OOP 11. nädala praktikum

TTÜ IDK0051 sügis 2015

Täna vaatame lambdasid teoreetilisema nurga alt ning alternatiivi lambdale – viidet meetodile. Eelmisel nädalal vaatasime, kuidas lambdasid kasutada. Võrdlesime seda foreach tsükliga. Universaalsem kuju on aga:

(argumendid) –> (lambda keha)

Sellest üldkujust saab kergesti järeldada, et lambdal võib olla ka mitu argumenti, näiteks:

```
(s, p) -> doSomething(s, p.getValue())
```

Samas kui proovida tavalist streami-meetodit kasutada sellise lambdaga, siis see ei õnnestu.

1. Kasutades eelmise praktikumi koodi alusena, looge tudengite kollektsiooni põhjal uus stream (NB! Ülejäänud ülesanded ei eelda eelmise praktikumi koodi olemasolu):

```
students.stream().filter((s,m) -> s.getGrade());
```

Millised vead saate (üks peaks olema filter() meetodi ja teine getGrade() kohta)?

2. Esimene veateade ütles teile, et olete valet tüüpi lambda filter() meetodi argumendiks kirjutanud. Teine seda, et lambda signatuur ei vasta ootustele. Selgub, et lambdalgi on signatuur ja tüüp, aga kuidas? Esmalt seda uurimegi natuke lihtsama näite varal.

Oletame, et reisibüroo on tellinud teilt infosüsteemi oma sihtkohtade kohta. Igas sihtkohas on temperatuur, kuid kliendid soovivad seda näha erineval skaalal. Seetõttu hoitakse temperatuuri klassis kelvinites ning iga kliendi päring määrab, kas talle tagastatakse see Celsisuse või Fahrenheit skaalal. Looge vastav sihtkoha klass Destination, mis realiseeriks seda liidest:

```
package travel;

import java.util.function.DoubleFunction;

public interface DestinationModel {
	// return name
	public String getName();
	// return temp in Kelvins
	public double getKelvin();
	// return temp using provided converter method
	public double getAvgWeather(DoubleFunction<Double> tempHandler);
}
```

Ilmselt ei teki teil probleeme konstruktori ja getName() meetodi sisu loomisega. getAvgWeather() meetodi signatuuris on aga kahtlane argument tüübiga DoubleFunction<Double>. Tegemist on funktsiooniga – getAvgWeather() meetod ootab oma argumendina viidet funktsioonile.

DoubleFunction kirjelduse leiate siit:
https://docs.oracle.com/javase/8/docs/api/java/util/function/DoubleFunction.html

Oluline on järgmine lõik: „Represents a function that accepts a double-valued argument and produces a result.” Tutvuge sellega, aga meetodi sisu pole vaja hetkel luua.

3. Looge realiseerivas klassis konstruktor, mis võtab argumentidena kohanime ja keskmise temperatuuri kelvinites (double-tüüpi). Salvestage need klassi väljadele.

4. Looge testklass ja seal kolm erinevat sihtkohta, kus on erinev nimi ja temperatuur igaühel.

5. Looge klassi Destination meetod, mis konverteeriks konstruktoris määratud temperatuuri Celsiuse skaalale ja tagastaks selle (looge uus meetod, hetkel jätke getAvgWeather() kõrvale).

http://en.wikipedia.org/wiki/Conversion_of_units_of_temperature#Celsius_.28centigrade.29

6. Looge samasugune meetod, mis konverteeriks temperatuuri Fahrenheiti skaalale

7. Testklassis kutsuge loodud meetodeid välja ja trükkige väljund konsooli.

8. Nüüd on eesmärgiks saavutada sama tulemus ilma loodud meetodeid kasutamata.

1. Kutsuge nüüd testklassist välja getAvgWeather() meetod ning kirjutage talle argumendiks lambda expression, mis teeeb sama asja, mida teie punktis 5 loodud meetod.
2. Kui praegu getAvgWeather() tulemuse väljastate, tagastab ta 0.0, kuid kood kompileerub. Nagu nägite, siis teie lambda expression on tüüpi DoubleFunction ehk selline funktsioon, mis soovib argumendiks double-tüüpi väärtuse ning tagastab teie soovitud väärtuse. Antud juhul märksime, et soovime Double väärtust, kuna:

```
public double getAvgWeather(DoubleFunction<Double> tempHandler);
```

Muutke nüüd getAvgWeather() meetodit nii, et see tagastaks selle lambda abil temperatuuri Celsiuse skaalal, kasutades argumendina saadud lambdat. Millise nimega meetodit välja kutsuda? Lambda pakub meetodi **keha**, aga **nime** annab DoubleFunction. Valik ei ole üleliia suur:

https://docs.oracle.com/javase/8/docs/api/java/util/function/DoubleFunction.html

9. Kirjutage testklassis samasugune lambda-meetod ka Fahrenheit tarvis.

10. Kui te nüüd vaatate uuesti DoubleFunction dokumentatsiooni, siis märkate, et tegemist on *funktsionaalse liidesega* ehk liidesega, millel on ainult üks meetod. Javas on selliseid liideseid olemas üksjagu:

https://docs.oracle.com/javase/8/docs/api/java/util/function/package-summary.html

Neid saab ka ise kirjutada. Kuid... enne kui seda teeme, siis tasub märgata, et sellisele meetodile ei pea argumendiks andma *lambda-expressionit*, vaid võib vabalt anda ka viite olemasolevale meetodile. Viidatud meetodi signatuur peab vastama küsitule.

Lisage klassi Destination tavalised staatilised meetodid temperatuuri konverteerimiseks, mille signatuur vastaks teie loodud lambdade signatuurile (oleksid sama tüüpi argumendid ja return type).

11. Kutsuge nüüd testklassis välja uuesti meetod getAvgWeather, kuid ärge andke argumendiks lambdat, vaid viited äsjaloodud meetoditele. Vihje: samamoodi nagu Optionali puhul andsite meetodi viite kaasa ifPresent() meetodile.