Java lambda labor - IMSC

Készítette: Goldschmidt Balázs, BME IIT, 2020.

Bevezető

A Java nyelvben a lambdával karöltve megjelentek a kollekciók *stream*-jei. A *stream*-eket tekinthetjük a C++-os *iterátorral határolt intervallumok* Java-s megfelelőinek.

https://docs.oracle.com/javase/8/docs/api/java/util/stream/Stream.html

Minden *Collection*-ből lehet *stream*-et kérni (*Collection.stream()*), ami az elemek összességét reprezentálja, és amin különféle műveleteket tudunk végezni. A műveletek eredménye legtöbbször egy új *stream* lesz, az eredeti kollekció pedig érintetlen marad. A műveletek kombinálásával tudunk az elemekből különféle szűréseket, számolásokat, rendezéseket kialakítani. A *stream*-en végezhető műveletek alapvetően 3 csoportba oszthatók:

- szűrő (filtering) nem módosítja az elemeket, de nem minden elemet enged át, pl. komplex számokból csak az egységkörön belülieket engedi tovább. *Stream.filter*
- átalakító (mapping): az egyes elemeken végez valami transzformációt, pl. komplex számokról levágja a képzetes részt. *Stream.map*
- redukáló (reducing): az elemek összességén végez valamit, pl. átlagot számol. *Stream.max, Stream.min, Stream.reduce*
- lekérdező (behaviour): valamit lekérdez, de az elemek nem változnak, pl. kiíratjuk az elemeket. *Stream.forEach, Stream.forEachOrdered*

A fenti műveleteket összefoglalóan *filter-map-reduce* (vagy csak röviden *map-reduce*) műveleteknek is nevezzük.

Egyszerű példák:

```
// hallgatók listája
List<Student> l = …;
//kiíratunk mindenkit
l.stream().forEach(System.out::println);
//megszámoljuk a 3.5-nél jobbakat
long cnt = l.stream().filter(s-> (s.getAverage()>3.5)).count();
```

A *stream*-ek gyakorlásához az eddigi sörnyilvántartó alkalmazásunkat piszkáljuk tovább. A meglevő listázó, szűrő metódusokat (*list*, *search*, *find*, *remove*) fogjuk *stream*-ek segítségével átalakítani.

1 Listázás stream-ekkel

A meglevő, sokat szenvedett *list* parancsunkat boncolgatjuk tovább. A *list* jelen állapotában 3 részfeladatot old meg:

i. paraméterek feldolgozása

ii. kombinált komparátor előállítása

iii. rendezett lista kiíratása

A fenti felsorolás elemeit igyekszünk lambda és stream alapon újraírni.

a) Módosítsuk az elemek rendezését és kiírását úgy, hogy (a *Collections.sort* és a meglevő ciklus elhagyásával) a sörök listájából *stream*-et kérünk, és ezen végezzük a rendezést és az elemenkénti kiíratást.

b) A komparátor összeállítását is végeztessük streammel! Próbáljuk a ciklust úgy átalakítani, hogy egy stream-en a *map* és *reduce* metódusokkal elvégezhető legyen a komparátorok összeállítása. A *reduce* végeredményéből *get()* metódussal tudjuk a várt komparátort kinyerni.

c) A paraméterek feldolgozását is próbáljuk streammel megoldani! Készítsünk egy *List*-et a cmd tömbből (*Arrays.asList*), és fordítsuk meg (*Collections.reverse*). Az eredményen hívhatunk *stream*-et és ezen hívjuk a *forEachOrdered* metódust.

   Itt sajnos a meglevő ciklusban levő feltételt és kétlépéses műveletet olyan összetett lambda-kifejezéssé kell alakítanunk, ami az eddigiekhez képest nem tűnik szépnek.

Az utolsó részfeladat mutatja, hogy vannak esetek, amikor egyszerűbb a hagyományos ciklusos megoldást alalkalmazni, mint a streammel erőlködni. Mi ennek az oka?

Tippek:

a) Használjuk a *sorted* és a *forEachOrdered* metódusokat!

b) A *map* metódusban megadandó kifejezésben a *comps* hashmapet használjuk.

   A *reduce* metódusban egy paraméterre lesz szükségünk, ami két komparátorból egyet csinál:

```
(cmp1,cmp2)->(cmp1.thenComparing(cmp2))
```

2 Szűrések streammel

A listázás után vegyük sorra, hogy a *search* és *find* parancsok hogyan valósíthatók meg streamek segítségével!

a) Alakítsuk át a *search* parancsot!

   Érdemes inkrementálisan haladni.

   Először alakítsuk az egyes *if*-ek belsejét *stream*-essé! Vegyük észre, hogy nagyon sok azonos kódelem szerepel, lényegében csak a *filter* metódus paramétere különbözik.

   Emeljük ki a filter paraméterét (*Predicate<Beer>* típus), csak ennek a beállítása történjen az *if*-ek törzsében, az érdemi streames kód az *if*-ek után fusson le, ahol a *filter* a feljebb beállított predikátumot kapja paraméterül.

b) A *search*-höz hasonlóan alakítsuk át a *find*-ot is. Vegyük észre, hogy a *find* parancs implementációja lényegében csak a predikátumaiban különbözik a *search*-től. Csodáljuk meg, hogy az így kapott kód mennyivel olvashatóbb és áttekinthetőbb, mint a korábbi. A predikátum kiemelése egyértelművé teszi, hogy mi alapján szűrünk, a közös kód pedig azt, hogy mi is az alapalgoritmus.

3 Új parancs megvalósítása

Vegyünk fel egy új parancsot (*stats*), ami különféle statisztikákat gyűjt és ír ki.

a) Hozzunk létre új metódust vagy osztályt az eddigi parancs-megvalósításnak megfelelően. A metódus írja ki, hogy "hello world".

   Ne felejtsük, hogy a "*stats*"-ot a parancs-listába is betegyük.

   Ha ez megvan, próbáljuk ki. A parancsfeldolgozásba elvileg nem kellett belenyúlnunk.

b) Az első statisztika legyen az, hogy hány db sörünk van az egyes sörfajták esetén! Ennek eredménye legyen egy *Map<String, Long>*-ba gyűjtve.

c) Írassuk ki a kigyűjtött adatokat *stream* segítségével.

d) A második statisztika legyen az egyes sörfajták átlagos erőssége. Ezt egy *Map<String, Double>*-be gyűjtsük ki.

e) Írassuk ki a kigyűjtött eredmény tartalmát is.

Tippek

b) Használjuk a Stream osztály collect metódusát!

A collect metódusnak paraméterként a Collectors osztály groupingBy metódusával készíthetünk megfelelő collectort.

A groupingBy-t két paraméterrel kell ellátni: az egyik a Map kulcsát állítja elő (egy sörből kiveszi a fajtát), a másik egy számolgató rutin, amit a Collectors.counting() valósít meg legegyszerűbben.

c) A Map-en az entrySet-et hívjuk meg, és ezen készítsünk stream-et, majd az elemeken olyan forEach-et, ami szépen kiíratja ez egyes String-Long párokat.

d) Hasonlóan a fenti gyűjtéshez, a collect-et kell használni. Ennek második paramétere most legyen a Collectors.averagingDouble-metódus visszatérési értéke, ebben kell megadni, hogy az egyes sörök melyik mezőjén szeretnénk a stílusonkénti átlagképzést végezni.

e) Mint az előző kiíratásnál. :)