DAT108 Oblig2 h22 – Lambda, strømmer, mm

Sist oppdatert av Lars-Petter Helland 13.09.2022

Innleveringsfrist er søndag 2. oktober. Vi har som mål å rette innleveringene og godkjenne innen 2 uker etter frist.

Øvingen leveres ut i 2 omganger:

- Oppg 1, 2 og 3 leveres ut tirsdag 13. august
- Ny versjon inkl. Oppg 4 leveres ut senest én uke før innlevering

Øvingen skal gjennomføres i grupper på **2-4** studenter.

NB! Det er nytt gruppesett for hver innlevering.
Dere må registrere gruppen på nytt selv om dere er samme gruppe som sist!
Gruppesettet for Oblig2 er **«DAT108 Oblig2 gruppe x».**

Send en melding til lærer (Lars-Petter) om dere ønsker å levere i ener-gruppe eller i gruppe med flere enn 4 studenter. **Vi har som mål at flest mulig av de som leverte alene på Oblig1 nå skal levere sammen med andre.**

Innleveringen er en zip i Canvas.
NB! Zip-en skal hete Oblig2_gr17.zip for gruppe 17 osv (dere forstår) ...
Denne skal inneholde:

1. Et pdf-dokument med:
    a) En liste av hvem som er med i gruppen (for å unngå gratispassasjerer).
    b) Skjermutskrift fra kjøringer av programmene.
    c) Svar på eventuelle teorispørsmål.
2. Løsning på oppgavene: **Ett** Eclipse-Java-prosjekt for lambda- og streams-oppgavene (Oppgave1, 2 og 3), og **ett** Eclipse-JEE-prosjekt for Oppgave4.

Oppgave 1 - Lambda-uttrykk

a)

Klasse med main()-metode: **Oppg1a**

Anta at vi har en liste av heltallsstrenger:

```
List<String> listen = Arrays.asList("10", "1", "20", "110", "21", "12");
```

Oppgaven din er å skrive disse ut på skjermen sortert etter tallverdi, dvs. 1, 10, 12, 20, 21, 110. Du skal bruke en av de innebyggede sort()-metodene i Collections til å gjøre sorteringsjobben. Bruk et lambdauttrykk til å representere en Comparator som sort() kan benytte seg av.

b)

Klasse med main()-metode: **Oppg1b**

Oppgaven din er å lage en metode som utfører en heltalltsberegning på to heltall den mottar som parametre. Hvilken beregning som utføres angis med en tredje parameter. Noe slikt:

```
public static int beregn(int a, int b, ???) {
    ...
}
```

Deretter skal du bruke denne metoden i et par eksempler.

Vis (i main) eksempler på bruk av denne metoden til å beregne:

i. Summen av 12 og 13
ii. Den største av -5 og 3
iii. Avstanden (absoluttverdien av differansen) mellom 54 og 45

Den tredje parameteren ved kall til beregn skal alltid angis som en variabel, f.eks:

```
int sum = beregn(12, 13, summerFunksjon);
```

Tips: BiFunction

Oppgave 2 - Lambda-uttrykk

Klasse med main()-metode: **Oppg2**
Andre: **Ansatt** (klasse), **Kjonn** (enum)

Ansatte har fornavn (String), etternavn (String), kjonn (Kjonn), stilling (String) og aarslonn (int).

Vi tenker at vi har en liste av ansatte og skal lage en metode som utfører lønnsoppgjør (oppdaterer de ansattes lønn etter en viss algoritme).

Denne metoden ligger i Oppg2, og ser slik ut:

```
private static void lonnsoppgjor(List<Ansatt> ansatte, ???) ...
```

Hvordan lønnen endres kan variere, f.eks. :

i. Et fast kronetillegg
ii. Et fast prosenttillegg
iii. Et fast kronetillegg hvis du har lav lønn
iv. Et fast prosenttillegg hvis du er mann

osv...

Vi ønsker at de ulike måtene **den nye lønnen skal beregnes** oppgitt som en funksjonsparameter til metoden lonnsoppgjor(...) slik at lonnsoppgjor(...) blir generell og uavhengig av type beregning.

Skriv en løsning med klassene Oppg2 (med main)), Ansatt og Kjonn.

Bruk lambda-uttrykk (i main) for de ulike typene lønnsøkning, og lag en kjørbar løsning med en liste av 5 ansatte der du tester ut de fire ulike typene lønnsøkning i listen over (og evt. flere hvis du vil). Du må gjerne lage en hjelpemetode i Oppg2 som skriver ut hele listen, f.eks:

```
private static void skrivUtAlle(List<Ansatt> ansatte) ...
```

Tips: La parameteren til lonnsoppgjor(...) være av typen Function<Ansatt, Integer> (tilsv. en f(ansatt) := beregning av ny lønn). Dette tilsvarer om du selv hadde definert en funksjonell kontrakt med metoden Integer beregnNyLonn(Ansatt a). I lønnsoppgjøret settes da den nye lønnen med a.setLonn(<beregning av ny lønn>) for hver av de ansatte.

Oppgave 3 - Streams

Klasse med main()-metode: **Oppg3**

Dere kan bruke Ansatt-klassen og listen av ansatte (kopier over til Oppg3.java) fra forrige oppgave. Ansatt-listen blir input til alle delspørsmålene.

Skriv en klasse Oppg3 med en main()-metode som inneholder løsningen på delspørsmålene.

Løsningene skal være funksjonelle, dvs. bruk stream(), map(), filter(), reduce(), osv. Alle løsningene/svarene skal lagres i variabler som deretter skrives ut på skjermen.

a) Lag en ny liste som kun inneholder etternavnene til de ansatte.

b) Finn ut antall kvinner blant de ansatte.

c) Regn ut gjennomsnittslønnen til kvinnene.

d) Gi alle sjefer (stilling inneholder noe med "sjef") en lønnsøkning på 7% ved å bruke streams direkte i stedet for metoden du laget i Oppg2. Skriv ut listen av ansatte etter lønnsøkningen.

e) Finn ut (true|false) om det er noen ansatte som tjener mer enn 800.000,-

f) Skriv ut alle de ansatte med System.out.println() uten å bruke løkke.

g) Finn den/de ansatte som har lavest lønn.

h) Finn ut summen av alle heltall i [1, 1000> som er delelig med 3 eller 5.

Oppgave 4 – [Denne blir lagt til senere]