() {.
        try {   imgs = get();   }  //vagy get(1000);
        catch (InterruptedException ignore) {}
        catch (java.util.concurrent.ExecutionException e) {
            ...
        }
    }
};
```

```
private class FlipTask extends SwingWorker<Void, Integer> {
    protected Void doInBackground() {
        Random random = new Random();
        while (!isCancelled()) {
            publish(random.nextInt(1000));
        }
        return null;
    }
    protected void process(List<Integer> r) {
        ...
    }
}
```

- Háttér task-nak támogatni kell a megszakítását, ez két módon lehetséges:
  - Bejezés interrupt hatására (ahogy a szálaknál), isInterrupted..
  - isCancelled() metódus hívása periódikusan. True-t ad vissza, ha a worker cancel metódusát meghívátk.

Swing Timer

- Egy vagy több Action eventet generál a megadott idő után.
- Az általános timer-el ellentétben, ez a GUI-val kapcsolatos időzítéses feladatok elvégzésére való.
- Előre létrehozott időzítő szálat használ.
- A GUI taskok autómatikusan az EDT-n hajtódnak végre.
- Felhasználási módok:
  - Task végrehajtása egyszer, késleltetés után
  - Ismétlődő feladatok végrehajtása.

Használata

- A timer létrehozásakor meg kell adni egy actionListener-t amely lefut a megadott időben.
- A létrehozáskor meg kell adni a végrehajtások közötti várakozási időt.
- setRepeats(false) hívással megadható, hogy a timer ne ismétlődjön.
- Start metódus indítja a timer-t
- Stop-al megállítható.