# Tema 4

# Student:Oprean Andrei

# Grupa:30228

## 1. Obiectivul temei

Procesarea datelor transmise de senzorii unei ñsmart houseò. Aceste date reprezinta timpul de inceput si de sfarsit al unei activitati definite printr-o eticheta specifica(de exemplu ñSleepò). Aceasta procesare se va face cu ajutorul facilitatilor introduse de Java 8.

## 2. Analiza problemei, modelare, scenarii, cazuri de utilizare

### 2.1 Analiza problemei

Pentru rezolvarea acestei problem e trebuie sa folosim facilitatile introduse odata cu lansarea Java 8. Aceste facilitate, conform documentatiei Oracle sunt:

- **Lambda expression** ï Adds functional processing capability to Java.
- **Method references** ï Referencing functions by their names instead of invoking them directly. Using functions as parameter.
- **Default method** ï Interface to have default method implementation.
- **New tools** ï New compiler tools and utilities are added like ójdepsô to figure out dependencies.
- **Stream API** ï New stream API to facilitate pipeline processing.
- **Date Time API** ï Improved date time API.
- **Optional** ï Emphasis on best practices to handle null values properly.
- **Nashorn, JavaScript Engine** ï A Java-based engine to execute JavaScript code.

### 2.2 Modelare

Se poate observa citind rezumatul functionarii fiecarei facilitati, ca vom avea nevoie in principal de expresii Lambda pentru a procesa datele, de Stream API pentru a citi si a schimba diferite structuri de date si de Date Time API pentru a modela timpul de inceput si de sfarsit citit din fisier. Pentru asta vom avea nevoie de o descriere mai detaliata a acestor functionalitati.

Stream: a sequence of elements from a source that supports aggregate operations.

- **Sequence of elements:** A stream provides an interface to a sequenced set of values of a specific element type. However, streams don't actually store elements; they are computed on demand.
- **Source:** Streams consume from a data-providing source such as collections, arrays, or I/O resources.
- **Aggregate operations:** Streams support SQL-like operations and common operations from functional programing languages, such as filter, map, reduce, find, match, sorted, and so on.

Expresii Lambda:

> - Lambda expressions are a new and important feature included in Java SE 8. They provide a clear and concise way to represent one method interface using an expression. Lambda expressions also improve the Collection libraries making it easier to iterate through, filter, and extract data from a Collection. In addition, new concurrency features improve performance in multicore environments.

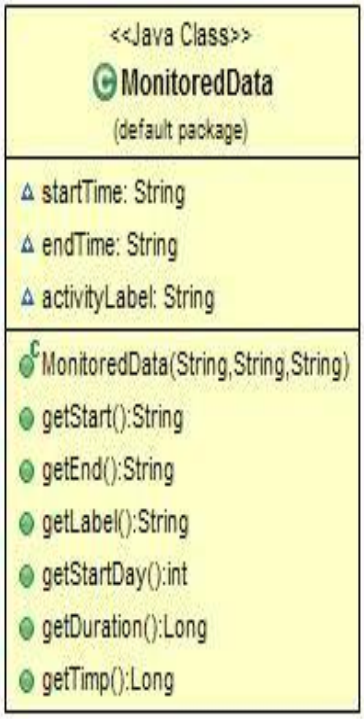
## 2.3 Scenarii

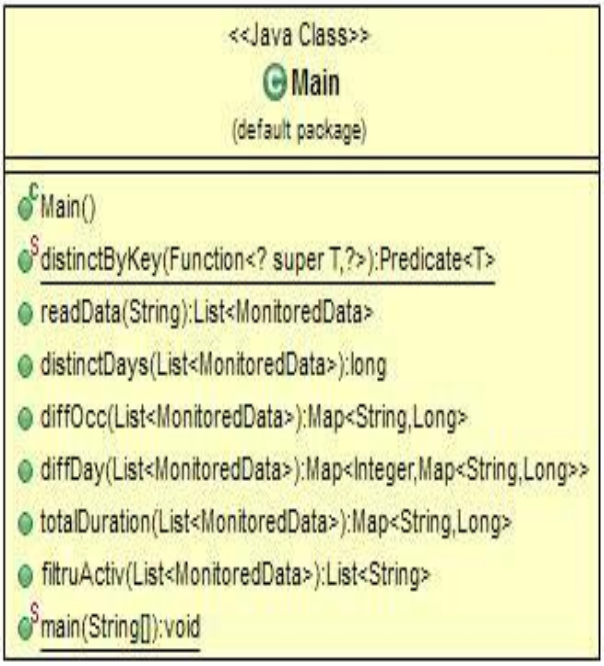
Diferite scenarii de utilizare pot aparea cu odata cu schimbarea fisierului din care sunt citite datele, adica diferitele valori returnate de senzori. Aplicatia eset construita sa functioneaza pe cazuri generale astfel ca oricare ar fi datele de intrare, datele de iesire sunt deterministe.

## 2.4 Cazuri de utilizare

Pentru acest proiect nu exista cazuri de utilizare, aplicatia procesand orice date de intrare intr-un mod specific. Utilizatorul nu poate influenta datele de intrare, care vin de la senzorii casei, deci nu poate infulenta din datele de iesire. Astfel ca daca datele transmise de senzori sunt valide atunci aplicatia le va filtra corespunzator cerintei.

# 3.Proiectare




Proiectarea este relativ simpla, cel putin din punct de vedere al cantitatii de cod scrise. Pentru modelarea datelor transmise am creat o clasa MonitoredData care declara 3 variabile de instanta : startTime, endTime, activityLabel in care voi salva datele de pe ficare linie din fisierul de intrare(o linie corespune cu o activitate). Pentru procesarea datelor am creat clasa Main care pe langa metodele de procesare contine si metoda main. Pentru procesarea datelor am folosit functionalitatile din Java 8 descrise mai sus.

Pentru metoda cu care citesc datele din fisier, readData(), folosesc urmatoarea secventa de cod :

```
try (Stream<String> stream = Files.lines(Paths.get(file))){
        buffer = stream.collect(Collectors.toList());
 }catch (IOException e){
        e.printStackTrace();
```

Ca si structuri de date folosesc o serie de List<> si Map<>:

-List<MonitoredData> in care stochez datele citite din fisier

-Map<String, Long> in care voi stoca, pentru fiecare actiune definita cu ajutorul unui label, numarul de aparitii in fisier

-Map<Integer, Map<String, Long>> in care voi stoca pentru fiecare zi, numarul de activitati pentrecute in acea zi

## 4.Implementare

Implementarea este simpla, avand in vedere ca acesta este si scopul Java 8. Pentru a rezolva problema 1, nu a fost nevoie decat de o simpla linie de cod:

```
long dist = d.stream()
        .filter(distinctByKey(p->p.getStart().substring(0,10))).count();
```

Din punctul acesta de vedere doar citirea a necesitat inca o structura repetitiva care pentru fiecare linie citita sa o imparta in datele specifice clasei MonitoredData.

```
for (String i : buffer){
            String[] parts = i.split(" ");
            String[] parts1 = parts[1].split("           ");
            String[] parts2 = parts[2].split("           ");
            date.add(new MonitoredData(parts[0]+" " + parts1[0],parts1[1]+ " " +
parts2[0],parts2[1]));
                        }
```

Ca si alta metoda specifica streamului am folosit .filter() , de exemplu, pentru a filtra activitatile care nu au durata macar 10 ore.

```
Files.write(Paths.get("Prob4.txt"),()->rez.entrySet()
            .stream()
            .filter(x -> x.getValue() > 600)
            .<CharSequence>map(g->g.getValue() + " "  +  "\n\n\n\n\n" +
g.getKey()).iterator());
```

## 5. Rezultate

Ca rezultat, am obtinut o aplicatie pentru procesarea datelor transmise de o serie de senzori speicifici unei ñsmart houseò. ñHowever, when we talk about home automation and smart homes, the capabilities go quite a bit further. And instead of individual devices working independently, a smart home integrates multiple sub-systems that are all controlled by a master home automation controller. This main automation controller is like the home automation system's quarterback, receiving input from all devices around the home, issuing commands and controlling everything.

These controllers generally run complex software, allowing them to execute single or multiple actions based on a variety of events. These events can come in many forms but can essentially be broken down to just two categories: timed and triggered.ò Avand aceasta interpretare a conceptului de òsmart house ò putem observa imediat aplicabilitatea unui astfel de program care poate oferi date exacte despre activitatile zilnice pe o perioada indelungata de timp. Asfel ca beneficile folosiri acestuia ar putea avea impact asupra sanatatii si pana la felul cum ne gestionam timpul in fiecare zi.

# 6. Concluzii, dezvoltari ulterioare

In concluzie, tema aceasta ma ajutat sa inteleg mai bine noile concepte introduse de Java 8 si in principal, crearea de cod compact atunci cand avem de a face cu un flux de date mare. Cu

ajutorul stream-urilor si a expresiilor Lambda putem procesa si mapa datele in aproape orice si mod si pe aproape orice structura de date.

Se pot remarca un lung sir de dezvoltari ulterioare, avand in vedere autmatizarea tot mai avansata a fiecarei parti din viata noastra.

# 7.Bibliografie


http://www.oracle.com/technetwork/articles/java/ma14-java-se-8-streams-2177646.html

http://www.mkyong.com/java8/java-8-streams-filter-examples/

http://www.mkyong.com/tutorials/java-8-tutorials/

http://www.javaworld.com/article/2092260/java-se/java-programming-with-lambda-expressions.html

https://stackoverflow.com/questions/32054180/java-8-stream-to-file