**DOCUMENTATIE TEMA 5**

**Processing sensor data of daily living activities**

**Csillag Szabolcs Andras**

**Grupa 30227**

**Profesor: Dan Mitrea**

Cuprins:

1. Cerintele temei

2. Analiza problemei

3.Proiectare

4.Implementare

5.Testare

6.Dezvoltari ulterioare si Concluzii

7.Bibliografie

# Cerintele Temei

( cerintele problemei )

Sa consideram un design pentru implementarea si testarea unei aplicatii , care analizeaza comportamentul unei persoane in urma unor senzori instalati in casa. Data este impartita in cateva zile , pe mai multe linii din fisierul Activities.txt ,care este preluat si descarcat . Istoricul activitatilor persoanei respective este stocat in tuple ( start\_time , end\_time , activity\_label) ,unde:

* Start\_time : reprezinta data exacta pentru inceperea unei activitati ;
* End\_time : reprezinta data exacta pentru terminarea unei activitati ;
* Activity\_label: reprezinta numele activitatii care are loc in intervalul acesta : Leaving , Toileting , Showering , Sleeping , Breakfast , Lunch , Dinner , Snack , Spare\_Time/TV , Grooming ;

**Obiective:**

(Obiectivul principal pe care il urmarim in acest proiect)

Scrieti un program care foloseste function programming in java cu lambda expressions si stream processing , pentru a performa task-urile listate in tabelul de mai jos . Rezultatul pentru fiecare task trebuie scris intr-un fisier .txt separat . ( fiecare .txt trebuie denumit in functie de taskul pe care il face task\_number.txt , un exemplu pentru taskul 1 ar fi : Task\_1.txt ) .

( Aici sunt prezentate cele 6 Tak-uri)

|  |  |
| --- | --- |
| **Task** | **Task Description** |
| Task\_1 | Definiti o clasa MonitoredData cu 3 componente: start\_time , end\_time si activit . Sa citim datele din fisierul Activity.txt folosind stream si sa facem split pe fiecare linie in cele 3 parti ,pe urma sa cream o lista de obiecte de tip MonitoredData . |
| Task\_2 | Sa numaram zilele distincte care apar in monitoring data . |
| Task\_3 | Sa numaram de cate ori a aparut fiecare activitate pe intreaga perioada monitorizata . Sa returnam o structura de tip Map<String,Integer> , care sa reprezinte maparea pentru fiecare activitate distincta si numarul aparitii acesteia . Keya sa fie un String ,reprezentata de numele activitatii. Valoarea sa fie un integer corespondent pentru care activitatea respectiva apare pe intreaga perioada monitorizata . |
| Task\_4 | Sa numaram de cate ori o activitate a aparut pentru fiecare zi din intreaga perioada . Trebuie returnat o structura de tip Map<Integer,Map<String,Integer>> , care contine activitatea numarata pentru fiecare zi . Keya pentru map-ul mare va fi un integer care va reprezenta o zi . Valoarea pentru map-ul mare va fi un alt map ,cu keya de tip String , insemnand numele activitatii si valoarea fiind un integer , reprezentand de cate ori aparea aceea activitate in ziua respectiva . |
| Task\_5 | Pentru fiecare activitate sa determinam duratia , mai bine zis intervalul total pe parcursul intregii periozi . Ar fi recomandat sa returnam un Map<String,LocalTime> , prin care keya sa fie un string ,reprezentand numele activitatii , iar valoarea un LocalTime prin care sa fie durata totala activitatii respective . |
| Task\_6 | Sa filtram activitatile care au mai mult de 90% din tot monitoring records , cu duratia mai putin de 5 minute . Sa colectam aceste rezultate intr-o lista de Stringuri cu numele activitatilor respective si sa returnam string-ul . |

# Analiza Problemei

( use case-uri / scenarii care au o probabilitate mare sa apara )

**Use case-uri / Scenarii**

Utilizarea acestei aplicatii se face cu rularea programului si verificarea rezultatelor dorite in fisierele .txt generate . Acest program nu necesita de intrari de la tastatura , se foloseste doar de fisierul Activities.txt ,pe care utilizatorul il poate modifica , poate pentru a selecta alte zile si date rezultate din senzorii aflati in casa persoanei respective . Cu ajutorul acestei aplicatii ne putem face singuri un feedback , de exemplu cat timp pierdem mancand ,dormind , stand in baie etc ..

# Proiectare

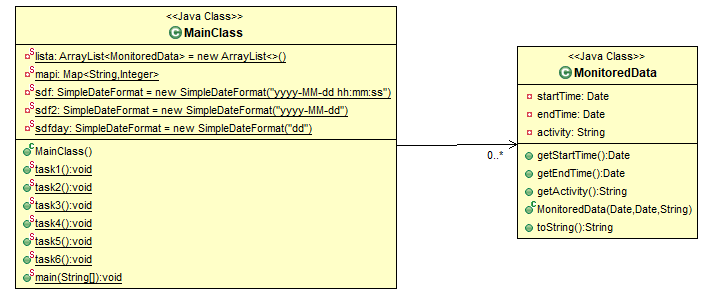
Aici voi descrie etapele de proiectare folosite pentru realizarea acestui proiect ,dar mai ales gandirea constructiei ,si metodele alese.

## a)Structuri de date

(Ce structuri de date am folosit ?) Pentru retinerea datelor din fisierul Activities.txt folosesc o lista de tipul MonitoredData . Am folosit stream-rui si lambda expressions. Voi descrie implementarea fiecarui task in parte si voi relua acolo discutia despre structurile de date folosite .

## b)Diagrama de clase

Pentru realizarea acestei teme am folosit doar 2 clase , voi discuta detaliile lor mai jos , esentialul este ca MainClass are o lista de MonitoredData .



# Implementare

Este implementat doar un pachet default ,pentru ca sunt doar 2 clase.

**1 Clasa MonitoredDAta :**

-contine 3 variabile instant : startTime si endTime ( de tip Date) , activity ( de tip String)

-sunt implementate gettere pentru cele 3 variabile instant: getStartTime() , getEndTime() , getActivity() ;

-un constructor simplu care primeste ca parametrii : 2 Date si 1 String si cu aceste valori primite ca parametru le atribuie valorile celor 3 variabile instante .

-toString este metoda prin care este returnat String-ul cum sa se afiseze un obiect MonitoredData .

**2 Clasa MainClass :**

**-**toate metodole arunca eceptia IOException .

**-**am facut cateva variabile instante , pentru a nu ma complica cu transmiterea prin parametru a acestor variabile : lista ( o lista de tip MonitoredData) ,mapi ( Map < String , Integer > ) si 3 SimpleDateFormat ,care ma va ajuta la lucrarea mai usoara cu variabilele de tip data .

**-**metoda main: este banal de simpla ,nu face decat sa apeleze cele 6 metode pentru fiecare task.

**a)metoda task1** : citesc in variabila rows ( lista de tip String) tot fisierul Activities.txt , pe urma folosind stream processing si lambda expression fac un map si un split (“\t\t”) , dupa care pentru fiecare linie din rows incerc sa adaug linia respective la lista ( variabila instant ) . Inchid rows . Pe urma fac un file task1 cu numele corespunzator : Task\_1.txt , daca acesta nu exista il cream. Cu acest nou File ,vom construi un nou PrintWriter , cu ajutorul careia vom afisa in fisierul Task\_1.txt toate elementele din lista ( var instant ) . Dupa care inchidem acest PrintWriter .

**b)metoda task2** : Oarecum intr-un rand se rezolva acest task2 , trebuie numerate zilele distinct :

Captură1.PNG Am facut aici cu ajutorul metodelor stream : folosim .stream pe lista noastra cu toate datele , mapez cu ajutorul expresiile lambda si formatul sdf2 : yyyy/MM/dd din StartTime , dupa care folosesc .distinct() pentru a le pastra doar pe cele unice , dupa care .count() va numara cate astfel de zile unice sunt .Avem cast la int si aceasta se salveaza in variabila dayCount .

Pe urma fac un file task2 cu numele corespunzator : Task\_2.txt , daca acesta nu exista il cream. Cu acest nou File ,vom construi un nou PrintWriter , cu ajutorul careia vom afisa in fisierul Task\_2.txt variabila locala dayCount . Dupa care inchidem acest PrintWriter .

**c)metoda task3** : Si acest task ,fiind unul simplu , se poate rezolva cu o singura linie de cod:

Captură2.PNG

Voi incerca sa explic aici ce se intampla , pentru ca poate fi un pic neclar : rezultatul va fi memorat in mapi ,care este o variabila instanta a clasei si ne reamintim ca este de tipul Map<String,Integer> . Vom folosi tot lista calculate in task-ul 1 , ii “dam” .stream() , folosim .collect pentru a putea salva sub forma unei colectii cum si zic toMap . Ce inseamna mai exact aceste expresii lambda , pai keya va lua x.getActivity() , adica numele unei activitati , valoarea map-ului va fi initial 1 (pentru ca pana cum avem pentru activitatea respectiva o aparitie) , urmatoarea expresie poate fi un pic unusual : val1 si val2 reprezinta valoarea veche pe care a avut-o keya aceasta ,respective cheia noua . Map-ul poate memora doar o valoare ,pentru o keie , atunci vom semnaliza ca vom postra suma lor . De exemplu la urmatoarea aparitie se va calcula 1+1 deci 2 aparitii ,pe urma 3 si tot asa .

Pe urma fac un file task3 cu numele corespunzator : Task\_3.txt , daca acesta nu exista il cream. Cu acest nou File ,vom construi un nou PrintWriter , cu ajutorul careia vom afisa in fisierul Task\_3.txt acest map creat, cu ajutorul unui foreach . Dupa care inchidem acest PrintWriter .

**d)metoda task4** : Pentru realizarea acestui task ,nu o sa imi fie de ajuns doar cate zile distinct sunt , o sa am nevoie de o lista de numere , care sa specific cate o zi unica din lista noastra .(days) Folosind un foreach prin care parcurg toata lista , si pun in lista days toate zilele unice . Am incercat initial sa folosesc getDay() ,dar acesta returna ziua dintr-o saptamana , mie imi trebuia ziua dintr-o luna , asa ca am formatat data “dd” pe urma i-am facut un parseInt ,pentru a o memora intr-un int (aux\_int) . Avand aceasta lista de zile ,vom face map2 de tipul Map<Integer,Map<String,Integer>> care va avea cheia una din lista de zile , iar valoarea va fi un alt map ,identic cu taskul3 doar ca are un filter ,prin care precizam ca startTime sau EndTime sa fie ziua din keie . Verific si startTime si endTime ,pentru ca exista posibilitatea ca o activitate sa inceapa intr-o zi sis a se termine in alta.

Pe urma fac un file task4 cu numele corespunzator : Task\_4.txt , daca acesta nu exista il cream. Cu acest nou File ,vom construi un nou PrintWriter , cu ajutorul careia vom afisa in fisierul Task\_4.txt map2 creat . Dupa care inchidem acest PrintWriter .

**e)metoda task5** : Implementarea acestui task este asemanatoare cu task3 ,doar ca aici nu vom pune acolo 1 ,ci durata pe care o are activitatea respectiva la acel moment ,adica luand pixul si foaia am facut o idee prin care daca scad secundele,minutele si orele din endTime cu cele din startTime ( minutele fiind inmultite cu 60 si orele cu 3600) .Exista o anumita posibilitate in care acest rezultat sa ofere un raspuns negative pentru care o activitate se termina ziua urmatoare . Asa ca verific ,daca orele din endTime<orele din startTime ,adun la rezultat 24 ore.

Pe urma fac un file task5 cu numele corespunzator : Task\_5.txt , daca acesta nu exista il cream. Cu acest nou File ,vom construi un nou PrintWriter , cu ajutorul careia vom afisa in fisierul Task\_5.txt map3. Dupa care inchidem acest PrintWriter .

**f)metoda task6** : O sa avem nevoie de un map4 auxiliar in care vom face ceva asemanator ca si in task-ul 3 ,doar ca acolo nu vom pune direct 1 . Daca durata acestei activitati e mai mica de 5 min ,adica 300 secunde ( 5 \* 60 ) ,atunci vom pune 1 pe valoarea acestui map , in rest vom pune 0.

Pe urma fac un file task6 cu numele corespunzator : Task\_6.txt , daca acesta nu exista il cream. Cu acest nou File ,vom construi un nou PrintWriter , cu ajutorul careia vom afisa in fisierul Task\_6.doar acele activitati care au cel putin 90% din mapi (task3) in map4 acum format . Dupa care inchidem acest PrintWriter .

# Testare

Testarea se face cu o simpla rulare a programului care teoretic nu afiseaza nimic , dar genereaza cele 6 fisiere .txt prin care ne putem verifica de corectitudinea codului . Fisierul Activities.txt este descarcat de pe site-ul cerintei .

# Concluzii si Dezvoltari Ulterioare

Acest proiect nu a necesitat de folosirea multor clase , dar aceste stream-uri si expresii lambda m-au salvat de screierea unui cod mai lung . La prima vedere sunt un pic greu de folosit , dar o data intelese pot devein eficiente in programarea java . As avea o mica observatie , in context am folosit “keya” pentru a semana cu “key” din engleza , sa fie usor de inteles legatura key -> value . Am explicat linie cu linie ce se intampla in acest program si cum functioneaza acesta , sper s ail inteleaga tot omul cand citeste aceasta documentatie . Nu am vorbit despre ce import-uri am folosit , ei bine toate cele necesare . Programul poate fi rulat de asemenea si cu un jar file .

# Bibliografie

Pentru intelegerea stream-urilor: <https://www.youtube.com/watch?v=t1-YZ6bF-g0>

Cerinta: <http://www.coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_5/Assignment_5.pdf>

Fisierul Activities.txt : <http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_5/>

<https://docs.oracle.com/javase/8/docs/api/java/util/Date.html>