

Java Swing MVC labor

Készítette: Budai Péter, BME IIT, 2015.

A feladatok megoldásához felhasználandó osztályok leírásait az alábbi URL-en találja meg:

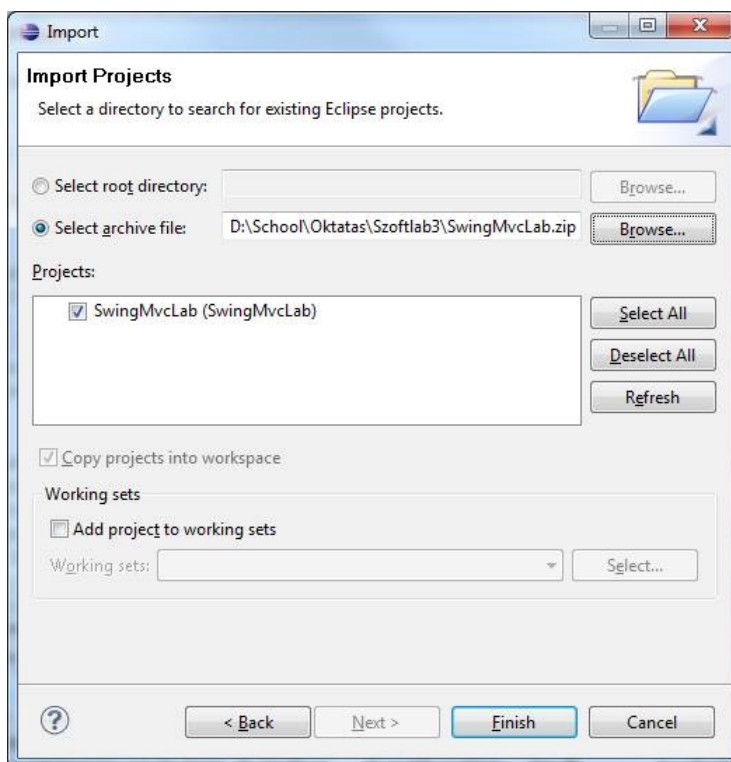
<http://download.oracle.com/javase/8/docs/api/>

Továbbá nagy segítséget jelenthet az alábbi URL-en elérhető *JTable Tutorial* is, amely az összes gyakori (és a feladatban előforduló) problémára mutat megoldást:

<http://download.oracle.com/javase/tutorial/uiswing/components/table.html>

1 Projekt megnyitása

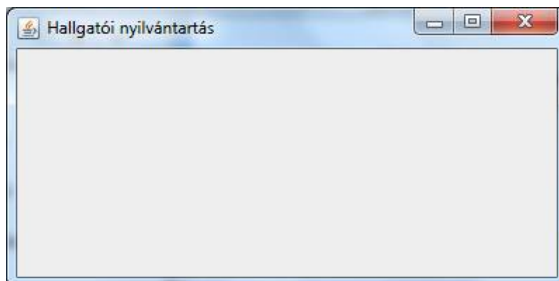
A tárgyhonlap heti laborfeladatához tartozó oldaláról töltsse le a feladat megoldásához használható Eclipse projekt vázat. Ez egy ZIP archívum, amelyet közvetlenül be lehet importálni az Eclipse-be. Ehhez válassza a **File/Import...** menüpontot, majd ott keresse ki a **General/Existing projects into workspace** lehetőséget! A felbukkanó ablakban tallózza ki a letöltött archívumot, majd kattintson a **Finish** gombra (lásd az alábbi ábrát).



A beimportált projektben három Java fájl és egy **.dat** kiterjesztésű állomány található, melyek egy kezdetleges hallgatói nyilvántartó program részei. A **students.dat** fájlban találhatóak a hallgatói adatok (név, neptunkód, aláírás, jegy) szerializálva. A **StudentFrame** osztály valósítja meg az alkalmazás felhasználói felületét, ami egyelőre csak annyit tesz, hogy betölti a hallgatói adatokat

(melyeket a **StudentData** osztály egy példánya tárol a **data** attribútumban), megjelenít egy üres ablakot, bezáráskor pedig újra kimentí az adatokat.

Röviden tanulmányozza át, majd próbálja ki az alkalmazást! Az alábbi üres ablakot kell kapnia:



Fontos, hogy a feladatok megoldása során csak ott módosítsa a kiinduló forráskódot, ahol azt a kommentek nem tiltják!

2 Táblázat létrehozása

Adjon hozzá egy táblázat komponenst (**JTable**) az ablak középső területéhez (**BorderLayout.CENTER**)! Ügyeljen rá, hogy a táblázat töltsse ki az egész ablakot (**setFillViewportHeight** metódus), illetve hogy lapozható legyen a gördítősáv segítségével! Ehhez a táblázatot egy lapozható panelbe kell elhelyezni (**JScrollPane**).

Ahhoz, hogy a táblázatban adatok is szerepeljenek, meg kell adnunk, hogy mit használjon adatforrásként. Ehhez a **StudentFrame** osztályban már szereplő **data** objektumot fogjuk majd használni, ami a **StudentData** osztály egy példánya. De a **StudentData** osztály jelenlegi formájában nem képes ellátni az adatforrás szerepét. Ehhez a **StudentData** osztályt származtatnia kell az **AbstractTableModel** ősosztályból!

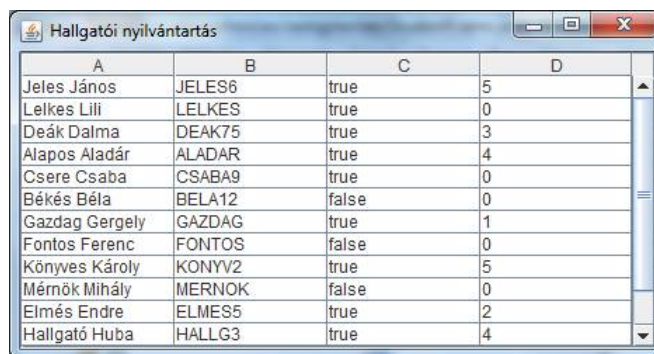
Magyarázat: Az **AbstractTableModel** osztály megvalósítja a **TableModel** interfészt, még hozzá úgy, hogy minden metódus törzse üres. Ezért ha ebből az osztályból származtatunk, sokkal kevesebb metódust kell felüldefiniálnunk, mintha magát a **TableModel** interfészt valósítottuk volna meg a **StudentData** osztállyal.

A **StudentData** osztályban definiálja felül és implementálja az **AbstractTableModel** osztály megfelelő metódusait (**getColumnCount**, **getRowCount**, **getValueAt**) úgy, hogy a táblázat komponens meg tudja jeleníteni az adatokat! *Emlékeztető:* A táblázatnak 4 oszlopa van (név, neptun, aláírás, jegy) és annyi sora, ahány hallgatónk van a **students** listában. Segítségül a **getValueAt** függvényt megadjuk:

```
public Object getValueAt(int rowIndex, int columnIndex) {
    Student student = students.get(rowIndex);
    switch(columnIndex) {
        case 0: return student.getName();
        case 1: return student.getNeptun();
        case 2: return student.hasSignature();
        default: return student.getGrade();
    }
}
```

Ha elkészült a **StudentData** osztály, akkor csak annyi van hátra, hogy a táblázat komponensen beállítsuk, hogy a **data** attribútumot használja adatmodellként. Ehhez vagy adjuk át konstruktor-paraméterként a **JTable** objektum létrehozásakor, vagy utólag állítsuk be a **JTable** objektum **setModel** függvényével!

Ha mindent jól csinált, akkor valami hasonlót kell kapnia:



A	B	C	D
Jeles János	JELES6	true	5
Lelkes Lili	LELKES	true	0
Deák Dalma	DEAK75	true	3
Alapos Aladár	ALADAR	true	4
Csere Csaba	CSABA9	true	0
Békés Béla	BELA12	false	0
Gazdag Gergely	GAZDAG	true	1
Fontos Ferenc	FONTOS	false	0
Könyves Károly	KONYV2	true	5
Mérnök Mihály	MERNOK	false	0
Elmes Endre	ELMES5	true	2
Hallgató Huba	HALLG3	true	4

3 Oszlopok testre szabása

Fejlessze tovább a programot úgy, hogy a táblázat fejlécei a megfelelő szövegeket (sorrendben *Név*, *Neptun*, *Aláírás*, *Jegy*) tartalmazzák! Ehhez az **AbstractTableModel** őssztály megfelelő metódusát (**getColumnName**) kell felüldefiniálnia a **StudentData** osztályban.

Szintén az őssztály egy másik metódusának (**getColumnClass**) felüldefiniálásával érje el, hogy a táblázat *Aláírás* és *Jegy* oszlopai a típusuknak megfelelő formában kerüljenek megjelenítésre! Például az aláírásnál a *true/false* értékek helyett egy jelölőnégyzet szerepeljen, ahogy az alábbi ábrán is látható!



Név	Neptun	Aláírás	Jegy
Jeles János	JELES6	<input checked="" type="checkbox"/>	5
Lelkes Lili	LELKES	<input checked="" type="checkbox"/>	0
Deák Dalma	DEAK	<input checked="" type="checkbox"/>	3
Alapos Aladár	ALADAR	<input checked="" type="checkbox"/>	4
Csere Csaba	CSABA9	<input checked="" type="checkbox"/>	0
Békés Béla	BELA12	<input type="checkbox"/>	0
Gazdag Gergely	GAZDAG	<input checked="" type="checkbox"/>	1
Fontos Ferenc	FONTOS	<input type="checkbox"/>	0
Könyves Károly	KONYV2	<input checked="" type="checkbox"/>	5

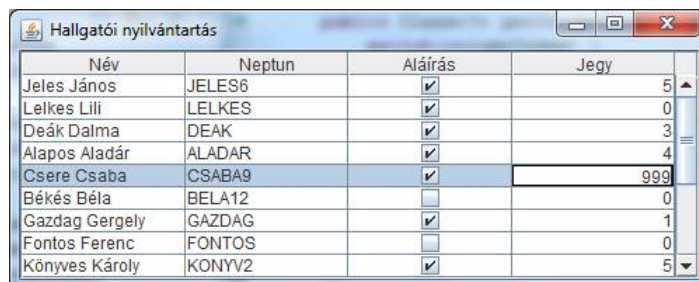
*Java*slat: Az oszlop típusának visszaadásához el kell kérnie a megfelelő típushoz tartozó típusleíró. Ezt az osztálynév után írt **.class** kulcsszóval kaphatjuk meg. Például **String** esetén **String.class**-t kell visszaadni, **Boolean** esetén **Boolean.class**-t, **Integer** esetén **Integer.class**-t.

4 Cellák szerkesztése

Tegye lehetővé a felhasználó számára a jegy oszlopában szereplő értékek módosítását! Ehhez bővítsa a **StudentData** osztályt az **AbstractTableModel** megfelelő metódusainak felüldefiniálásával!

Egyrészt tegye lehetővé a 3. és 4. oszlopban szereplő cellák szerkesztését (**isCellEditable**), valamint az osztály képes legyen eltárolni a felhasználói felületen bevitt értéket a **students** listában (**setValueAt**).

Próbálja ki az alkalmazást! Indítsa el, módosítsa valamelyik cella értékét, majd zárja be, és indítsa el újra a programot! Ha mindent jól csinált, a módosított érték az alkalmazás újraindítása után is megmarad.



Név	Neptun	Aláírás	Jegy
Jeles János	JELES6	<input checked="" type="checkbox"/>	5
Lelkes Lili	LELKES	<input checked="" type="checkbox"/>	0
Deák Dalma	DEAK	<input checked="" type="checkbox"/>	3
Alapos Aladár	ALADAR	<input checked="" type="checkbox"/>	4
Csere Csaba	CSABA9	<input checked="" type="checkbox"/>	999
Békés Béla	BELA12	<input type="checkbox"/>	0
Gazdag Gergely	GAZDAG	<input checked="" type="checkbox"/>	1
Fontos Ferenc	FONTOS	<input type="checkbox"/>	0
Könyves Károly	KONYV2	<input checked="" type="checkbox"/>	5

5 Adatmodell módosítása

Tegye lehetővé, hogy a táblázatba új hallgatókat is fel lehessen venni!

Ehhez vegyen fel az ablak alsó területére (**BorderLayout.SOUTH**) egy új panelt (**JPanel**), amin szerepeljen két beviteli mező a név és neptunkód számára (**TextField**, 20 és 6 karakter szélesek), a hozzájuk tartozó címkék (**JLabel**) és egy gomb (**Button**, azzal a felirattal, hogy *Felvesz*)! *Java*slat: A nevet és a neptunkódot tartalmazó **TextField**-eknek vegyen fel tagváltozót az ablak osztályba (pl. **nameField** és **neptunField**), mivel a gomb eseménykezelőjében szükség lesz rájuk! A panel elrendezése maradhat az alapértelmezett **FlowLayout**.



Név	Neptun	Aláírás	Jegy
Jeles János	JELES6	<input checked="" type="checkbox"/>	5
Lelkes Lili	LELKES	<input checked="" type="checkbox"/>	0
Deák Dalma	DEAK	<input checked="" type="checkbox"/>	3
Alapos Aladár	ALADAR	<input checked="" type="checkbox"/>	4
Csere Csaba	CSABA9	<input checked="" type="checkbox"/>	0
Békés Béla	BELA12	<input type="checkbox"/>	0
Gazdag Gergely	GAZDAG	<input checked="" type="checkbox"/>	1
Fontos Ferenc	FONTOS	<input type="checkbox"/>	0
Könyves Károly	KONYV2	<input checked="" type="checkbox"/>	5
Mérnök Mihály	MERNOK	<input type="checkbox"/>	0

Név: Neptun:

A hallgatói adatokat kezelő **StudentData** osztályban vegyen fel egy új metódust, amivel új hallgatót lehet felvenni az adatmodellbe:

```
public void addStudent(String name, String neptun) { ... }
```

A metódus adja hozzá az új hallgató adatait a hallgatók listájához! A hallgató neve és neptunkódja legyen a paraméterek által megadott érték, az aláírása **false**, a jegye pedig **0** (nulla) értékű! Ahhoz, hogy az új hallgató a felhasználói felületen is megjelenjen, a táblázatot értesíteni kell arról, hogy az

adatok megváltoztak. Ehhez hívja meg az **AbstractTableModel** osztály megfelelő metódusát (**fireXXX**)! Melyik metódust célszerű használni?

Végül adjon hozzá a **StudentFrame** osztályban egy eseménykezelőt (**ActionListener**) a *Felvesz* feliratú gombhoz! A gomb megnyomására adja hozzá az új hallgatót (az imént elkészített metódus segítségével) a hallgatói adatokhoz, még hozzá a két beviteli mezőben szereplő névvel és neptunkóddal!

Ha mindent jól csinált, akkor a gomb lenyomására megjelenik az új hallgató a táblázat utolsó sorában.



Név	Neptun	Aláírás	Jegy
Mérnök Mihály	MERNOK	<input checked="" type="checkbox"/>	0
Elmes Endre	ELMES5	<input checked="" type="checkbox"/>	2
Hallgató Huba	HALLG3	<input checked="" type="checkbox"/>	4
Nagy Norbert	NAGY31	<input checked="" type="checkbox"/>	3
Író Irma	IRO448	<input type="checkbox"/>	0
Kiss Pista	KISSP1	<input type="checkbox"/>	0
Szabó Szilárd	SZABO2	<input type="checkbox"/>	0
Új Hallgató	UJ1234	<input type="checkbox"/>	0

Név: Új Hallgató Neptun: UJ1234 **Felvesz**

6 Rendezés

Módosítsa úgy az alkalmazást, hogy a táblázat fejlécére kattintva az adott oszlop szerint növekvő sorrendben történjen a sorok rendezése! A rendezésre kiválasztott oszlopon történő újabb kattintás ezúttal az adott oszlop szerint csökkenő sorrendbe rendezze a sorokat! Újabb kattintás esetén megint növekvőbe, és így tovább felváltva. Egy másik oszlop kiválasztásakor először mindig növekvő legyen a rendezés iránya!



Név	Neptun	Aláírás	Jegy
Jeles János	JELES6	<input checked="" type="checkbox"/>	5
Könyves Károly	KONYV2	<input checked="" type="checkbox"/>	5
Alapos Aladár	ALADAR	<input checked="" type="checkbox"/>	4
Hallgató Huba	HALLG3	<input checked="" type="checkbox"/>	4
Deák Dalma	DEAK75	<input checked="" type="checkbox"/>	3
Nagy Norbert	NAGY31	<input checked="" type="checkbox"/>	3
Elmes Endre	ELMES5	<input checked="" type="checkbox"/>	2
Gazdag Gergely	GAZDAG	<input checked="" type="checkbox"/>	1
Békés Béla	BELA12	<input type="checkbox"/>	0

Név: Neptun: **Felvesz**

Ehhez be kell állítania a **JTable** komponensen, hogy milyen rendező objektumot használjon a táblázat sorainak rendezéséhez (**setRowSorter**). Az alapértelmezett rendező osztály (**TableRowSorter**) egy példánya most tökéletesen megfelel a célra.

Javaslat: A megoldás sokkal egyszerűbb, mint gondolná. Ha elakad, nézze meg a feladatkiírás elején szereplő URL-en a *JTable Tutorial*-t, abban még példakód is szerepel.

7 Cellák testre szabása

Módosítsa az alkalmazást oly módon, hogy (az alábbi ábrához hasonlóan) a táblázatban különböző háttér-színnel legyenek jelölve azok a hallgatók, akik átmentek (**zöld**) és azok, akik egyelőre bukásra állnak (**piros**), mert vagy az aláírásuk nincs meg vagy a jegyük rosszabb kettesnél!



Név	Neptun	Aláírás	Jegy
Jeleš János	JELES6	<input checked="" type="checkbox"/>	5
Lőrinc Lili	LELKES	<input checked="" type="checkbox"/>	0
Deák Dalma	DEAK75	<input checked="" type="checkbox"/>	3
Alajos Aladár	ALADAR	<input checked="" type="checkbox"/>	4
Csere Csaba	CSABA9	<input checked="" type="checkbox"/>	0
Békes Béla	BELA12	<input type="checkbox"/>	0
Gazdag Gergely	GAZDAG	<input checked="" type="checkbox"/>	1
Fontos Ferenc	FONTOS	<input type="checkbox"/>	0
Könyves Károly	KONYV2	<input checked="" type="checkbox"/>	5
Mernők Mihály	MERNOK	<input type="checkbox"/>	0
Elmes Endre	ELMES5	<input checked="" type="checkbox"/>	2
Hallgató Hanna	HALLG3	<input checked="" type="checkbox"/>	4

Ehhez egy (vagy több) saját **StudentTableCellRenderer** osztályt kell implementálnia a **StudentFrame** osztályon belül, ami megvalósítja a **TableCellRenderer** interfészt. Ezt az osztályt azután hozzá lehet rendelni a **JTable** komponenshez (**setDefaultRenderer**), aminek hatására az adott típusú cellák kirajzolásához a megadott saját osztályt fogja használni a táblázat.

A **StudentTableCellRenderer** osztály **getTableCellRendererComponent** metódusa adja vissza azt a Java Swing komponenst, amit az adott cella megjelenítéséhez a **JTable**-nek használnia kell.

JavaSlat: Mivel három különböző adattípust megjelenítő cella van a táblázatunkban, célszerű inkább valamiféle csomagoló osztályt létrehozni, mely az eredeti megjelenítő (**table.getDefaultRenderer()**) objektumot használja, és csak módosítja az általa megfelelően előkészített komponenst oly módon, hogy az a megfelelő háttérszínnel jelenjen meg.

```
class StudentTableCellRenderer implements TableCellRenderer {

    private TableCellRenderer renderer;

    public StudentTableCellRenderer(TableCellRenderer defRenderer) {
        this.renderer = defRenderer;
    }
}
```



```
public Component getTableCellRendererComponent(JTable table,
Object value, boolean isSelected, boolean hasFocus,
int row, int column) {
    Component component = renderer.getTableCellRendererComponent(
        table, value, isSelected, hasFocus, row, column);

    // Kikeressük az éppen megjelenítendő hallgatót a külső,
    // StudentFrame osztály data tagváltozójából,
    // megállapítjuk, hogy bukásra áll-e vagy sem,
    // és ez alapján átállítjuk a komponens háttérszínét:
    // component.setBackground(...)

    return component;
}
```

Figyeljen arra, hogy a rendezés miatt a megjelenítő osztály által kapott sor indexek eltérhetnek az adott hallgató adatainak az adatmodellben elfoglalt helyétől! Az aktuális modellbeli indexet a táblázathoz tartozó **RowSorter** példány megfelelő metódusa tudja megmondani nekünk (`table.getRowSorter().convertRowIndexToModel(row)`).