Papp János István  
E63L0I  
Programtervező informatikus BSc

Java Programozás  
2. Beadandó programozási feladat

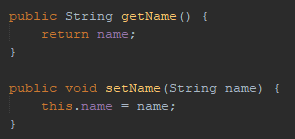
# A feladat

A feladat egy tetszőleges bolt online vásárlási felületének megvalósítása volt Java programozási nyelven, grafikus felülettel. 88 terméket gyűjtöttem össze, eltároltam a termékek nevét, árát, típusát, valamint a belőlük raktáron lévő mennyiséget.

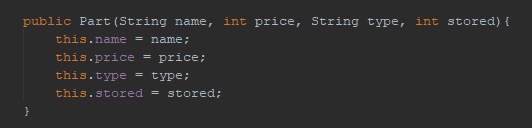
# A feladat megoldása

## Part osztály

Kezdve a legegyszerűbb felépítésű osztállyal, a part osztály a tárolt számítógép alkatrészek példányosításához szükséges. Attribútumai a termékek attribútumainak felelnek meg, amiknek az osztály konstruktora (1. ábra) ad értéket. Ezen kívül az osztály tartalmazza az attribútumokat külön elérő és beállító settereket és gettereket (2. ábra).



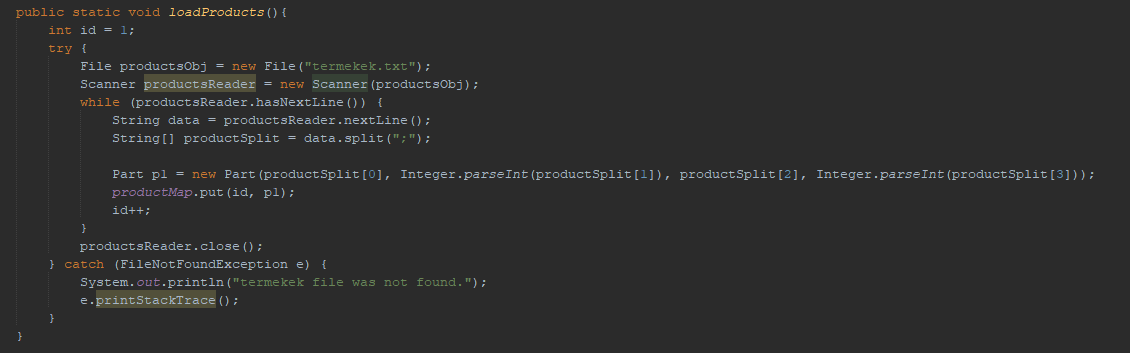
2. ábra: egy setter és egy getter.



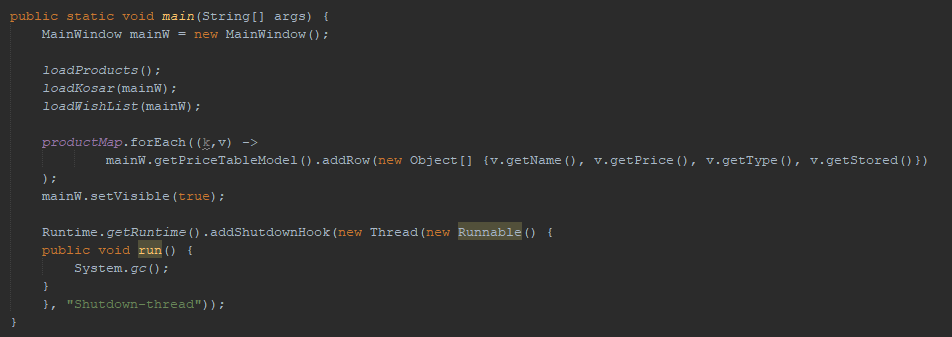
1. ábra: a Part osztály konstruktora.

## MainController osztály

A program fő osztálya, a MainController osztály felel az adatok fájlból való kezdeti betöltéséért (3. ábra) és a fő ablak osztály példányosításáért (4. ábra). Ezen kívül a program bezárásakor még meghívja a java garbage collectorát a kód tisztán tartása érdekénben (4. ábra).



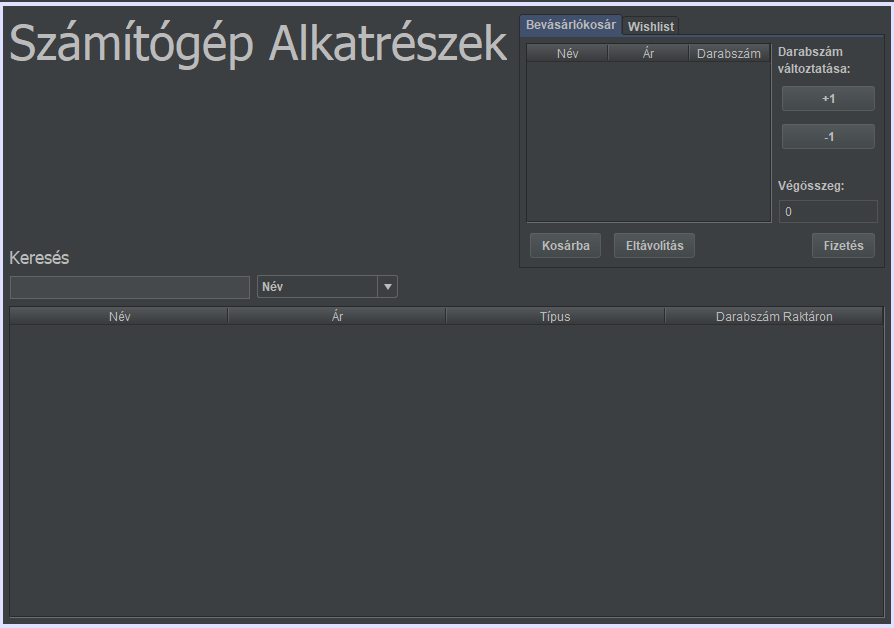
3. ábra: az egyik betöltő függvény.

Az ábrán szereplő betöltő függvény a termékek betöltéséért felel, melyet úgy visz véghez, hogy egy ideiglenes Map-be betölti és azok elemeit hozzáadja a MainController main függvényében a fő ablakban szereplő productsTable táblázathoz.

4. ábra: a MainController-beli main függvény amelyben a fő ablak példányosítása, a fájlokból való betöltés és a garbage collection történik.

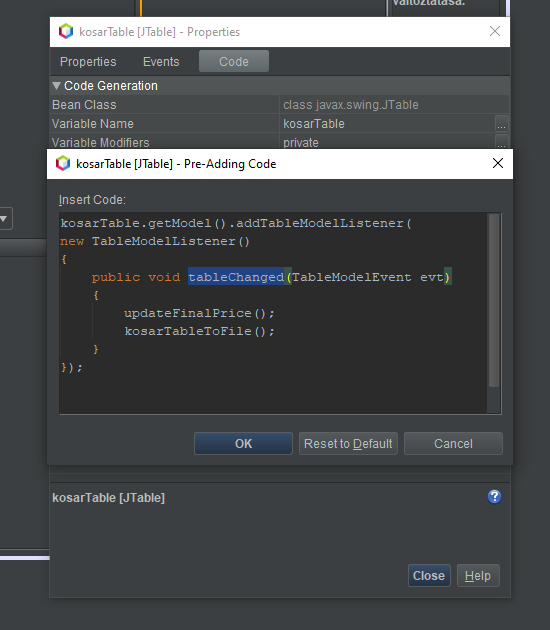
## MainWindow osztály

Ez az osztály az, amellyel a felhasználó elsőként találkozik. Az osztályban lévő JFrame objektum és a benne található grafikus elemek a program és a felhasználó közötti könnyű kommunikációt teszik lehetővé. Itt található az összes terméket egybegyűjtő táblázat, keresési mező, valamint az egymást fedő kosár és wishlist, saját funkcióikkal beágyazva (5. ábra).



5.ábra: a MainWindow osztály grafikus felülete

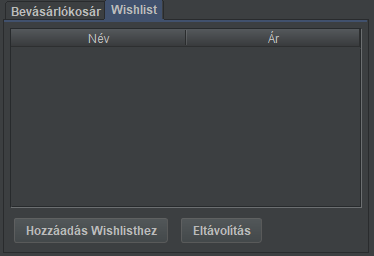
A keresési mező egy lenyíló lista alapján dönti el, hogy mely típus elemeit hasonlítsa össze a keresési mező szövegével, ahol egy keyReleased event futtatja le a keresést, ezáltal biztosítva hogy minden gomblenyomás után történjen meg a művelet. A kosárba a fő táblázatban kijelölt elemet a bevásárlókosár “Kosárba” felíratú gombja teszi, a kosár táblázat pedig minden változtatás során egy tableChanged event alapján (6. ábra) frissíti a végösszeg mezőt.



6. ábra: a kosarTable tableChanged eventje

A wish list - hasonlóan a bevásárló kosárhoz - hozzáadás és törlés funkciókkal rendelkezik. Feladata, hogy eltárolja és elmentse azokat a termékeket, amelyeket a vásárló nem kíván azonnal megvásárolni, de később lehet, hogy igen (7. ábra)

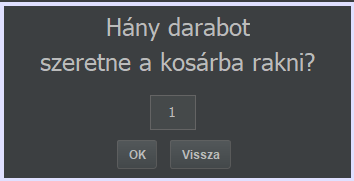
A kosarat és a wish listet egy tabbed pane tárolóba ágyaztam, hogy egymáshoz közel, és könnyen elérhetőek legyenek, valamint, hogy a felhasználónak minél kevesebb külön ablakot kelljen számontartania.



7. ábra: a wish list és funkciói

## SetAmountWindow Osztály

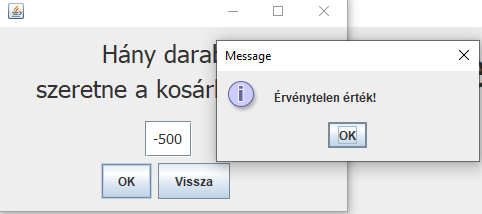
Ez az osztály az előzőhöz hasonlóan egy JFrame objektumot tartalmaz, feladata a MainWindow osztályban található priceTable táblázatban kijelölt termék megvásárolni kívánt darabszámának beállítása és a termék kosárhoz adása (8. ábra).



8. ábra: a SetAmountWindow

A beviteli mező ellenőrzi, hogy a felhasználó csakis 0-nál nagyobb és a tárolt termék darabszámánál kisebb értéket írhasson a mezőbe, ha ennek nem tesz eleget, a program tájékoztatja a felhasználót a problémáról (9. ábra). A “vissza” gomb megnyomásával a művelet visszavonható.

## PayWindow Osztály



9. ábra: példa egy érvénytelen értékre

A fizetés lebonyolításához az osztály bekéri a felhasználó bankkártyaadatait, és a megfelelő mezők beírása közben ellenőrzik le, hogy beléjük megfelelő típusú érték került-e. Ha nem, erről tájékoztatják a felhasználót, és kitörlik a hibás karaktert. A bankkártyasztám beviteli mezeje ezen kívül minden negyedik szám után automatikusan beilleszt egy szünetet, hogy a bankkártyaszámok megjelenítési formája megfelelően legyen tükrözve (10. ábra).

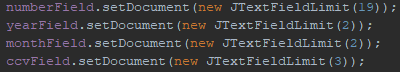


10. ábra: a bankkártyaszám beviteli mező

A beviteli mezők rendelkeznek maximum karakterszámmal, amit az utolsó osztály segítségével valósítanak meg. Az ablak legalján egy szerkeszthetetlen szöveges mező tájékoztatja a felhasználót a fizetendő összegről. Az adatok megadása után, ha minden érték megfelelő, a fizetés gombra kattintva a szoftver “kézbesíti” a kosár termékeit a felhasználó számára egy külön fájlba. Ez az egy fájl tárolja minden felhasználó megrendeléseit.

## JTextFieldLimit Osztály

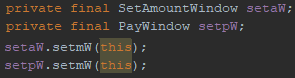
Ez az osztály nem rendelkezik bonyolult feladattal: a PayWindow osztályban szereplő beviteli mezők maximális karakterszámának megadására használtam fel. A PlainDocument ősosztály gyerek osztályaként, hozzá lehet adni bármely beviteli mezőhöz, a setDocument parancsot felhasználva (11. ábra).



11. ábra: a PayWindow beviteli mezői megkapják a maximális méretüket

# Kapcsolat az ablakok osztályai között

Nagy fejtörést okozott számomra, hogy hogyan oldjam meg a kommunikációt a program két segéd ablakai és a MainWindow között anélkül, hogy minden esetben új példányokat hozzak létre az osztályokból. Mivel, getterek és setterek nem voltak elegendők ezért a PayWindow és SetAmountWindow osztályba elhelyeztem egy referenciát a MainWindow osztályból. Ezzel azt értem el, hogy új ablakok példányosítása helyett a MainWindow osztályban a másik két ablakot egyszer példányosítom és paraméterben megadom a MainWindow példányt a két ablaknak (12. ábra).



12. ábra: a segéd ablakok példányosítása

# Összegzés

A feladat egy meglepően más nézőpontot és megoldásmenetet igényelt, mint az eddigi feladatok, és felfrissítő volt a grafikus felület világába belekóstolni. A munkám során rengeteg új tapasztalattal gyarapodtam, amely lehetővé tette a számomra, hogy a jövőben magabiztosan állhassak hozzá a Java grafikus felületének implementálásához.