ZH1

1. Feladat - Zenei albumok katalógusa (40%)

A mellékelt pontosvesszővel tagolt album_catalog.csv fájl egy zenei kiskereskedés katalógusát tartalmazza, a megvásárolható könnyűzenei albumokkal. Egy sor egy kiadásnak felel meg, de egy adott együttes adott albumának több kiadása is előfordulhat. Írjunk egy java programot, amely segít a kereskedőnek rendszerezni a katalógus tartalmát!
A program beolvassa a katalógus fájlt, és eltárolja a benne lévő adatokat. A kiadásokat egy-egy objektum reprezentálja, melynek mezői tárolják a katalógusban az albumokról fellelhető információt (ezek sorrendben: katalógus azonosító, együttes neve, album címe, címke, formátum, értékelés, kiadás dátuma valamint egy egyedi release ID szám.) Megjegyzés: bizonyos kiadásokhoz nem áll rendelkezésre a kiadás éve, ez esetben a fájl megfelelő sorának megfelelő oszlopában 0 szerepel.

- A kereskedő szerint a katalógusban vannak ismétlődések. Az azonos releaseID-vel rendelkező sorokat csak egyszer adjuk hozzá a gyűjteményhez!
- Egy kereskedőpartner szeretne egy listát kapni a legjobb albumokról. Minden kiadáshoz tartozik egy értékelés, amely egy 1 és 10 közötti egész szám. Írjunk egy olyan függvényt, amely egy integerre visszaadja a legalább akkora értékelést kapott kiadások listáját! Írjunk egy függvényt, amely egy együttes nevére visszaadja az albumainak listáját!
- Írjunk egy másik függvényt, amely az együttes és az album nevére visszaadja az album kiadásainak listáját.
- A kereskedő szeretné, ha a katalógusa sorrendben lenne. Hozzunk létre egy olyan csv fájlt az eredetivel analóg módon, amelyben az együttesek abc sorrendben szerepelnek, az együttesen belül az album címek is abc sorrendben, az azonos albumok pedig a kiadás éve szerint növekvő sorrendben. Ha a felsorolt három adat mind egyezik, akkor a releaseID döntsön.

Készíts egy konzolos menüt, amelyből a megírt függvények meghívhatók.

2. Feladat - Reptér (60%)

Ebben a feladatban egy kisebb reptér működését kell szimulálni.
A reptéren egy kifutópálya van, ahonnan a gépek felszállnak, és ugyanide érkeznek is. Ezen kívül a reptérnek van 10 dokkolója, ahol a leszállt repülők várakozhatnak. A repülőket a példánkban szálak fogják reprezentálni.
A reptérre random időközönként, átlagosan másodpercenként egy repülő érkezik, ami megnézi, hogy szabad-e a leszállópálya ÉS van-e szabad dokkoló. Ha igen, akkor a repülőgép nagyon gyorsan, 0.3 másodperc alatt leszáll. Miután leszállt a már korábban lefoglalt dokkolóba megy, majd szabaddá teszi a leszállópályát. A repülők addig köröznek a reptér körül, amíg nem sikerül leszállniuk és 0.15 másodpercenként újrapróbálkoznak.
A repülőgépek a reptéren 5 másodpercet töltenek ki- és berakodással. Ezután ismét megnézik, hogy szabad-e a felszállópálya és a repülőgép 0.3 másodperc alatt felszáll. Ha nem sikerül felszállniuk akkor 0.15 másodperc múlva újrapróbálkoznak. Miután felemelkedett egy repülő megszakítja a kapcsolatot a reptérrel.
A feladat megoldása során ne használj busy waitinget, használd helyette a Java nyelv beépített lehetőségeit, amit a gyakorlaton is használtunk. A feladat megoldásakor figyelj arra, hogy a programod szálbiztos legyen!
A program folyamatosan naplózza a gépek tevékenységeit, például:

- 08:30:11 A(z) 59. gép leszállt és elfoglalta a(z) 3. dokkolót.
- 08:30:14 A(z) 45. gép elhagyta a(z) 5. dokkolót és felszállt.

- 08:30:14 A(z) 60. gép nem tudott leszállni.
- 08:30:15 A(z) 61. gép leszállt és elfoglalta a(z) 8. dokkolót.
- 08:30:15 A(z) 60. gép nem tudott leszállni.
- 08:30:15 A(z) 59. gép nem tudott felszállni.
- 08:30:16 A(z) 59. gép elhagyta a(z) 3. dokkolót és felszállt.
- 08:30:16 A(z) 60. gép nem tudott leszállni.
- stb.