

MINISTERUL EDUCAȚIEI NAȚIONALE



UNIVERSITATEA TEHNICĂ
DIN CLUJ-NAPOCA

PROCESSING SENSOR DATA OF DAILY LIVING ACTIVITIES

Documentatie

Facultatea: Automatica si Calculatoare

Roman Alexandru | Grupa 30228

CUPRINS

1. Obiectivul si descrierea proiectului
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii

1. Obiectivul si descrierea proiectului

Acest proiect vizeza filtrarea activitatilor personale monitorizate de catre senzori si selectarea informatiilor necesare despre acestea in conformitate cu taskurile proiectului.

Cerinta proiectului este de a implementa si testa o aplicatie pentru analiza comportamentului unei persoane, comportament analizat de senzori instalati in propria lui casa. Activitatile facute de persoana respectiva sunt salvate ca si tuple(start_time, end_time, activity_label), unde start_time si end_time reprezinta data si ora la care a inceput, respectiv la care s-a sfarsit activitatea. Activitatile detectate de catre senzori sunt : Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner, Snack, Spare_Time/TV, Grooming. In total 10 activitati. Data acumulata este impartita in zile

Datale sunt salvate intr-un fisier numit Activities.txt care poate fii descarcat la adresa: http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_5/.

Programul trebuie implementat utilizand functional programming in limbajul Java folosind expresii lambda si procesare pipeline a stremului pentru a executa taskurile necesare.

Rezultatele sunt salvate 6 fisiere .txt diferite(fiecare fisier este numit Task_1.txt de exemplu pentru taskul 1, si as mai departe pana la Task_6.txt pentru taskul 6).

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

Dupa cum am mentionat si mai sus, acest proiect are 6 taskuri de rezolvat. Hai sa vorbim putin despre cerintele acestora. Desigur unele dintre aceste taskuri ofera detalii despre implementarea lor, precum tipul structurii de date pe care scesta taskuri il vor returna dar despre aceste aspecte vom vorbi la capitolul implementare.

Task_1: cere definirea unei clase cu numele “MonitoredData”, care vom memora 3 stringuri ai anume timpul de incepere a activitatii, timpul de inceput al activitatii, respectiv numele acestei activitati.

Task_2: cere numararea zilelor distincte care apar in fisier.

Task_3: cere numararea a cate dati fiecare activitate a aparut dealungul intregii perioade de monitorizare.

Task_4: cere numararea a cate dati fiecare activitate a aparut pentru fiecare zii distincta din perioada de monitorizare

Task_5: cere calcularea intregii durate pe care s-a desfasurat fiecare activitate in parte dealungul perioadei de monitorizare.

Task_6: cere filtrarea activitatilor care au mai mult de 90% din timp o durata de mai putin de 5 secunde.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

In continuare vom vorbi putin despre proiectarea proiectului. In primul rand, proiectul contine doar 3 clase si o interfata, toate fiind puse In default package. Clasele proiectului se numesc:

-MainClass

-TaskManager

-MonitoredData

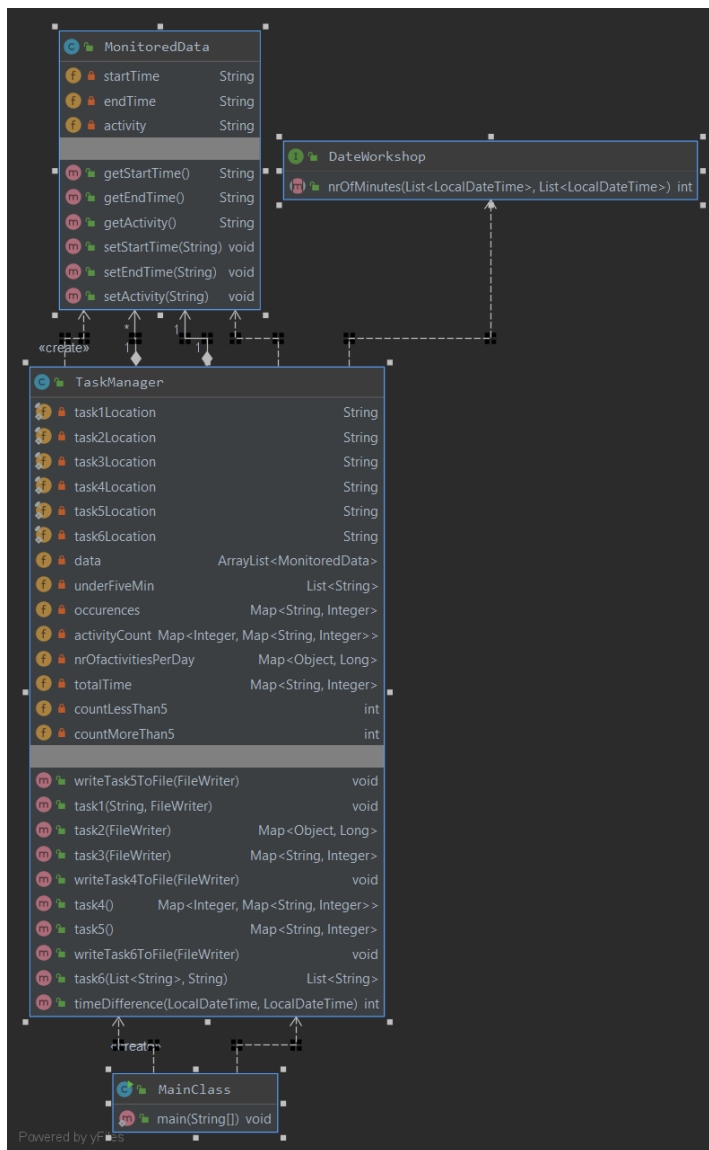
Iar numele interfetei este DateWorkshop.

Va voi spune putin despre functionalitatile fiecarei clase iar in sectiunea de mai jos vom vorbi despre implementarea acestora impreuna cu implementarea metodelor sale.

Clasa care va contine metoda main va fii desigur MainClass. Clasa TaskManager este ptractic clasa principala a proiectului. Ea va contine constantele de tip String cu pathurile spre fisiere si va contine structurile de date impreuna cu metodele care vor rezolva taskurile. o instanta a clasei MonitoredData va contine blueprintul unei linii din fisier.

Din asta deducem ca vom avea o structura de date de tip List de tipul clase MonitoredData pentru a salva datele intregului fisier Activities.txt. Interfata DateWorkshop o vom folosii pentru a declara o metoda ce va fii implementata in TaskManager folosind lambda.

Mai jos vom putea observa diagrama UML a proiectului. Putem observa 2 tipuri de relatii dintre clase. Prima este relatia de agregare dintre TaskManager si MonitoredData deoarece MonitoredData se afla in compozitia clasei TaskManager. A doua este relatia de dependenta si o putem observa atat intre MainClass si TaskManager(MainClass nu poate exista fara TaskManager) cat si intre TaskManager si MonitoredData deoarece MonitoredData este o variabila instanta a clasei TaskManager, variabila fara de care clasa nu poate exista. De asemenea o putem observa intre TaskManager si DateWorkshop deoarece in constructorul din TaskManager se declara folosind conceptul de lambda unica metoda a interfetei si anume nrOfMinutes, iar apoi se executa.



Structurile de date folosite in proiect sunt cele de tip List si Map. List am folosit pentru a stoca structuri cu elemente de acelasi tip iar Map acolo unde elementele aveau nevoie de o cheie pentru a le accesa. De mentionat este ca taskurile au fost realizate folosind prelucrarea folosind streamuri precum descrie cerinta asa ca am avut nevoie de colectii de date.

```

private ArrayList<MonitoredData> data = new ArrayList<>();
private List<String> underFiveMin;
private Map<String, Integer> occurrences;
private Map<Integer, Map<String, Integer>> activityCount;
private Map<Object, Long> nrOfactivitiesPerDay;
private Map<String, Integer> totalTime;
  
```

O interfata grafica nu a fost necesara pentru acest proiect deoarece citirea datelor se va face din fisierul Activities.txt iar iesirea rezultatelor prelucrate se va face in fisierele Task_1, 2..etc mentionate mai sus.

4. Implementare

Acum ca v-am pus la curent cu cerintele proiectului si cu cateva detalii de proiectare vom putea vorbi despre implementare unde voi explica functionalitatea fiecarei clase si metode si modul cum au fost folosite.

Clasa **MainClass** : este clasa care contine metoda main.

```
public static void main(String[] args){
    try {
        TaskManager manager = new TaskManager();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

In aceasta clasa se instantiaza Clasa TaskManager si se ocupa de tratarea exceptiilor de tip IOException din constructorul clasei TaskManager

Clasa **TaskManager** : este clasa care se ocupa de rezolvarea taskurilor.

```
private static final String task1Location = "Task_1.txt";
private static final String task2Location = "Task_2.txt";
private static final String task3Location = "Task_3.txt";
private static final String task4Location = "Task_4.txt";
private static final String task5Location = "Task_5.txt";
private static final String task6Location = "Task_6.txt";
```

In aceasta clasa sunt declarate constantele de tip string care reprezinta fisierele ce vor fii deschise de FileWriter pentru a scrie.

Constructorul clasei arunca exceptia in antetul functiei, motiv pentru care exceptiile vor fii tratate in clasa MainClass. In acest constructor, pentru fiecare task, se va instantia clasa FileWriter cu argumentul cu numele fisierului de output a taskului respectiv, apoi se va apela functia sau functiile care executa taskul si se ocupa de scrierea in fisier, iar apoi se va inchide FileWriterul.

Pentru rezolvarea primului task am creat functia task1 care se ocupa atat de rezolvarea lui cat si de scrierea rezultatelor in fisierul destinatie, Task1.txt. Aceasta este functia care va citii de asemenea din fisier. A fost creata structura data de tip List<MonitoredData>. Dupa cum putem vedea si mai jos, folosind functia lines(Paths.get("filename")) am luat linie cu linie din fisier, am impartit linia dupa spatiile din ea deoarece snt 2 taburi intre fiecare sectiune, si am adaugat acele stringuri intr-o instanta noua a clasei MonitoredData. Cu ajutorul functiei collect am transformat toate aceste instante a clasei MonitoredData intr-un list< MonitoredData >.

Cu ajutorul functiei write din clasa FileWriter am scris rezultatele in fisier.

```
data = (ArrayList<MonitoredData>) Files.lines(Paths.get(fileName))
    .map(s -> s.split( regex: "\\s{2}"))//each line split
    .map(d -> new MonitoredData(d[0], d[1], d[2]))
    .collect(Collectors.toList());
```

Pentru rezolvarea celui de-al doilea task care cere numararea zilelor distincte care apar in fisier am creat functia task2. Pentru rezolvarea acestui task m-am folosit de functia stream() pentru prelucrarea colectiilor de date, am impartit timpul de start in bucati (an luna, zi, ora, ect..) si am numarat cu ajutorul functiei Collectors.groupBy de cate ori a aparut acea zii in structura data. Apoi desigur numam cate zile am parcurs.

Pentru rezolvarea celui de-al treilea task care cere numararea a cate dati fiecare activitate a aparut dealungul intregii perioade de monitorizare am creat functia task3 lucru realizat de functia Collectors.counting() adaugata ca al doilea parametru a functiei Collectors.groupingBy. Ceea ce vom numara este numarul de cate ori apare fiecare activitate in structura data. functia Collectors.groupingBy va returna o structura de tipul Map<String, Long> asa ca vom transfera datele intr-o structura de tipul Map<String, Integer> pe care o vom si returna, conform cerintei. Cheia va fii un string cu numele activitatii iar valoarea va fii un Integer care va reprezenta numarul de aparitii ale acestei activitati. Scrierea in fisier a fost de asemenea facuta in aceasta functie deoarece deja parcurgeam structura de date asa ca am evitat sa mai creez o functie care o parcurge din nou doar pentru a scrie in fisier.

Pentru rezolvarea celui de-al patrulea task care cere numararea a cate dati fiecare activitate a aparut pentru fiecare zii distincta din perioada de monitorizare. Pentru a rezolva acest task m-am folosit deasemenea de structura de date rezultata in functia task2. In cerinta taskului al doilea nu cerea returnarea a unei structuri de date dar eu am returnat o structura de tip Map<Object, Long> care stocheaza numarul zilei impreuna cu numarul de activitati in acea zii. Desigur pe noi ne intereseaza doar cheia acelei structuri si anume numarul zilei. Folosind functia filter() am parcurs structura de date returnata la task2 si am filtrat activitatile care se intamplau

intr-o zii anume. Apoi in alta structura am numarat de cate ori se intampla fiecare activitate din ziua respectiva. Rezultatul l-am pus intr-un map de tipul Map<Integer, Map<String, Integer>>.

Integerul este ziua respectiva, iar valoarea este alt Map care contine numele activitatii impreuna cu frecventai ei din acea zii. Am folosit functia writeTask4ToFile pentru a afisa datele

Pentru rezolvarea celui de-al cincelea si al saselea task care cere calcularea intregii durate pe care s-a desfasurat fiecare activitate in parte dealungul perioadei de monitorizare am creat task5. Pentru rezolvarea taskului 5 am fost nevoit sa folosesc structura de date returnata la task3adica occurrences. Deci am pus intr-o lista de stringuri fiecare activitate(doar numele) si l-am parcurs. Apoi am parcurs lista si am filtrat date activitatilor pe rand, pentru fiecare activitate in parte. Folosind functia split am separat fiecare linie in an, luna, zii, ora, etc. Am stocat rezultatele in 2 List<LocalDateTime> unul timeStart si timeEnd. Am creat interfata **TimeWorkShop** care declara o functie nummita nrOfMinutes. Folosind sintaxa lambda am declarat acea functie si am executat-o. Funtia parcurge listele timeStart si timeEnd si se foloseste de rezultatul functiei timeDifference pentru a rezulva de asemenea si exerciciul 6. Va voi explica in ce consta asta. Funtia timeDifference returneaza rezultatul in minute(int) dintre 2 instante a clasei LocalDateTime.

Taskul 6 cere filtrarea activitatilor care au mai mult de 90% din timp o durata de mai putin de 5 secunde. Asa ca vom verifica fiecare durata a fiecarei activitati in parte si daca 90%+ sunt mai mici de 5 minute vom adauga numele activitatii intr-o lista folosind functia task6.

Pentru sccrierea rezultatelor in fisierele respctive vom avea functiile writeTask5ToFile si writeTask6ToFile.

5. Rezultate

Rezultatele programului pot fii vizualizate in fisierele .txt de output a fiecarui task.

La Taskul 1 se vor scrie start_time, end_time, activity_label pentru fiecare actiitate in parte. Practic se va afisa continutul structurii de date de tip List<MonitoredData>.

Task_1 - Notepad
File Edit Format View Help

MonitoredData structure contains:

| | | |
|---------------------------------|-------------------------------|-------------------------------|
| start_time: 2011-11-28 02:27:59 | end_time: 2011-11-28 10:18:11 | activity_label: Sleeping |
| start_time: 2011-11-28 10:21:24 | end_time: 2011-11-28 10:23:36 | activity_label: Toileting |
| start_time: 2011-11-28 10:25:44 | end_time: 2011-11-28 10:33:00 | activity_label: Showering |
| start_time: 2011-11-28 10:34:23 | end_time: 2011-11-28 10:43:00 | activity_label: Breakfast |
| start_time: 2011-11-28 10:49:48 | end_time: 2011-11-28 10:51:13 | activity_label: Grooming |
| start_time: 2011-11-28 10:51:41 | end_time: 2011-11-28 13:05:07 | activity_label: Spare_Time/TV |
| start_time: 2011-11-28 13:06:04 | end_time: 2011-11-28 13:06:31 | activity_label: Toileting |
| start_time: 2011-11-28 13:09:31 | end_time: 2011-11-28 13:29:09 | activity_label: Leaving |
| start_time: 2011-11-28 13:38:40 | end_time: 2011-11-28 14:21:40 | activity_label: Spare_Time/TV |
| start_time: 2011-11-28 14:22:38 | end_time: 2011-11-28 14:27:07 | activity_label: Toileting |
| start_time: 2011-11-28 14:27:11 | end_time: 2011-11-28 15:04:00 | activity_label: Lunch |
| start_time: 2011-11-28 15:04:59 | end_time: 2011-11-28 15:06:29 | activity_label: Grooming |
| start_time: 2011-11-28 15:07:01 | end_time: 2011-11-28 20:20:00 | activity_label: Spare_Time/TV |
| start_time: 2011-11-28 20:20:55 | end_time: 2011-11-28 20:20:59 | activity_label: Snack |
| start_time: 2011-11-28 20:21:15 | end_time: 2011-11-29 02:06:00 | activity_label: Spare_Time/TV |
| start_time: 2011-11-29 02:16:00 | end_time: 2011-11-29 11:31:00 | activity_label: Sleeping |
| start_time: 2011-11-29 11:31:55 | end_time: 2011-11-29 11:36:55 | activity_label: Toileting |
| start_time: 2011-11-29 11:37:38 | end_time: 2011-11-29 11:48:54 | activity_label: Grooming |

In al diulea fisier vom gasii rezultatele taskului 2 si anume numarul de zile distincte care apar in perioada de monitorizare, dupa cum puteam vedea, numarul aceste este etal cu 14.

Task_2 - Notepad
File Edit Format View Help

numarul de zile distincte care apar in monitoring data este : 14

In fisierul Task_3.txt com gasii toate activitatile insiruite impreuna cu frecventa acesteia in perioada de monitorizare

Task_3 - Notepad

File Edit Format View Help

```
Activity: Leaving appeared 14 times
Activity: Breakfast appeared 14 times
Activity: Sleeping appeared 14 times
Activity: Snack appeared 11 times
Activity: Grooming appeared 51 times
Activity: Showering appeared 14 times
Activity: Spare_Time/TV appeared 77 times
Activity: Toileting appeared 44 times
Activity: Lunch appeared 9 times
```

In fisierul Task_4.txt vom putea gasii sortate pe zile, toate activitatile care s-au intamplat in acea zii impreuna cu frecventa cu care s-au intamplat in acea zii. Zielele sunt sortate crescator si nu a fost nevoie de agaugarea datei intregi deoarece sunt doar 14 zile iar structura din cerinta cerea doar un Integer si nu un LocalDateTime.

Task_4 - Notepad

File Edit Format View Help


```
day 1
    activity Breakfast happened 1 times
    activity Toileting happened 2 times
    activity Grooming happened 3 times
    activity Sleeping happened 1 times
    activity Leaving happened 1 times
    activity Spare_Time/TV happened 6 times
    activity Showering happened 1 times
    activity Lunch happened 1 times

day 2
    activity Breakfast happened 1 times
    activity Toileting happened 3 times
    activity Grooming happened 4 times
    activity Sleeping happened 1 times
    activity Spare_Time/TV happened 7 times
    activity Snack happened 1 times
    activity Showering happened 1 times
    activity Lunch happened 1 times

day 3
    activity Breakfast happened 1 times
    activity Grooming happened 3 times
    activity Toileting happened 2 times
    activity Sleeping happened 1 times
    activity Leaving happened 1 times
    activity Spare_Time/TV happened 4 times
    activity Showering happened 1 times

day 4
    activity Breakfast happened 1 times
    activity Grooming happened 2 times
    activity Toileting happened 4 times
    activity Sleeping happened 1 times
    activity Leaving happened 1 times
    activity Spare_Time/TV happened 6 times
```


In fisierul Task_5 putem vizualiza durata intreaga pe toata perioada de monitorizare a fiecarei activitati. Putem observa ca cea mai scurta durata a fost Snack iar perioada care in total a durat cel mai mult a fost Spare_Time/TV.

 Task_5 - Notepad

File Edit Format View Help

```
Activity Leaving had a total duration of 1658 minutes
Activity Breakfast had a total duration of 171 minutes
Activity Sleeping had a total duration of 7856 minutes
Activity Snack had a total duration of 4 minutes
Activity Grooming had a total duration of 139 minutes
Activity Showering had a total duration of 85 minutes
Activity Spare_Time/TV had a total duration of 51710 minutes
Activity Toileting had a total duration of 124 minutes
Activity Lunch had a total duration of 310 minutes
```

Vizualizand fisierul Task_6 putem observa activitatea care a avut peste 90% din durate mai putin de 5 minute si anume Snack, care este deasemenea si cea mai scurta durata.

 Task_6 - Notepad

File Edit Format View Help

```
Activity Snack - has more than 90% of the monitoring records with duration less than 5 minutes
```

6. Concluzii si posibilitati de dezvoltare ulterioara

In concluzie, fiecare proiect ma ajuta sa-mi aprofundez cunostintele in java, si in programare, in general si desigur, imi deschide mintea catre noi posibilitati de rezolvare a problemelor legate de programare si implementare a solutiilor alese. Acest proiect m-a determinat sa invat sa folosesc noi concepte de programare, precum prelucrarea colectiilor cu ajutorul streamurilor sau precum implementarea functiilor anonime cu ajutorul expresiilor lambda.

In Implementare acestui proiect nu m-am lovit de buguri mari sau de probleme mari de implementare dar lucrand la acest proiect am reusit sa inteleg conceptele de programare mentionate mai sus, si anume lambda expressions si stream processing.

Acest proiect, ca si oricare altul, are posibilitati de dezvoltare in anumite domenii. Desigur programul implementat de mine este unul micut dar ar putea fii dezvoltat sa proceseze cantitati mai mari de date. Un exemplu ar putea fii procesarea datelor vizitatorilor unui site. Fiecare persoana care acceseaza un site intra pe anumite sectiuni unde petrece mai mult sau

mai puțin timp, sau caută lucruri într-un search box, etc. Toate acțiunile asemănătoare care pot fi descrise prin date pot fi prelucrate de un algoritm asemănător.