Schrijf je code in het Engels.

Gebruik naming conventions!

Labo week 4

Lesdoelen:

- Functioneel programmeren met Lambda
- Werken met streams

Maak voor elk labo een nieuw project aan!

Oefening 1 : The Return of the Bonobo

Download van digitap de labo opgave code en plaats deze in een nieuwe 'zoo' package. In de Main staat een lijst van alle Bonobo's van dierenpark Planckendael.
Opgelet! Deze lijst is aangepast t.o.v. labo 2, neem zeker de nieuwe code!

Los dezelfde vragen als in labo 2 op, maar nu enkel gebruik makende van streams.

1. Zorg ervoor dat de standaard sortering van Bonobo op naam is, sorteer de lijst en toon deze. Controleer of de sortering correct is.

2. Sorteer daarna de lijst op geboortejaar oplopend én indien twee geboortejaren hetzelfde zijn, gebruik de standaard sortering. Pas de standaard sortering niet aan. Sorteer de lijst en controleer of de sortering correct is. **Gebruik geen comparator maar een lambda voor de sorteerfunctie**.

3. Maak aan de hand van de lijst van Bonobo's een nieuwe lijst aan van enkel de mannelijke Bonobo's. Sorteer deze nieuwe lijst op id aflopen en controleer de lijst. **Gebruik opnieuw geen comparator maar een lambda.**

En los ook deze extra vragen op, uiteraard enkel met streams en lambda's

4. Maak een lijst van alle namen van de Bonobo's geboren tussen 2000 en 2010. Sorteer deze lijst van namen alfabetisch en controleer het resultaat

5. Maak een lijst van alle geboortejaren van de Bonobo's. Zorg ervoor dat er geen duplicaten in de lijst zitten en sorteer de lijst aflopend. Controleer het resultaat.

6. Geef de alfabetisch eerste Bonobo terug waarvan de naam met een 'B' begint. Controleer het resultaat (dit zou Balina moeten zijn).

7. Is er een Bonobo geboren in 2000? Controleer dit via een stream.

8. Bereken de gemiddelde leeftijd van de Bonobo's (de getAge() methode krijg je al cadeau). Los dit op met een stream.

9. Tel hoeveel mannelijke en hoeveel vrouwelijke Bonobo's er zijn. Kan je dit met één stream oplossen? Of heb je er twee nodig? Bekijk hiervoor zelf de groupingBy collector.

Oefening 2 : Bonobo oppasser

Elk van onze Bonobo's krijgt een oppasser. De oppasser zijn Hans en Erik (fictieve namen 😊). De Bonobo's geboren voor 2000 krijgen Hans als oppasser, de andere Erik.

Maak een nieuwe class oppasser (CareTaker) met twee attributen naam (name) en bonobo (Bonobo).

Maak een nieuwe factory class om oppassers aan te maken (CareTakerFactory). De factory class heeft één method assignCareTaker . Deze method neemt als input een Bonobo en als output een CareTaker class en bevat de code om per Bonobo een oppasser aan te maken volgens de logica hierboven (Hans of Erik).

Schrijf een stream oplossing om van de lijst van Bonobo's een nieuwe lijst van CareTakers te maken.

En ga dan nu met deze nieuwe lijst van CareTakers aan de slag om via een stream de namen (enkel de namen) van alle Bonobo's van Erik op te zoeken en sorteer deze alfabetisch.

Oefening 3 : Dieren uit de kooi halen

Download van digitap de labo opgave code en plaats deze in een nieuwe 'animal' package. Deze code bevat de Cat en Dog en de Cage classes.

Als potentiële nieuwe baasjes een dier zoeken willen we dat onze kooi het correcte dier volgens de specificaties van het baasje aanbiedt. Bijvoorbeeld, het baasje wil het jongste, mannelijke dier uit de kooi. Of een vrouwelijk gesteriliseerd dier, leeftijd maakt niet uit.

Er zijn enorm veel mogelijkheden en we willen niet voor elke mogelijkheid een nieuwe method op de kooi zetten.

Maak één methode getAnimal() op de kooi aan en zorg ervoor dat je dynamisch kan meegeven wat de filtercriteria zijn. De methode geeft een dier van het type van de kooi terug, of null als er geen dier volgens die criteria is.

Voorzie minstens 2 verschillende zoekcriteria :

- Het jongste, mannelijke dier
- Een vrouwelijk, gesteriliseerd dier

Los dit op met lambda's en stream en test je code.

Denk aan de Predicate interface en Comparators om dit probleem op te lossen