

# Szórakoztatóelektronikai eszközök programozása

## házi feladat

Berze Gergő Dávid

### Ötlet:

Webshop alkalmazás JavaFX technológiával PC-re.

A program egy áruház árukészletét tartja nyilván. A megvalósítás a tulajdonosi és a vásárlói felületet is tartalmazza, továbbá egy bejelentkező ablakot.

Az eladó termékekről tárolt adatok:

- a termék neve
- ára
- kategóriája (kategóriákba van sorolva minden termék)
- készleten lévő darabok száma
- egy fénykép a termékről (pontosabban a kép elérési útvonala, alapértelmezetten: *photos mappa*)

### Funkcionalitások:

Bejelentkező ablak:

- bejelentkezés
- nyelv választás a felhasználói felülethez

Admin (tulajdonos):

- kategóriákat tud létrehozni és törölni
- termékeket tud létrehozni és törölni
- termékeket új/másik kategóriába sorolhat
- mindenről látja a nyilvántartást

User (vásárló):

- kereshet kategória és név szerint is
- „kosár” funkció
- vásárlás

## Megvalósítás:

A program a Model View Control architektúrán alapszik, és próbáltam figyelembe venni az OO tervezési heurisztikákat.

### Felhasznált fájlok és szerepük:

Mivel az ablakok felépítése nagyon hasonló. Az egyszerűség kedvéért, ahol a megjegyzésben az „MVC” kulcsszó szerepel, az értelmezés a következő:

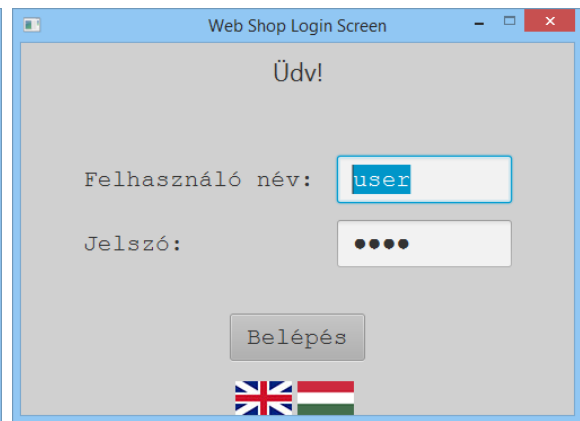
- \*.java – Controller
- \*.fxml – Modell
- \*.css – View

fájl(ok) neve	szerepük	megjegyzés
Main.java	main osztály, betölti a bejelentkező ablakot	
Logger.java, LogXML.fxml, application.css, EN.properties, HU.properties	bejelentkező ablakhoz használt fájlok	<i>MVC</i> *.properties – többnyelvűséghez használt kifejezések fájljai
Admin.java, AdminXML.fxml	tulajdonos ablak és funkciói	<i>MVC</i> nincs különösebb formázása ( <i>View</i> )
User.java, UserXML.fxml, User.css	vásárló felület és funkciói	<i>MVC</i> a megjelenés Windows 8-as témákra emlékeztet
Item.java	tárolt termékek	fájlba írható
Stock.java	raktárban tárolt termékek listája, és a raktár módosításának funkciói	fájlba írható
Cart.java	kosár és kezelése	

### Bejelentkező ablak:



1. ábra: Angol bejelentkező felület



2. ábra: Magyar bejelentkező felület

Az inicializációs szakaszban írtam felül a zászlók (kattintás) esemény kezelőit.

Így az ablak alján lévő zászlókra (képekre) kattintva tudunk nyelvet választani. Egy GridPane-ben helyezkednek el a Labelök, a TextField, a gomb és a PasswordField. A gombra kattintva, illetve bármelyik mezőben az *Enter* billentyűt lenyomva tudunk belépni.

A jogosultság ellenőrzés nagyon egyszerű: sztringek összehasonlításán alapszik. Amennyiben jó adatokat adtunk meg, azok függvényében betöltődik egy másik *Stage*.

Kétféle Stage töltődhet be, az Admin és a User. Azért lettek nagyban különszedve, mert az Admin felület az alap JavaFX-es kinézetet kapta, míg a felhasználói élmény érdekében, a User egy, a Windows 8 rendszertől jellemző, vizuális megjelenést kapott.

Ezt egy .css fájl betöltésével értem el.

A gomb Action Eventet a mezők pedig Key Eventet kezelnek.

Jelenleg két felhasználó van: admin és user (nem kis- nagybetű érzékeny).

## Admin felület:

The screenshot displays an administrative interface with a light gray background and a red 'X' icon in the top right corner. It is divided into two main sections: 'Categories' on the left and 'Items' on the right. The 'Categories' section contains a list of categories: 'alkohol', 'szerszám', 'tészta', 'tejtermék', 'háztartás', 'édesség', and 'üres'. Below this list are buttons for 'Add Category' and 'Remove Category...', followed by an empty text input field. The 'Items' section contains a list of items: 'Soproni', 'Kalapács', 'Szarvacska', 'Tej', 'Mackó sajt', 'Szóda patron', and 'Csoki'. To the right of this list are buttons for 'Add new item' and 'Remove item'. Below the items list, there is a 'Recategorize' button and a text input field. Below the input field, the current category is shown as 'alkohol', followed by 'Price 210 Ft' and 'Pieces 98'. At the bottom center of the interface, the text 'Welcome!' is displayed.

3. ábra: Admin ablak

Az inicializációs szakaszban betölti a fájlba írt árukészletet; amennyiben ez nem lehetséges, egy, az ablak alján lévő „párbeszéd panelben” tájékoztat minket a problémáról. Ezután betölti a *ListView*-kba az árukészlet információit, és létrehoz, majd beállít egy *Tooltip*<sup>1</sup>. Az ablakon egy *BorderPane*-ben van elhelyezve minden, azon belül *H*- és *VBox*okban.

Az *Item* listában, ha kiválasztunk valamit, megjelennek a rá vonatkozó információk. Ezt a *ListView* *getSelectedItem* metódusával és *Label*ökkel valósítottam meg. Kiválasztás után a terméket, egy másik kategóriába helyezhetjük, a *Recategorize* gombbal. Ezt úgy tudtam megvalósítani, hogy a termékek listájából először eltávolítom, majd az új kategóriával hozzáadom azt.

---

<sup>1</sup> Lásd később.

Az *Add new item* gomb megjelenít egy kitölthető űrlapot.

4. ábra: Admin ablak a megjelenő ablakkal

Ezekben a mezőkben kell megadni az új termék adatait.

A hozzá tartozó kép elérési útja, ki van töltve, egy alapértelmezett kép elérési útjával.

Az *Open* gomb megnyit egy fájl böngészőt, amivel könnyebben lehet a képek közül választani.

A *Create* gomb blokkolva van, amíg ki nem töltöttünk minden mezőt.

A *Price* és *Pieces* mezőkbe, csak egész értékű szám karakterek kerülhetnek, ezért rendeltem hozzá az előbb említett *Tooltip*-et, ami megjelenik akkor is, ha a *Create*-re kattintunk, és rosszul töltöttük ki ezeket.

Amennyiben jól töltöttünk ki mindent, az eseménykezelő létrehozza az új terméket és átadja az árukészletnek.

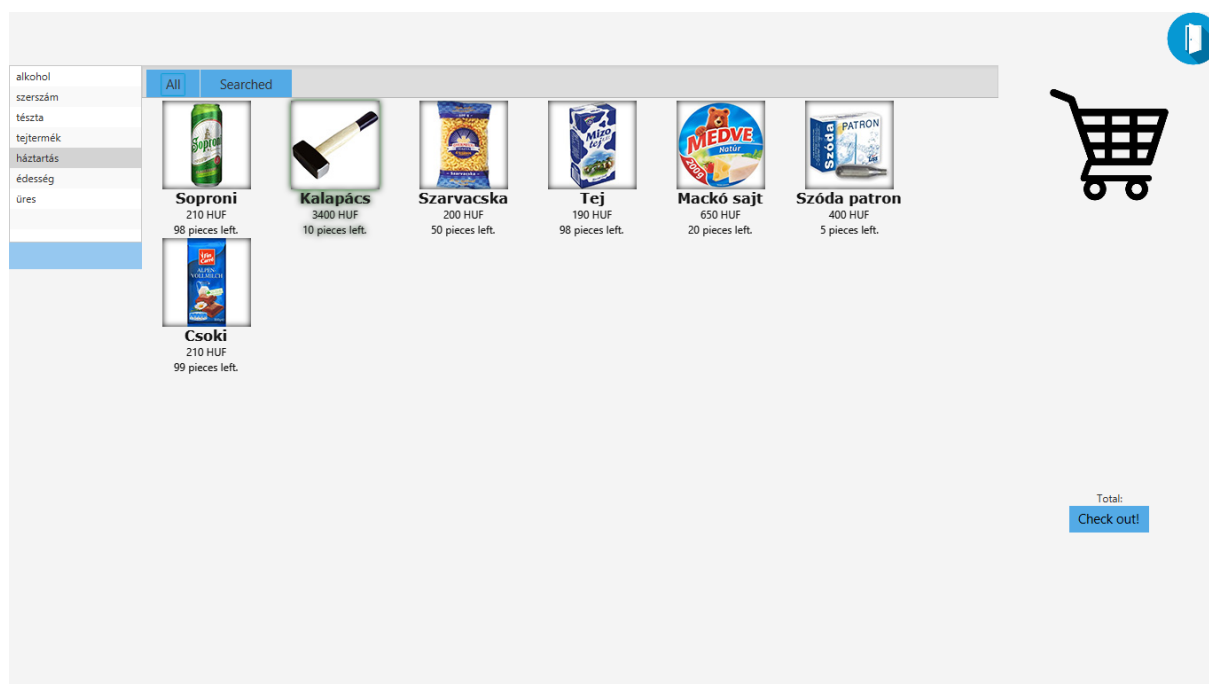
Ha hozzá adtunk egy új terméket a listához, akkor az imént megjelent ablak újra, láthatatlanná válik.

Kategória és termék eltávolítására is van lehetőség.

A párbeszéd panel tájékoztat minket a műveletek sikerességéről.

Kilépni a jobb-felső sarokban lévő piros X-szel tudunk, minekután a program menti a változásokat, felülírva a korábbi állapotot!

## Vásárló felület:



5. ábra: Ragyogás a Kalapácscon

Az inicializációs rész itt is a beolvasással kezdődik, majd az ablak kitöltésével folytatódik.

Itt is egy *BorderPane*-be került minden, aminek a középső részében egy *TabPane*, 2 *Tab*-el, azon belül pedig egy *GridPane* van. Ebbe a *GridPane*-be töltődnek be az elemek, külön *VBox*okba helyezve.

A *VBox*szok feltöltése:

Az árukészleten végigiterálva betöltöm a termék képét egy *Image*-be, és külön *Label*ökbe beírom a termék adatait. Ezeket hozzáadom a *VBox*hoz, majd beállítom annak méreteit. A *VBox* ezután kap egy *Transition* effektet.

Itt írom felül a box kattintás és „*OnMouseEntered*” eseményeit. Ha a kurzort a doboz fölé húzzuk, kap egy külső, zöldes ragyogást.

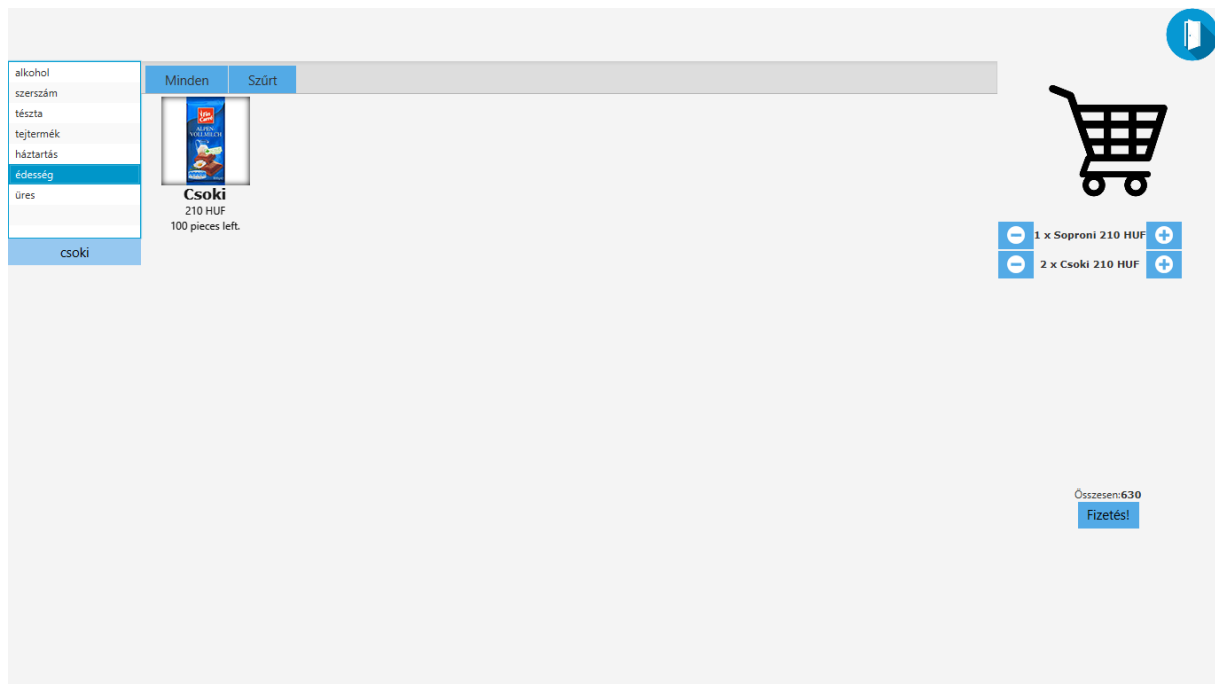
A rács elemeire kattintva a Kosárba tudjuk őket pakolni. Ha már a kosárban van egy termék és újra rákattintunk, akkor a kosárban lévő darabok száma eggyel megnő.

A baloldalon fel van sorolva az összes kategória, melyekre, ha rákattintunk, akkor csak az abba a kategóriába tartozó elemek látszanak.

A kategóriák alatt szerepel egy kereső sáv, amivel név alapján tudunk keresni.

Amennyiben kerestünk, vagy választottunk egy kategóriát, csak a szűrésnek megfelelő elemek jelennek meg. Erre a célra szolgál a *Searched/Szűrt* fül.

Ha újra az összes terméket szeretnénk látni a szűrés után, akkor a *Minden/All* fülre kell kattintanunk.



6. ábra: Keresés és kosár reprezentálása

Jobb oldalon, egy kép alatt, egy VBoxba kerülnek a kosárba tett elemek.

+/- gombokkal lehet növelni illetve csökkenteni a gombok közt szereplő termék kosárban lévő számát. Ha a mínusz gombbal elérjük a nullát, a termék kikerül a kosárból.

Alább egy összeget nyilvántartó Label és egy *Fizetés!/Check out!* feliratú gomb található.

A fizetés funkció jelenleg csak levonja a raktárban lévő termékek számából a megvásároltakét, és visszanyitja a kezdő képernyőre.

Ha kilépünk, de nem fizettünk, akkor a kosár tartalma elveszik.

A termékek és a kosár változásának frissítésére, egy nem túl elegáns, de hatékony módszert alkalmaztam: minden elemet törlök, majd újra feltöltöm a már változott adatokkal.

### További osztályok:

1. Item:

A tárgyak modellezéséért felelős, így paraméterei a tárgyról nyilvántartott információk, és metódusai pedig, ezeknek a getter, setter függvényei.

A feladat komplexitási követelményei miatt, nem adatbázis alapú a rendszer, hanem csak egyszerűen fájlba szerializál. (Ezért implementálja ez az osztály a *Serializable* interfészt.)

2. Stock:

Ez az osztály modellezi a valóságbeli raktárt, így kettő listát tartalmaz: a raktáron lévő tárgyakat<sup>2</sup>, és, hogy milyen kategóriákba vannak azok sorolva.

Ezeket először egy *ArrayList*-ben tároltam, de probléma merült fel velük, mert a JavaFX nem tudja *ListView*-ban megjeleníteni az *ArrayList*-eket. Ezért *ObservableList*-et alkalmaztam, ami szintén nem volt jó, mert az *ObservableList* nem szerializálható. Végül a *Stock* osztályban visszatértem az *ArrayList*-ekhez (a szerializálhatóság érdekében,) és az *Admin* illetve *User* osztályban (, amik használják a grafikus elemeket) létrehoztam egy *Convert* metódust, ami egy *ArrayList*-et kap paraméterként, és ennek a listának a tartalmát adja vissza egy *ObservableList*-ben, amit később már tudtam használni.

További metódusai már szerepeltek fentebb, ezeken kívül főként getter függvényei vannak.

3. Cart:

Nem létfontosságú osztály, csak a Kosár kezelését könnyítette meg nagyban.

Nevet, árat és darabszámot tartalmaz, illetve ezek manipulációihoz kapcsolódó metódusokat.

---

<sup>2</sup> Mivel *Item*-eket tárol a listánk, így ez az osztály is *Serializable*.