## Spinner

A benne lévő érték léptethető (Nyíl billentyűk vagy nyilakra kattintás).

Például:



Az Editable tulajdonság határozza meg, hogy be lehessen-e írni más értéket.

Konstruktor:

|  |  |
| --- | --- |
| Spinner<T>() | spinner létrehozása |

Például:

private Spinner<Integer> spResz;

Metódusok:

|  |  |
| --- | --- |
| void **setValueFactory**(SpinnerValueFactory<T> value) | az értékeket generáló objektum megadása |
| SpinnerValueFactory<T> **getValueFactory**() | az értékeket generáló objektum lekérdezése |

Az inicializáláskor meg kell adni azt a metódust, amely a léptetéskor a következő értéket generálja.

Ha például egész számokat szeretnénk léptetni 1 és 30 között, 10 kezdőértékkel, akkor ezt kell írni az initialize metódusba:

spResz.setValueFactory(  
 new SpinnerValueFactory.IntegerSpinnerValueFactory(1, 30, 10));

Ezután így lehet a spinner értékét lekérdezni, illetve beállítani:

Integer r = spResz.getValue();  
spResz.getValueFactory().setValue(15);

## ObservableList

Olyan lista, amelynek elemeit meg lehet jeleníteni egy vezérlőben (ComboBox, ListView, …).

A vezérlő getItems metódusával kérdezhető le.

A JCF bővítése. Az ArrayList-től abban különbözik, hogy figyeli az elemek változásait.

Az ArrayList metódusai használhatók az ObservableList esetén is: add, AddAll, get, set, remove, clear, contains, isEmpty, size.

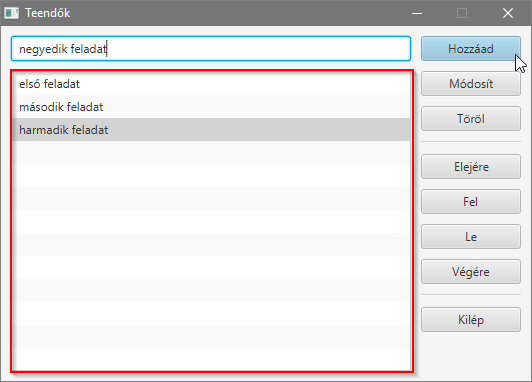
## ListView

Objektumokat listáz. Ha az objektum nem String típusú, akkor a toString() metódusával átalakítja.

Alapértelmezésben egyszerre egy elem jelölhető ki, de ez módosítható.

Egy elem kijelölése: kattintás, nyilak, Home, End, PgUp, PgDn.

Szükség esetén automatikusan megjelennek a görgetősávok.



Konstruktorok:

|  |  |
| --- | --- |
| ListView<T>() | üres lista létrehozása |
| ListView<T>(ObservableList<T> items) | új lista a megadott elemekkel |

Metódusok:

|  |  |
| --- | --- |
| ObservableList<T> **getItems**() | elemek listájának lekérdezése |
| void **setItems**(ObservableList<T> items) | elemek listájának beállítása |

Elemek kezeléséhez le kell kérni az elemek listáját:

ListView lista = new ListView();  
lista.**getItems().addAll**("sonka", "szalámi", "sajt"); // több elem hozzáadása  
lista.**getItems().add**("gomba"); // egy elem hozzáadása  
lista.**getItems().get**(0); // 0-dik elem tartalma  
lista.**getItems().set**(0,"hús"); // 0-dik elem módosítása  
lista.**getItems().remove**(0); // 0-dik elem törlése

A kijelöléssel kapcsolatos műveletekhez le kell kérni a kijelölésmodellt:

lista.**getSelectionModel().select(0)**; // 0-dik lem kijelölése  
// kijelölt elem indexe (ha nincs kijelölve: -1):  
int i = lista.**getSelectionModel().getSelectedIndex()**;   
// kijelölt elem tartalma (ha nincs kijelölve: null):  
String s = lista.**getSelectionModel().getSelectedItem()**;

Eseménykezelő, amely a kijelölés változásakor fut le:

lista.**getSelectionModel().selectedItemProperty().addListener**(  
 (v, regi, uj) -> {  
 // *utasítások*

});

## TableView és TableColumn

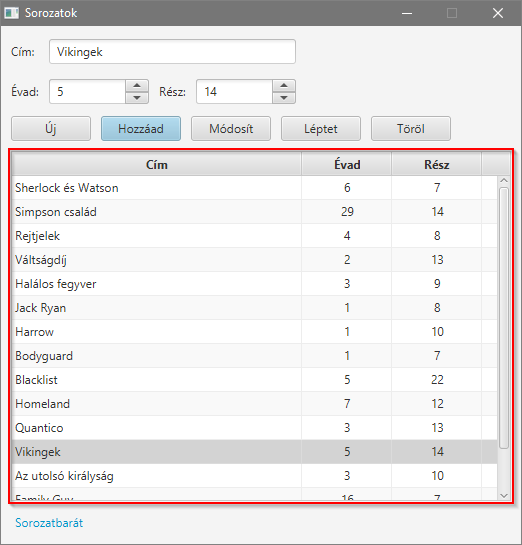
A TableView objektum egy táblázatot jelenít meg.   
Az oszlopokat TableColumn típusú objektumok alkotják, amelyeket egy ObservableList tárol.

A sorok egy-egy objektumot jelenítenek meg (pl. Sorozat). Ennek egy-egy mezője kerül az oszlopokba (pl. cím, evead, resz). Minden mezőhöz készíteni kell egy getter és egy setter metódust. Például ha a mező neve cím, akkor kell egy getCim és egy setCim metódus. Az elnevezésektől nem lehet eltérni!

A táblázat sorait is egy ObservableList tárolja.

Alapértelmezésben egyszerre egy sor jelölhető ki, de ez módosítható.   
Egy elem kijelölése: kattintás, nyilak, Home, End, PgUp, PgDn.

Szükség esetén automatikusan megjelennek a görgetősávok.



**TableColumn** konstruktorok:

|  |  |
| --- | --- |
| TableColumn<S, T>() | üres oszlop létrehozása S a sorok típusa,  T az oszlop típusa |
| TableColumn<S, T>(String fejlec) | üres oszlop létrehozása a megadott fejléccel |

**TableColumn** metódusok:

|  |  |
| --- | --- |
| void **setText**(String title) | fejléc szövegének megadása |
| void **setSortable**(boolean rendezheto) | oszlop szerinti rendezhetőség megadása (alapértelmezett: true) |
| void **setCellValueFactory** (PropertyValueFactory f) | Olyan objektum megadása, amely előállítja az oszlop celláinak értékét. Általában a PropertyValueFactory osztály egy példánya. |

Oszlop létrehozása és cell factory megadása úgy, hogy a cím mező jelenjen meg az oszlopban:

TableColumn<Sorozat, String> oCim = new TableColumn("Cím");  
oCim.setCellValueFactory(new PropertyValueFactory<>("cim"));

**TableView** konstruktorok:

|  |  |
| --- | --- |
| TableView<S>() | üres táblázat létrehozása S a sorok típusa, |
| TableView<S>(ObservableList lista) | üres táblázat létrehozása a megadott listával |

**TableView** metódusok:

|  |  |
| --- | --- |
| ObservableList **getColumns**() | a táblázat oszlopait tartalmazó lista lekérdezése |
| ObservableList **getItems**() | a sorok adatait tartalmazó lista lekérdezése |
| void **setItems**(ObservableList lista) | adatokat tartalmazó lista hozzárendelése a táblázathoz |
| TableViewSelectionModel  **getSelectionModel**() | kijelölési modell lekérdezése |

Oszlop hozzáadása a táblázathoz:

TableView<Sorozat> tablazat = new TableView<>();  
tablazat.**getColumns**().**add**(oCim);

A sorok kezeléséhez le kell kérni az elemek listáját:

tablazat.**getItems().add**(new Sorozat()); // új sorozat hozzáadása  
tablazat.**getItems().set**(i, new Sorozat()); // i-dik sor módosítása  
tablazat.**getItems().remove**(i); // i-dik sor törlése  
tablazat.**getItems().size(**); // sorok száma

A kijelöléssel kapcsolatos műveletekhez le kell kérni a kijelölésmodellt:

tablazat.**getSelectionModel().select(i)**; // i-dik sor kijelölése  
// kijelölt elem indexe (ha nincs kijelölve: -1):  
int i = tablazat.**getSelectionModel().getSelectedIndex()**;   
// kijelölt elem tartalma (ha nincs kijelölve: null):  
String s = tablazat.**getSelectionModel().getSelectedItem()**;

Eseménykezelő, amely a kijelölés változásakor fut le:

tablazat.**getSelectionModel().selectedItemProperty().addListener**(  
 (v, regi, uj) -> {  
 // *utasítások*

});

## Hangfájl lejátszása

A hangfájlokat az AudioClip osztály segítségével játszhatjuk le. Deklaráció:

**AudioClip** clip;

A hangfájlt a forrásfájlokhoz kell másolni, és utána így inicializálni a változót:

clip = new **AudioClip**(getClass().getResource("JB.wav").toString());

Az ismétlések számának megadása:

clip.**setCycleCount**(INDEFINITE);

Az INDEFINITE végtelen ismétlést jelent, helyette egy számot is meg lehet adni.

Lejátszás indítása:

clip.**play**();

Lejátszás megállítása:

clip.**stop**();

Annak vizsgálata, hogy megy-e a lejátszás:

if (clip.**isPlaying**()) …

## Időzítés

**Egyszeri** időzítés:

Timeline idovonal = new Timeline(new KeyFrame(  
 Duration.millis(2000),  
 e -> csinald()));  
idovonal.play();

2 másodperc múlva meghívja a csinald() metódust.

**Periódikus** időzítés:

Timeline idovonal = new Timeline(new KeyFrame(  
 Duration.millis(2000),  
 e -> csinald ()));  
idovonal.setCycleCount(Timeline.INDEFINITE);  
idovonal.play();

2 másodpercenként múlva meghívja a csinald() metódust.

Időzítés **kikapcsolása**:

idovonal.stop();