Haladó Java ZH, 2023.06.12.

Feltételek

- A feladat megoldását önállóan, más segítsége nélkül kell elkészíteni.
  - Kommunikáció csak az oktatókkal megengedett.
  - Az elkészített megoldást nemcsak a ZH végéig, hanem egészen a ZH napjának végéig nem szabad megosztani mással (pl. fórumba vagy publikus verziókezelő rendszerbe felöltés).
  - A megoldás elkészítéséhez használható a Java API és a JUnit dokumentációja. Ezek a Canvasből letölthetők, kicsomagolhatók.
- Az elkészített megoldást **zip** formátumba csomagolva kell feltölteni a Canvasbe.
  - A `zip` tartalmazza a forrásfájlokat a megfelelő szerkezetben.
    - A megkapott `jar` fájlokat nem kell beadni.
  - A **ZH végén kb. 10 percet érdemes fenntartani** a kód tisztázására, fordíthatóvá tételére, tömörítésére, beküldésére.

1. feladat - 8 pont

A `linked.txt` egy szöveget tartalmaz láncolt ábrázolással: az első byte az első szövegdarab hosszát tartalmazza byte-ban, majd maga a szövegdarab következik, végül pedig a következő szövegdarab pozíciója 2 byte-on `low-endian` formátumban ábrázolva. A többi szövegdarab is leírása is hasonló szerkezetű. Az utolsó szövegdarab végén `-1` található, mivel annak nincs rákövetkezője. A feladatod, hogy a `java.nio` csomagbeli osztályok segítségével írd ki a szövegdarabokat helyes sorrendben.

2. feladat - 12 pont

Adott a `Malev2012Summer.ssim` nevű adatállomány, amely a 2012-ben csődbe ment Malév azon év nyári időszakára tervezett menetrendjét tartalmazza SSIM formátumban. A formátumról a következőket kell tudni a feladat megoldásához:

- Az első karakter `3` vagy `4`, ezek közül a `3`-mal kezdődő sorok tartalmazzák a járatok elsődleges adatait.
- A kiinduló állomás `3` betűs IATA kódját a `36`-`38` oszlopok tartalmazzák.
- A célállomás `3` betűs kódját a `54`-`56` oszlopok tartalmazzák.
- Az indulási idő (`2` számjegy az óra, `2` számjegy a perc) a `39`-`42` oszlopokban található az UTC időzóna szerint.
- Az érkezési idő (`2` számjegy az óra, `2` számjegy a perc) a `57`-`60` oszlopokban található az UTC időzóna szerint.

Célállomások listája - 5 pont

Készíts egy statikus függvényt egy tetszőleges osztályban, amely visszaadja az összes célállomást, ahová a Malév a 2012-es nyári időszakban repülni tervezett. A visszaadott lista betűrendben tartalmazza az összes Budapestről (`BUD`) célállomás IATA kódját, `BUD` nélkül, és természetesen minden kódot egyetlen egyszer. (A következő részfeladat miatt az érkező repülőtér legyen `BUD`, a listában pedig a kiinduló repülőterek kódjai szerepeljenek.) Kizárólag stream-eket, lambdákat és collector-okat használj a feladat megoldásához.

Adott idő alatt elérhető célállomások listája - 4 pont

Bővítsd az előző részfeladatban megírt statikus függvényt egy egész típusú paraméterrel, amelyben a függvény hívója megadhatja, hogy hány percen belül elérhető célállomások listájára kíváncsi. Szintén stream-ekkel és lambdákkal oldd meg a feladatot.

- Segítség: időadatok összehasonlításához használhatod a (Java sztenderd könyvtárában megtalálható) `ChronoUnit` osztály `between()` metódusát. Ehhez a karakterláncként megadott időadatokat `LocalTime`-má kell alakítani, amihez a `DateTimeFormatter` segítségét célszerű igénybe venni.
- Figyelem: a menetrendben vannak olyan járatok is, amelyek Budapestről éjfél előtt indulnak, de a célállomásra éjfél után érkeznek. Éppen ezért ne ezeket, hanem a Budapestre érkezőket vedd figyelembe, mert ezek mindig az indulás napján érkeznek.

Teszt - 3 pont

Teszteld a megoldásodat JUnit 5 Juniper segítségével. Használj CSV paraméterezett tesztet, amelyben célállomásokból és időadatokból álló párokat adsz meg, legalább hármat.

3. feladat - 20+4 pont

Annotáció bináris műveletek tulajdonságaira - 10 pont

Készíts el egy `BinaryPropertyType` nevű felsorolási típust, amely a következő értékeket valamelyikét veheti fel: `COMMUTATIVE`, `NONCOMMUTATIVE`, `ASSOCIATIVE` és `NONASSOCIATIVE`. A metódusokra alkalmazható `@BinaryProperty` nevű annotáció paramétere egy ilyen elem, alapértelmezése nincs.

Készíts továbbá egy `BinaryPropertiesCheck` osztályt egy publikus, osztályszintű `checkBinaryProperties` sablonmetódussal. A metódus paraméterként megkapja az osztály hivatkozását, és `3` azonos típusú paramétert.

- A metódus pontosan akkor adjon vissza igazat, ha az osztály összes metódusára, amely el van látva a `@BinaryProperty` annotációval, teljesül a feltétel.

    - Példa: ha a vizsgált osztály `f` metódusa `@BinaryProperty(ASSOCIATIVE)` annotációval van ellátva, és a `checkBinaryProperties` három paramétere `1`, `2` és `3`, akkor megvizsgálandó, hogy `f(f(1, 2), 3)` és `f(1, f(2, 3))` értéke megegyező-e.
    - Minden ilyen `f` metódusról feltételezhető, hogy osztályszintű.
    - A metódust az önelemzés `invoke` hívásával kell működtetni. Itt fontos, hogy az első paraméter `null` legyen, mert a meghívott `f` nem példányszintű.
- Csakis azokat a metódusokat kell figyelembe venni, amelyeknek pontosan két paramétere van, és ezek, valamint a visszatérési érték típusa is megegyezik a `checkBinaryProperties` sablonparaméterével.
- Ez a metódus akkor ér teljes pontot, ha (a lehetőségekhez képest a lehető leginkább) folyam alapú a megoldás kódja.
- Segítség: a generikus metódus paramétereinek típusa akkor is `Integer` lesz, ha `int` típusú értékekre hívod. Ekkor a metódusok `int` típusú paraméterei különbözőnek fognak látszani.
    - Ennek elkerülésére használd az `org.apache.commons.lang3.ClassUtils.isAssignable(Class, Class, boolean)` metódust, amelynek utolsó paraméterét `true`-ra állítva kompatibilisnek látod a primitív és az őket becsomagoló típusokat.
    - A szükséges `JAR` állományok a **letölthető fájlban találhatók**.

Többszörözhető annotáció - 5 pont

A `@Repeatable` metaannotáció használatával oldd meg, hogy a `@BinaryProperty` annotáció többszörözhető legyen. Egy metódus ugyanis lehet például egyszerre kommutatív és asszociatív.

Tesztelés - 5 pont

Írj JUnit 5 tesztet, amelyben legalább három tesztosztályon teszteled a megoldásodat.

- A egyikben minden metódus legyen helyesen annotálva, míg a másik kettőben legyen legalább egy hiba, de azok különbözők legyenek.
    - A metódusok mind osztályszintűek legyenek.
- Három általad választott típusú és értékű paraméterrel hívd a teszteket.

Bónusz: tulajdonság alapú tesztelés - 4 pont

Írj olyan tesztet is, ahol a tesztelő rendszer a `QuickCheck` eszközre bízza a három paraméter megválasztását egy adott tartományból.

A `QuickCheck` használatához szükséges `JAR` állományok és a működését leíró HTML oldal szintén a **letölthető fájlban vannak**.