Processing sensor data of daily living activities

Profesor: Student:

Ana-Maria Nanes Cristiana Tintesan

Grupa:30227

Disciplina: Tehici de Programare

2019-2020

# Cuprins

[Cuprins 2](#_Toc40902079)

[1.Analiza problemei 2](#_Toc40902080)

[2.Obiectivul lucrarii 3](#_Toc40902081)

[2.1 Obiectivul principal 3](#_Toc40902082)

[2.2 Obiectivul secundar 4](#_Toc40902083)

[3.Proiectare 4](#_Toc40902084)

[3.1 Decizii de proiectare 4](#_Toc40902085)

[3.4 Implementarea claselor 6](#_Toc40902086)

[3.5 Diagrama de clase 6](#_Toc40902087)

[4.Implementare 6](#_Toc40902088)

[4.1 Clasa MonitoredData 6](#_Toc40902089)

[4.2 Clasa MainClass 8](#_Toc40902090)

[5.Rezultate 9](#_Toc40902091)

[6.Concluzii 10](#_Toc40902092)

[7.Bibliografie 10](#_Toc40902093)

# 1.Analiza problemei

Cerinta acestui assignment: Implementati o aplicatie pentru a analiza comportamentul unei persoane. Datele despre activitatiile precticate de o persoana sunt citite dintr-un fisier text de intrare, in formatul urmator: start time, end time, activity. Start time reprezinta data si ora la care a inceput activitatea, end time indica data si ora la care s-a sfarsit activitatea, iar campul activity sugereaza denumirea activitatii. Activitatiile care pot fi practicate sunt urmatoarele: Leaving, Toileting, Showering, Sleeping, Breackfast, Lunch, Dinner, Snack, Spare\_Time/TV, Grooming. Datele sunt generate pentru o perioada de mai multe zile si sunt citite din fisierul Activities.txt in formatul prezentat anterior. Task-urile assignment-ului sunt urmatoare:

1. Definiti clasa Monitoreddata, care contine trei camputi: start time, end time si activity. Toate campurile mentionate sunt de tipul String. Cititi datele din fisierul Activities.txt folosind streams si despartiti fiecare linie citita din fisier in trei parti: start time, end time si activity si creati o lista de obiecte de tipul MonitoredData;
2. Numarati zilele in care a fost monitorizata persoana;
3. Numarati de cate ori apare fiecare activitate in decursul perioadei de monitorizare. Returnati o colectie de tipul map care reprezinta maparea dintre activitati si numarul de repetari ale acestora;
4. Numarati de cate ori aparea fiecare activitate in fiecare zi a perioadei de monitorizare;
5. Pentru fiecare activitate, calculati durata aceteia pe parcursul perioadei de monitorizare
6. Filtrati activitatiile care au mai mult de 90 la suta din monitorizarile inregistrate cu durata mai mica de 5 minute.

# 2.Obiectivul lucrarii

## 2.1 Obiectivul principal

Obiectivul principal al lucrarii este de a implementa o aplicatie pentru a analiza comportamentul unui persoane in functie de activitatile practicate de persoana respectiva. Pentru a putea indeplini task-urile cerute trebuie, in primul rand, sa intelegem cum functioneza expresiile lambda din java si cum functioneaza strem-urile.

Expresiile lambda din Java sunt introduse in Java 8. O expresie lambda este o fucntie care poate fi creata fara a fi implementa o clasa. Practic, expresiile lambda exprima instante ale unei interfete functionale( o interfata cu o singura metoda abstracta; un exemplu de interfata functionala este interfata runnable).

Conceptul de strem este introdus in Java 8 si este utilizat pentru a procesa colectii de obiecte. Un stream este o coloectie de obiecte care suporta diferite metode care pot fi aplicate pentru a obtine rezultatele dorite. Urmatoarele operatii sunt esentiale pentru lucrul cu stream-uri. Aceste operatii au fost folosite pentru a realiza cerintele din acest assignment:

* Map: Metoda map este utilizata pentru a mapa unele obiecte cu cele din interiorul colectiei, conform unui predicat specificat ca si argument;
* Filter: Metoda filter este utilizata pentru a filta elementele conform unui predicat dat ca si argument;
* Sorted: Metoda sorted este folosita pentru a sorta stream-ul;
* Collect: Metoda collect este folosita pentru a returna rezultatele operatiilor intermediare realizate in stream.
* Distint: Metoda distinct este folosita pentru a filtra sau colecta elementele distincte dintr-o coloectie.

## 2.2 Obiectivul secundar

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Alegerea structurilor de date | Structurile de date folosite pentru a duce la capat obiectivul principal | 3 |
| Impartirea pe clase | Sunt folosite doua clase pentru realizarea obiectivului principal:Clasa MainClass – este clasa care ruleza aplicatia si clasa MonitoredData in care sunt implementate primele trei task-uri din acest assignment | 3 |
| Implementarea solutiei | Vor fi descriese, pentru fiecare clasa in parte, campurile si metodele importante | 4 |
| Testare | Vor fi descries cateva scenarii de testare.In functie de un anumit set de date de intrare, se vor prezenta rezultatele simularii. | 5 |
| Dezvoltarea de use case-uri si scenarii care pot sa apara | Intr-un sistem de software un “use-case” este o lista de actiuni sau pasi de evenimente care definesc in mod obisnuit interactiunile dintre un rol si un sistem in atingerea unui obiectiv. | 3 |

# 3.Proiectare

# 3.1 Decizii de proiectare

Pentru realizarea proiectului, am luat urmatoarele decizii de implementare:

* Am impartit proiectul in doua clase, care vor fi detaliate ulterior: MainClass si MonitoredData;
* Am incercat sa folosesc cat mai multe metode simple, care sa nu contina multe linii de cod cu scopul de a oferi o intelegere rapida si usoara a codului;
* Folosirea unor nume sugestive pentru variabile;
* Pastrarea codului cat mai lizibil si usor de urmarit.

3.2 Structuri de date folosite

* In clasa MonitoredData este declarant un List de tipul MonitoredData pentru a putea pastra datele citite din fisier.

List<MonitoredData> splitData = new ArrayList<MonitoredData>();

* In clasa MainClass este declarant o colectie de tipul List pentru a putea apela metoda readData din clasa MonitoredData

List<MonitoredData> lst = m.readData();

3.3 Use case-uri/Scenarii

Utilizarea programului presupune citirea dintr-un fisier de text a datelor de intrare. Activities.txt este fisierul din care se vor citi datele despre activitatiile practicate pe parcursul perioadei de monitorizare. Acest fisier are urmatorul formar:

2011-11-28 02:27:59 2011-11-28 10:18:11 Sleeping

2011-11-28 10:21:24 2011-11-28 10:23:36 Toileting

2011-11-28 10:25:44 2011-11-28 10:33:00 Showering

Datele din fiser trebuie introduse conform formatului de mai sus pentru o buna functionarea a aplicatiei. Numarul de spatii dintre coloane trebuie respectat deoarece pentru indeplinirea primului task este citit continutul fisierului, iar apoi este folosita functia split pentru indentificarea celor trei coloane. Prima coloanal indica data, apoi ora la care incepe activitatea; a doua coloanal indica data, apoi ora la care se sfarseste activitatea, iar ultima coloanal prezinta denumirea activitatii desfasurate.

Pentru fiecare task in parte este creat un fisier txt de iesire, iar rezultatele generate in urma rularii programului vor fi trecute in fisierul de iesire pentru cerinta corespunzatoare.

## 3.4 Implementarea claselor

Proiectul este format din doua clase esentiale:

* Clasa MonitoredData – Cuprinde trei campuri de tipul String: startTime, endTime si activity. Pentru aceste trei campuri sunt generate metodele getStartTime, endStartTime, getActivity, set StartTime, setEndTime si setActivity. Task-urile 1, 2, 3 sunt indeplinite prin descrierea metodelor: readData(), countDays(), respective countActivities.
* Clasa MainClass – este clasa principala care ruleaza aplicatia. Aici sunt apelate toate metodele din clasa MonitoredData.

## 3.5 Diagrama de clase

A screenshot of a cell phone

Description automatically generated

# 4.Implementare

## 

## 4.1 Clasa MonitoredData

Clasa MonitoredData contine trei campuri esentiale pentru aplicatia noastra: startTime, endTime si activity. Toate aceste campuri sunt de tipul String. Constructorul clasei primeste ca si argumenente trei String-uri care indica timpul de inceput, timpul de sfarsit si denumirea activitatii practicate. Primele metode declarate sunt metodele de get si set pentru obtinerea informatiilor despre startTime, endTime si activity sau pentru a actualiza aceste informatii.

Metodele principale din aceasta clasa sunt:

* readData care are rolul de a deschide fisierul de intrare Activities si de a citi continutul acestuia folosint Strem. Continutul fisierului este memorat in colectia de tipul List numita data. Este parcurs fiecare rand din fisierul citit. Continutul fiecarui rand este impartit in trei parti: timpul de inceput, timpul de sfarsit si numele activitatii, folosind functia split. Tot in metoda readData este creat si fisierul de iesire Task\_1.txt in care sunt afisate activitatiile citite si datele despre aceste, conform cerintei;

public List<MonitoredData> readData() {

List<MonitoredData> splitData = new ArrayList<MonitoredData>();

try (Stream<String> stream = Files.lines(Paths.get("Activities.txt"))) {

List<String> data = stream.collect(Collectors.toList());

try {

FileWriter myWriter = new FileWriter("Task\_1.txt");

for (int i = 0; i < data.size(); i++) {

MonitoredData m = new MonitoredData();

String a = data.get(i);

String str[] = a.split(" ");

m.startTime += str[0];

m.endTime += str[1];

m.activity += str[2];

myWriter.write(m.startTime + " " + m.endTime + " " + m.activity);

myWriter.write("\n");

splitData.add(m);

}

myWriter.close();

} catch (IOException e1) {

System.out.println("Eroare la crearea primului fisier de iesire");

}

} catch (IOException e) {

e.printStackTrace();

}return splitData;

* Pentru cel de al-doilea task este descrisa metoda countDays. Zilele sunt numerate folosind stream-ul si apeland functia distinct pentru a ne asigura ca nu se numara o zi de mai multe ori. Metoda primeste ca si parametru o colectie de tipul MonitoredData pentru a putea parcurgele datele de intrare procesate la punctul anterior.
* Pentru task-ul cu numarul trei este descrisa matoda countActivities. De asemenea, argumentul trasmis acestei metoda este tot o colectie de tipul MonitoredData pentru a putea parcurge datele de intrare citite din fisier si procesate la primul punct din assignment. Este folosita functia map pentru a grupa fiecare activitate cu numarul ei de aparitii pe perioada de monitorizare. In interiorul metodei countActivities este creat si deschis fisierul de iesire cu numele Task\_3.txt in care sunt afisate rezultatele procesate de aceasta metoda. In cazul in care crearea fisierului esueaza, programul va arunca o exceptie de tipul IOException.

public void countActivities(List<MonitoredData> lst) {

Map<String, Long> rez = lst.stream().map(x -> x.activity)

.collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));

try {

FileWriter myWriter3 = new FileWriter("Task\_3.txt");

myWriter3.write(rez.values().toString());

String s = Arrays.asList(rez).toString();

myWriter3.write("\n");

myWriter3.write(s);

myWriter3.close();

} catch (IOException e3) {

System.out.println("Eroare la crearea fisierului pentru task3!");

}

}

## 4.2 Clasa MainClass

Clasa MainClass este clasa principala a programului deoarece contine o singura metoda esentiala: matoda statica main care are rolul de a rula aplicatia. In metoda main este creat fisierul de iesire pentru task-ul cu numarul 2 si sunt apelate cele trei metode importante din clasa MonitoredData descrise mai sus. Pentru apelul metodei readData este declaranta o colectie List cu elemente de tipul MonitoredData. In aceasta colectie numita in program lista sunt stocate informatiile citite din fiser, doar ca acum sunt grupate corespunzator dupa timpul de inceput, de sfarsit si denumirea activitatii. Pentru a doua cerinte este folosita colectia lista creata la punctul anterior, aceasta fiind trasmisa ca parametru pentru metoda countDays. Rezultatul returnat de countDays este convertit la tipul int, iar apoi este afisat in fisierul task\_2 sub forma unui String.Pentru task-ul trei este apelata metoda countActivities care primeste ca si parametru aceiasi colectie lst. La final este afisat pe ecran un mesaj corespunzator pentru a stii ca operatiile au fost efectuate cu success si afisate in fisierele de iesire corespunzatoare.

# 5.Rezultate

In urma testarii programumlui, s-a constatat ca operatiile de citire din fisierul text sunt corecte.Datele citite sunt prelucrate si afisate apoi in fisierul txt numit Task\_1. Operatia de numarare a zilelor pe care se desfasoara programul de monitorizare este corecta, iar rezultatul numararii este trecut in fisierul txt numit Task\_2. Metoda care numara de cate ori se repeat fiecare activitate functioneaza corect. Rezultatele numararii activitatiilor sunt trecute in fisierul text de iesire numit Task\_3.

Exemplu de date de intrare(vom prezenta primele linii din fisierul Activities.txt):

2011-11-28 02:27:59 2011-11-28 10:18:11 Sleeping

2011-11-28 10:21:24 2011-11-28 10:23:36 Toileting

2011-11-28 10:25:44 2011-11-28 10:33:00 Showering

2011-11-28 10:34:23 2011-11-28 10:43:00 Breakfast

2011-11-28 10:49:48 2011-11-28 10:51:13 Grooming

2011-11-28 10:51:41 2011-11-28 13:05:07 Spare\_Time/TV

2011-11-28 13:06:04 2011-11-28 13:06:31 Toileting

2011-11-28 13:09:31 2011-11-28 13:29:09 Leaving

2011-11-28 13:38:40 2011-11-28 14:21:40 Spare\_Time/TV

2011-11-28 14:22:38 2011-11-28 14:27:07 Toileting

2011-11-28 14:27:11 2011-11-28 15:04:00 Lunch

Rezultatele simularii:

* In fisierul Task\_1 vom avea urmatoarele date ( vom prezenta doar primele linii):

2011-11-28 02:27:59 2011-11-28 10:18:11 Sleeping

2011-11-28 10:21:24 2011-11-28 10:23:36 Toileting

2011-11-28 10:25:44 2011-11-28 10:33:00 Showering

2011-11-28 10:34:23 2011-11-28 10:43:00 Breakfast

2011-11-28 10:49:48 2011-11-28 10:51:13 Grooming

2011-11-28 10:51:41 2011-11-28 13:05:07 Spare\_Time/TV

2011-11-28 13:06:04 2011-11-28 13:06:31 Toileting

2011-11-28 13:09:31 2011-11-28 13:29:09 Leaving

2011-11-28 13:38:40 2011-11-28 14:21:40 Spare\_Time/TV

2011-11-28 14:22:38 2011-11-28 14:27:07 Toileting

2011-11-28 14:27:11 2011-11-28 15:04:00 Lunch

* In fisierul Task\_2 vom avea urmatoarele date:

14 – reprezinta numarul total de zile in care se desfasoara perioada de monitorizare

* In fisierul Task\_3 vom avea urmatoarele date:

[14, 14, 14, 11, 51, 14, 77, 44, 9]

[{Leaving =14, Breakfast =14, Sleeping=14, Snack =11, Grooming =51, Showering =14, Spare\_Time/TV=77, Toileting =44, Lunch=9}]

Pe prima linie din fisierul Task\_3 sunt trecute doar numerele care indica de cate ori s-a desfasurat fiecare activivate pe parcursul perioadei de monitorizare, iar a doua linie indica activitatea si numarul de repetari.

# 6.Concluzii

In urma realizarii acestui assignment, am reusit sa ma familiarizez cu expresiile lambda din Java si cu conceptul de strem. Aceasta tema nu a fost una foarte dificila deoarece s-a putut rezolva rapid, in putine linii de cod.

Posibilitati ulterioare de dezvoltare: indeplinirea task-ului 4, task-ului 5 si a task-ului 6 din cerintele proiectului.Dezvoltarea unei interfete grafice pentru introducerea si afisarea datelor doarece rezultale pot fi mai usor vizibile decat in fisierele text de iesire.

# 7.Bibliografie

<https://archive.ics.uci.edu/ml/datasets/Activities+of+Daily+Living+(ADLs)+Recognition+Using+Binary+Sensors>

<https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html>