Algemene oefening Java Advanced

Uitwerken van een reële case.
In groepjes van 2 a 3

Baseline: Klant met callcenter krijgt iedere dag call logs binnen en wil hier verschillende rapporten van trekken en lijsten opvraagbaar maken via verschillende methodes.

In fase 1 worden de call logs aangereikt via een static String object.

Voorzie een verzameling van call logs die automatisch gesorteerd wordt op prio en datum, geen duplicaten mag bevatten en geen null objecten
Voorzie een 2de verzameling van call logs die natuurlijk gesorteerd is op bedrijf.

Rapport 1: een print van call logs gesorteerd op prio
Rapport 2: een print van call logs gesorteerd op bedrijf
Rapport 3: een print van call logs die ouder zijn dan 2 jaar
Rapport 4: een print van call logs gesorteerd op datum
Rapport 3 en 4 mogen gecombineerd worden in een stream.

Voorzie een procedure die call logs per persoon kan ophalen.
Voorzie een andere procedure die call logs per status kan ophalen.

EXTRA
Maak een procedure die telkens 5 call logs verwerkt en de call logs met status CLOSED archiveert en de overige naar een aparte verzameling (ToProcess) kopieert tot er geen call logs meer zijn om te verwerken.

Fase 1
Requirements:

- Generics
- Collections
- Lamdas

Te volgen stappen:

1. Maak een object CallLog op basis van de data lijnen uit CallLogData.java
   (De eerste lijn is de header lijn)
2. Voorzie een enum voor de verschillende statussen
3. Parse alle data naar een Collection van CallLog objecten
   a. Automatisch gesorteerd op prio
   b. Natuurlijke sortering op bedrijf
4. Gebruik Lamda Expressions waar mogelijk

CallLogData.java

```
public class CallLogData {

	/*
	 * Example data for CallLog
	 */

	public static String getCallLogData() {
		StringBuffer buffer = new StringBuffer();

		buffer.append("id;naam;datum;tijd;bedrijf;omschrijving;prio;status\n");
		buffer.append("1;Jan Janssens;10/12/2015;10:00;PXL;random call;1;CLOSED\n");
		…

		return buffer.toString();
	}

}
```

Fase 2

Requirements:

- Streaming API
- File IO
- Exception Handling

Lees de testdata uit de verschillende csv bestanden

- testdata1.csv
- testdata2.csv
- testdata3.csv
- testdata4.csv
- testdata5.csv

Schrijf voor rapport 1 en 2 de resultaten naar een nieuwe csv file per rapport.
(vb calllog_rapport_1.csv)

Combineer rapport 3 en 4 en schrijf het resultaat in een archive folder voor calllogs die ouder zijn dan 2 jaar. De overige resultaten schijf je weg in csv files gegroepeerd op datum in jaar/maand/dag structuur. (vb: C:\data\results\2016\11\21\calllog_rapport_4.csv)

Gebruik een application.properties file om de locatie van de in te lezen bestanden te vinden en de weg te schrijven locaties op te vragen

- Input folder
- Output folder
    - Rapporten
    - Archive

Fase 3

Requirements:

- Multithreading
- Java command line

1. Voorzie een procedure die in een aparte thread een calllog bestand kan inlezen en een Collection met calllogs terug geeft.

2. Gebruik een aparte thread voor het creëren van de rapporten uit fase 2.

3. Package je applicatie in een runnable jar die je samen met een application.properties file eender waar kan runnen. (Indien de application.properties file niet kan gevonden worden moet degene die mee gepackaged is in jar file gebruikt worden).

EXTRA

Gebruik de CallLogLargeDataSet (=100 csv files met ieder 500 calllogs) en meet de performance met en zonder aparte threads uit punt 1 en 2.