OOP 10. nädal IDK0051, sügis 2015

Tänases praktikumis uurime *lambda expressioneid*. Eesmärk ei ole luua ühtset terviklikku süsteemi, vaid proovida läbi mitmeid erinevaid võimalusi ja uurida, kuidas lambdad toimivad.

Tuletage loengust meelde, et lambda on sisuliselt anonüümne (nimeta) meetod. Kiire sissejuhatuse teemasse leiate ka siit:

http://zeroturnaround.com/rebellabs/java-8-explained-applying-lambdas-to-java-collections/

1. Ettevalmistus

Lambdad on kõige kasulikumad, kui peate tegelema keeruliste objektide kollektsiooniga. Kasutame kõige lihtsamat kollektsiooni tüüpi – listi. Objektitüübiks valime Studenti. Aga esmalt looge endale vastav liides (interface), mille alusel Student objekte luua:

```
package student;

public interface StudentModel {
	// sets some arbitrary value for grade
	public void setGrade(int grade);
	// return the value of grade
	public int getGrade();
	// return name of the student
	public String getName();
	// return age of the student
	public int getAge();
	// return how many credit points s/he has earned
	public int getEarnedCreditPoints();
	// return number of credit points required to graduate
	public int getNominalCreditPoints();
}
```

ja realiseerige see konkreetses klassis Student.

2. Looge testklass ja seal *main* meetodis kümmekond erinevate omadustega tudengit.
	- pooled tudengid peaksid olema läbinud vähemalt 60EAP ja pooled vähem (getEarnedCreditPoints() väärtus tagastab läbitud EAP arvu). Lisage need listi.

3. Esmalt proovime Java 8 kollektsioonide uut meetodit forEach – itereerimine on viidud väliselt foreach tsüklilt sisemisele, argumendiks on lambda expression:

Väline itereerimine	***Sisemine itereerimine***

`for (Car c : cars) {`
`  c.setSpeed(40);`
`}`
	`cars.forEach(c -> c.setSpeed(40));`

Need näited on identsed. Realiseerige mõlemad iteratsioonid ilma copy-pasteta oma tudengite kollektsiooni kohta, seadistades igale tudengile mingi hinde meetodiga setGrade().

Kollektsiooni forEach meetod on kasulik, kui soovime kollektsiooni itereerida ja tegeleda iga üksiku elemendiga. Kui soovime teha operatsiooni kollektsiooni kui tervikuga, siis loome voo (stream) ja kasutame sealseid meetodeid. NB! Tuletage loengust meelde eager ja lazy meetodite erinevus.

4. Looge oma kollektsioonist voog ja filtreerige see nende näidete eeskujul:

```
Stream<Car> myStream = cars.stream();
```

filtreerime kollased autod:

```
Stream<Car> yellowCars =
                myStream.filter(c -> c.getColor()
                      .equals("yellow"));
```

leiame, mitu kollast autot oli:

```
long yellowCarCount = yellowCars.count();
```

Realiseerige sama funktsionaalsus tudengite kohta, arvutades, mitu tudengit on sellist, kes on läbinud vähemalt 60 ainepunkti. NB! Kas voogu saab korduvkasutada?

5. Kasutades *method chainingut*, kirjutage sama näide nüüd ühe käsuna:

```
Stream<Car> yellowCars = cars.stream()
                           .filter(c -> c.getColor().equals("yellow"));
```

See näide ei ole täielik – töötage välja versioon, kus tagastatakse kohe 60 EAP või rohkem teeninud tudengite arv.

6. Looge nüüd uus näide eelmist muutes nõnda, et tudengite arvu asemel tagastate kõik tudengid, kes on teeninud vähemalt 60EAP listina, kasutades meetodit:

```
.collect(Collectors.toList());
```

7. Nüüd soovite, et tagastatakse kindel Listi alamtüüp LinkedList. Kasutage meetodit, mis vajab viidet teisele meetodile, antud juhul listi konstruktorile. Kasutage viitena `LinkedList::new` ja meetodina:

```
      .collect(Collectors.toCollection(method_reference));
```

8. Nüüd soovite teada, mitu ainepunkti kokku on teeninud tudengid, kes on saanud individuaalselt vähemalt 60EAP (st kui kolm tudengit on saanud vastavalt 60, 62 ja 100 punkti, siis peab vastus olema 222 punkti). Selleks mapime väärtusi – meil ei ole enam tarvis tegeleda Student objektidega. Peale seda kui olema vajalikud Studentid välja filtreerinud, teisendame streami mapToInt() meetodiga numbristreamiks ja summeerime numbrid.

9. Looge kaks Optionali Student objektis – üks tudengiga ja teine tühi ja:

- kasutades ifPresent meetodit ja lambdat, lisage tudengile hinne (setGrade)
- väljastage tudengi nimi ja hinne konsooli kasutades ifPresent meetodit ja meetodi viidet (method reference) – soovitavalt realiseerige selleks Student klassis toString() meetod.