Sári Ákos - Album nyilvántartás

Adatszerkezetek, tervezési megfontolások

Az alábbi dokumentumban azt fejtem ki, hogy melyik osztály, és leginkább az azon belüli adatszerkezetek mire valóak, mit tárolnak, és hogy miért azt választottam a feladathoz.

Az osztályok, és azok metódusainak bővebb leírása megtalálható a generált javadoc oldalon, illetve szabványos javadoc kommentek formájában a forráskódon belül.

- **Album**: Szerializálható osztály (megvalósítja a Serializable interfacet), mely egy albumot reprezentál. Attribútumai az album 4 alapparamétere (előadó, cím, év, kiadó), valamint ezen kívül egy ID, azaz egy egyedi azonosító (egész szám), mely minden albumnál különböző. Ezt az ID-t szükséges akkor beírnia a felhasználónak, mikor egy albumot szeretne törölni a nyilvántartásból. Ez az osztály tartalmaz az alap gettereken és settereken felül egy matchesSearch metódust, mely egy adott kulcsszó alapján ellenőrzi, hogy az album megfelel-e a keresésnek vagy sem, és ezalapján tér vissza. Ez az osztály azért lett szerializálható, mert szerializálás segítségével történik a fájlba mentés, és deszerializálással történik a fájlból történő beolvasás.
- **AlbumLista**: Egy LinkedList<Album>, tehát egy album objektumokat tároló LinkedListből származtatott osztály, melyet az albumok tárolására használtam. Azért származtattam le így ebből, mert ilyen módon a LinkedList alapértelmezett metódusai, mint a size(), vagy az add() már adva voltak, és fel lehetett őket használni. ArrayList és LinkedList között hezitáltam, és a döntésben némi utánanézés segített. Azt olvastam, hogy az ArrayList hatékonyabb, gyorsabb olyan alkalmazásokban, ahol a keresés a gyakoribb művelet, a LinkedList viszont jobban teljesít, ha magasabb a hozzáadás/eltávolítás műveletek száma. Miután a programomban átlagban nagyobb rendszerességgel fordulnak elő ilyen műveletek, mint a keresés, így a LinkedList-re esett a választásom.
- **AlbumTablazat***: Ez az osztály használatos arra, hogy megjelenítse az összes albumot a nyilvántartásból Jtable segítségével. Az itt megjelenő adatokat módosítani is lehet (az ID-t kivéve), amennyiben duplán kattint a felhasználó egy cellára. Ezt a műveletet az alábbi osztály segítségével végzi.
- **AlbumTableModel**: DefaultTableModelből származó osztály, melynek felelőssége, hogy csak a megfelelő, tehát az ID-tól különböző oszlopban lévő cellák legyenek módosíthatóak, valamint hogy a módosítás ne csak a

táblázatban, hanem a nyilvántartásban is módosuljon, és hogy az oszlopok és sorok mind rendeltetésszerűen jelenjenek meg. Azért esett a DefaultTableModelre a választásom származtatásnál, mert miután utánanéztem, arra jutottam, hogy az AbstractTableModelben mindent magamnak kéne implementálni, addig a DefaultTableModelben elég felülírni bizonyos metódusokat ahhoz, hogy a programom által kívánt viselkedés megfelelően megvalósuljon.

- **BetoltesAblak***: Megjelenít egy fájl kiválasztó ablakot (JFileChooser), ahol egy megfelelő, .alb kiterjesztésű fájlt választhat ki a felhasználó. Ezekből a fájlokból kiválasztás után deszerializálódnak az adatok, és betöltődnek a nyilvántartásba. Azért választottam JfileChoosert ehhez, mert feleslegesnek gondoltam magamnak implementálni valamilyen fájlkiválasztó ablakot vagy más megoldást, amikor ez egyszerűen használható és megvalósítható, átlátható és felhasználóbarát.
- **FigyelmeztetesAblak***: Egy egyszerű, felhasználó valamilyen okból történő értesítésére szolgáló ablakot reprezentáló osztály. Futásidőben többször előkerül, és értesíti az alkalmazót különböző hibákról, vagy történésekről. Ilyen például, mikor nem szám lett beírva kiadási évhez, vagy üresen maradtak hozzáadásnál bizonyos mezők, hibás azonosítószám került beírásra törlésnél, satöbbi.
- **Fomenu***: A főmenüt reprezentáló osztály, a program elindításakor ezzel találja magát szembe a felhasználó. Az ezen lévő menüpontok segítségével lehet navigálni a program funckiói között.
- **FomenuListener**: A főmenün megjelenő gombokra irányított ActionListener interfacet megvalósító osztály. Ez nyitja meg a gombokhoz tartozó funkciók ablakait. A többi Jframe osztállyal ellentétben ezt az átláthatóság érdekében nem a Fomenu belső osztályaként implementáltam.
- **HozzaadasAblak***: Új album hozzáadására szolgáló ablakot reprezentáló osztály. A 4 paramétert 4 szövegdobozban kell megadni a felhasználónak. Üresen nem maradhat egyik sem, és értelemszerűen csak szám adható meg évszámként. Amennyiben nem így van, arról figyelmeztetést kap a felhasználó.
- **KeresesAblak***: A keresőablakot megjelenítő ablakot reprezentáló osztály. Ide írhat a felhasználó keresési kulcsszót vagy időintervalumot. Üresen nem hagyhatja a szövegdobozt ebben az esetben sem. A találatokat az alábbi osztály segítségével láthatja a felhasználó.
- **KeresesiEredmenyek***: A keresett kulcsszónak vagy időintervalumnak megfelelő találatok megjelenítésére szolgáló ablak. A találatok táblázatban (Jtable) jelennek meg, amennyiben vannak. Ha nincs találat, nem jelenik meg táblázat, helyette egy figyelmeztető ablak tudatja a felhasználóval ezt.
- **Main**: A főmenüt megjelenítő osztály, a program belépési pontja.

- **MentesAblak***: A BetoltesAblak-hoz hasonlóan egy JFileChooser típusú ablakot megjelenítő osztály, csak ebben az esetben a nyilvántartást szerializáljuk egy .alb kiterjesztésű fájlba, amiből később szükség esetén betölthető a programba a nyilvántartás. Ugyanabból a megfontolásból használtam itt is JfileChoosert, mint a betöltésnél.
- **TorlesAblak***: A törlésre szolgáló ablakot reprezentáló osztály. Itt a törölni kívánt album ID-jét kell megadni a törléshez. Amennyiben invalid azonosító lett megadva, arról figyelmeztetést kap a felhasználó, illetve arról is, ha sikeres volt a törlés.

  A csillaggal jelölt osztályok mind JFrame osztályból származtatott osztályok, hisz mindegyik egy ablak megjelenítésére szolgál/azt reprezentál. Azok, amik nem JFileChooserek, és tartalmaznak gombo(ka)t, ActionListener interfacet megvalósító belső privát osztályokat tartalmaznak annak érdekében, hogy lekezeljék a gombnyomásokat. (Kivétel ezalól a Fomenu, aminek a korábban leírtak alapján nem belső privát osztály végzi a gombnyomások figyelését, hanem a FomenuListener osztály.)

  A forráskódot nézegetve megfigyelhető, hogy a Mainen kívül minden osztály tartalmaz egy serialVersionUID statikus privát attribútumot, generált értékkel. Ezeket az Eclipse IDE generálta, és annak „javaslatára” adtam hozzá az osztályokhoz. Egész egyszerűen elegánsabbnak gondoltam ezt a megoldást annál, mintha SurpressWarningot alkalmaztam volna ellene.