

**TEHNICI DE PROGRAMARE FUNDAMENTALE
ASSIGNMENT 5**

**PROCESSING SENSOR DATA OF
DAILY LIVING ACTIVITIES**

DOCUMENTATIE

Bakk Cosmin-Robert
Grupa 30227

Cuprins

1. Obiectivul temei
2. Analiza problemei, modelare, scenarii, cazuri de utilizare
3. Proiectare
 - 3.1. Diagrama UML
 - 3.2. Proiectare clase
4. Implementare si testare
 - 4.1. Metode
 - 4.2. Testare
5. Concluzii
6. Bibliografie

1. Obiectivul temei

Obiectivul acestei teme de laborator este de a proiecta, implementa și testa o aplicație pentru a analiza comportamentul unei persoane înregistrate de un set de senzori instalați în casa acesteia. Datele despre activitatea persoanei pe un interval de timp vor fi salvate într-un fișier .txt, Activities.txt. Pe fiecare rând din fișierul de intrare vor fi scrise numele unei activități făcute de persoana, împreună cu timpii de început și de sfârșit a activității, în formatul “yyyy-MM-dd HH:mm:ss yyyy-MM-dd HH:mm:ss Activity name ”. Numele activităților este unul dintre următoarele: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Snack, Spare_Time/TV, Grooming. Pe baza datelor extrase din fișierul de intrare, se vor rezolva următoarele task-uri, iar rezultatele vor fi puse în câte un fișier .txt, specific fiecărui task:

Task 1: Scrierea tuturor datelor citite din fișier.

Task 2: Numărul de zile distincte în care s-a realizat înregistrarea activității.

Task 3: Numărul de apariții a fiecărei activități pe întreaga perioadă a înregistrării activității.

Task 4: Numărul de apariții a fiecărei activități pentru fiecare zi din perioada înregistrării activității.

Task 5: Durata totală a fiecărei activități.

Task 6: Filtrarea activităților pentru care mai mult de 90% din apariții au durată mai mică de 5 minute.

Obiectivele secundare sunt: respectarea paradigmei programării orientate pe obiect, folosirea funcțiilor pe stream-uri, folosirea expresiilor lambda și crearea unei arhive de tipul .jar pentru portabilitate și testarea mai ușoară a programului.

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

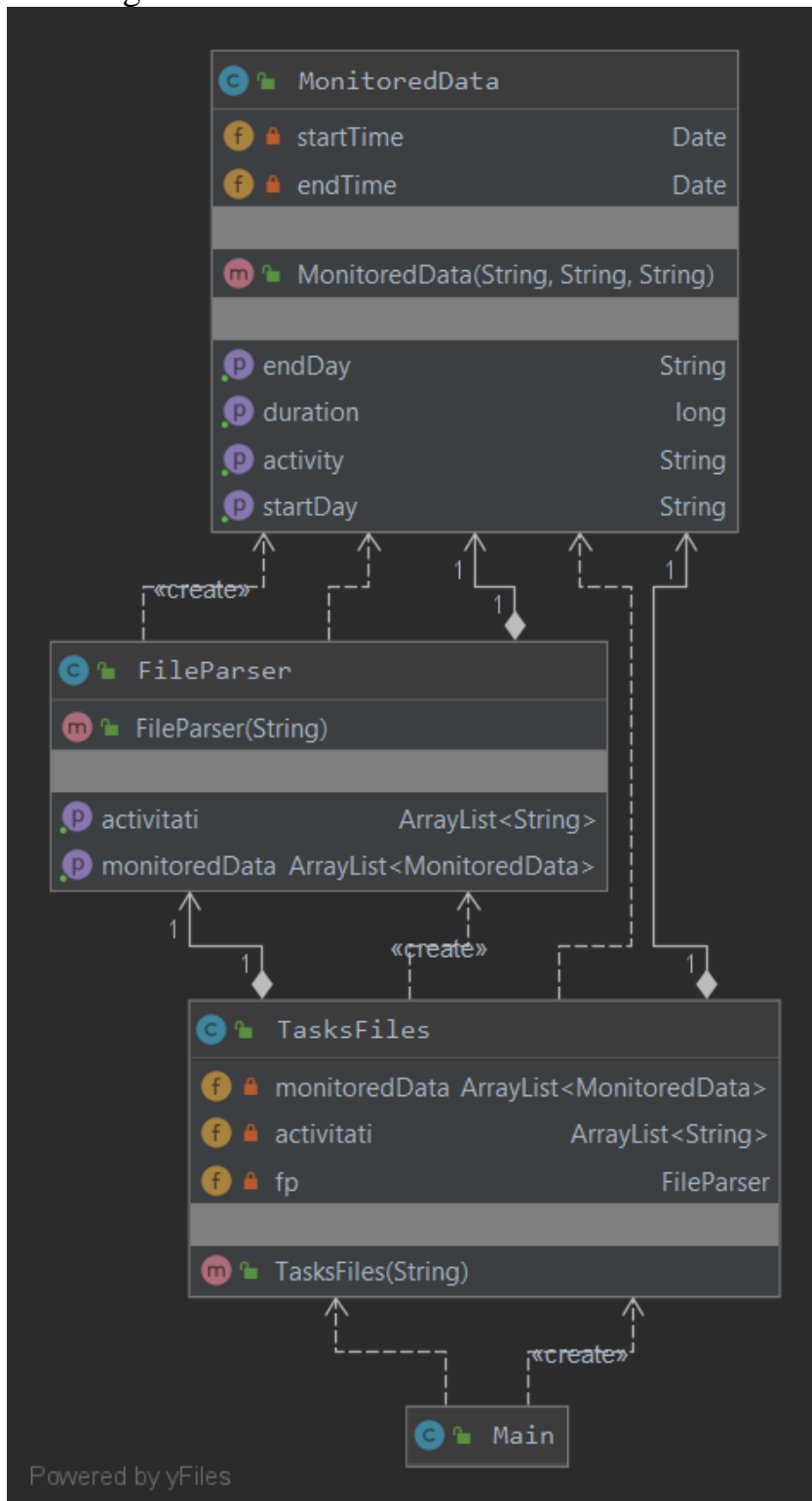
La executarea programului, datele despre activitatea persoanei sunt citite din fișierul de intrare și informațiile despre fiecare activitate vor fi salvate într-o listă de obiecte de tipul MonitoredData, care are un String activity și două variabile de tipul Date, startTime și endTime. Cu ajutorul acestei liste, vor fi implementate cele 6 task-uri. Pentru implementarea celor 6 task-uri, colecțiile au fost convertite în stream-uri, un stream fiind o secvență de elemente pe care se pot aplica operații agregate secvențiale și paralele. O operație aplicată unui stream poate fi văzută ca un query pe acel stream. Majoritatea operațiilor pe stream-uri accepta parametri, în care se pot folosi expresii lambda. O expresie lambda apelează o metodă care are un nume sau creează o metodă anonimă care oferă un comportament parametrizat. Elementul din stanga săgeții (->) este parametrul expresiei lambda, iar elementul din dreapta săgeții este corpul expresiei lambda, locul unde este implementată logica expresiei.

Pentru a împărtăși metodele proiectului, am creat cele 4 clase: clasa MonitoredData, care definește informațiile despre fiecare activitate care apare în fișierul de intrare, clasa FileParser, care se ocupă cu citirea din fișierul de intrare și cu scrierea în fișierele de ieșire, pentru fiecare task, clasa TasksFiles, unde sunt implementate metodele pentru rezolvarea task-urilor și clasa Main, care conține metoda main a proiectului.

Rezultatele obținute în urma executării celor 6 task-uri vor fi salvate în task_1.txt, task_2.txt, task_3.txt, task_4.txt, task_5.txt și task_6.txt.

3. Proiectare

3.1. Diagrama UML



3.2. Proiectare clase

Clasa MonitoredData: aceasta clasa defineste conceptul de activitate care a fost inregistrata pe intreaga perioada. Campurile clasei sunt: startTime, de tipul Date, reprezentand timpul de start al activitatii, endTime, de tipul Date, reprezentand timpul de final al activitatii si activity de tipul String, reprezentand numele activitatii.

Clasa FileParser: aceasta clasa se ocupa cu citirea din fisierul de intrare si cu scrierea in fisierele de iesire. Campurile clasei sunt: monitoredData, de tipul ArrayList <MonitoredData>, reprezentand lista completa de activitati care au fost realizate pe intreaga perioada si activitati, de tipul ArrayList <String>, reprezentand lista de nume unice ale activitatilor care apar in lista monitoredData.

Clasa TasksFiles: aceasta clasa se ocupa cu implementarea metodelor prin care se rezolva cerintele celor 6 task-uri. Campurile clasei sunt: monitoredData, de tipul ArrayList <MonitoredData>, reprezentand lista completa de activitati care au fost realizate pe intreaga perioada, activitati, de tipul ArrayList <String>, reprezentand lista de nume unice ale activitatilor care apar in lista monitoredData si fp, de tipul FileParser. reprezentand obiectul cu ajutorul caruia se apeleaza metodele clasei FileParser.

Clasa Main: aceasta clasa este clasa main a proiectului. In aceasta clasa se creeaza un obiect nou de tipul TasksFiles, in parametrul constructorului se scrie numele fisierului de input: "Activities.txt", iar apoi prin obiectul creat se apeleaza metodele prin care se rezolva cerintele celor 6 task-uri.

4. Implementare si testare

4.1. Metode

Clasa MonitoredData

```
public MonitoredData (String startTime, String endTime, String activity)
    - constructorul clasei, campul activity al clasei primeste String-ul activity primit ca parametru, iar pentru
      startTime si endTime, conversia din tipul String in tipul Date se realizeaza cu ajutorul metodei parse aplicata
      pentru un nou obiect de tipul SimpleDateFormat, cu pattern-ul "yyyy-MM-dd HH:mm:ss"
public String toString ()
    - metoda suprascrisa toString, pentru a afisa un obiect de tipul MonitoredData
public String getStartDay ()
    - metoda care returneaza doar datele despre zi a campului startTime (formatul "yyyy-MM-dd") sub forma de
      string
public String getEndDay ()
    - metoda care returneaza doar datele despre zi a campului endTime (formatul "yyyy-MM-dd") sub forma de
      string
public long getDuration ()
    - metoda care returneaza durata unei activitati in milisecunde, aceasta fiind egala cu diferenta dintre cele doua
      campuri de tipul Date, endTime si startTime, pe care a fost aplicata metoda getTime ()
public String getActivity ()
    - metoda care returneaza campul activity
```

Clasa FileParser

```
public FileParser (String path)
    - constructorul clasei, in acesta se apeleaza metoda read, avand ca parametru String-ul path
public void write (String path, String s)
    - metoda care primeste 2 String-uri ca parametrii, primul fiind path-ul, numele fisierului .txt in care se va
      scrie, iar al doilea fiind string-ul care se va scrie in fisier
    - scrierea in fisier se realizeaza cu ajutorul metodei write apelata printr-un obiect de tipul PrintWriter
public void read (String path)
    - metoda care primeste un String ca parametru, acesta reprezentand path-ul, numele fisierului .txt din care se
      vor citi datele
```

- liniile fisierului de intrare sunt salvate intr-un Stream de obiecte de tip String, acest stream fiind apoi colectat intr-o lista de string-uri, cu ajutorul metodei collect
- se parcurge lista de string-uri, fiecare element al listei fiind impartit prin regex-ul “\t+”, iar string-urile rezultate (vor fi in numar de 3), sunt folosite pentru constructorul unui obiect nou de tipul MonitoredData, care este apoi adaugat in lista monitoredData; de asemenea, daca al 3-lea string care rezulta din impartirea elementului iterat (numele activitatii) nu se gaseste in lista activitati, atunci va fi adaugat listei – astfel, in lista respectiva vor fi salvate toate numele activitatilor care apar in fisierul de intrare

```
public ArrayList <MonitoredData> getMonitoredData ( )
    - metoda care returneaza lista monitoredData a clasei
public ArrayList <String> getActivitati ( )
    - metoda care returneaza lista activitati a clasei
```

Clasa TasksFiles

```
public TasksFiles (String path)
    - constructorul clasei, primeste ca parametru String-ul path, numele fisierului, care este folosit pentru a
    initializa campul fp al clasei, iar apoi campurile monitoredData si activitati primesc valorile returnate de cele
    doua metode ale obiectului fp, getMonitoredData ( ), respectiv getActivitati ( )
public void createTask1 ( )
    - pentru rezolvarea primului task, se parcurge lista monitoredData, iar fiecare obiect iterat este concatenat la
    un string
    - la final, string-ul este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind
    string-ul “task_1.txt”, numele fisierului
public void createTask2 ( )
    - pentru rezolvarea celui de-al doilea task, intr-o variabila de tipul long se salveaza numarul elementelor
    distincte din stream-ul care are ca sursa lista monitoredData, mapat dupa String-urile returnate de metoda
    getStartDay ( ) aplicata fiecarui element din expresia lambda
    - la final, rezultatul este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind
    string-ul “task_2.txt”, numele fisierului
public HashMap <String, Integer> createTask3 ( )
    - pentru rezolvarea celui de-al treilea task, se creeaza un HashMap, activities, avand cheia un String,
    reprezentand numele activitatii, iar valoarea un Integer, reprezentand numarul de aparitii al activitatii pe
    intreaga perioada
    - intr-o lista de String-uri, allActivities, se vor colecta toate String-urile returnate de metoda getActivity ( ),
    dupa care a fost mapat stream-ul care are ca sursa lista monitoredData
    - pentru fiecare element al listei de String-uri activitati a clasei, care contine numele activitatilor distincte, se
    va pune in HashMap-ul activities o noua intrare: cheia acestei intrari va fi String-ul iterat din lista activitati,
    iar valoarea intrarii va fi numarul de elemente al stream-ului care are ca sursa lista allActivities, filtrat astfel
    incat sa contina doar String-urile egale cu String-ul iterat din lista activitati
    - se parcurge HashMap-ul activities si datele se salveaza intr-un String, care este trimis ca al doilea parametru
    al metodei write a obiectului fp, primul parametru fiind string-ul “task_3.txt”, numele fisierului
    - la final, se returneaza HashMap-ul activities obtinut ca rezultat
public HashMap <Integer, HashMap<String, Integer>> createTask4 ( )
    - pentru rezolvarea celui de-al patrulea task, se creeaza un HashMap, activityCountDays, avand cheia un
    Integer, reprezentand numarul zilei, iar valoarea un HashMap care are cheia un String, reprezentand numele
    activitatii, iar valoarea un Integer, reprezentand numarul de aparitii al activitatii pe durata zilei
    - se creeaza o lista de String-uri, allStartDays, in care se adauga toate valorile returnate de metoda
    getStartDay ( ) a elementelor din lista monitoredData
    - se creeaza o lista de String-uri, startDays, in care se adauga elementele unice ale listei allStartDays
    - se itereaza lista startDays, pentru fiecare iteratie se creeaza un HashMap nou, activityCount, avand cheia
    un String, reprezentand numele activitatii, iar valoarea un Integer, reprezentand numarul de aparitii al
    activitatii in ziua respectiva; se itereaza apoi lista de String-uri activitati a clasei, care contine numele
    activitatilor distincte, pentru fiecare element se pune o noua intrare in HashMap-ul activityCount: cheia
    acestei intrari reprezinta elementul iterat al listei activitati, iar valoarea intrarii reprezinta numarul de
    elemente al stream-ului care are ca sursa lista monitoredData, filtrat de urmatoarea conditie: String-ul care
    reprezinta ziua curenta sa fie egal fie cu cel returnat de metoda getStartDay ( ), fie de metoda getEndDay ( ),
    si String-ul care reprezinta numele activitatii sa fie egal cu cel returnat de metoda getActivity ( ); dupa ce se
```

itereaza toata lista activitati, se pune o noua intrare in HashMap-ul activityCountDays: cheia acestei intrari reprezinta numarul zilei curente (calculat ca numarul de elemente din acest HashMap + 1), iar valoarea intrarii reprezinta HashMap-ul activityCount

- se parcurge HashMap-ul activityCountDays si datele se salveaza intr-un String, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul "task_4.txt", numele fisierului

- la final, se returneaza HashMap-ul activityCountDays obtinut ca rezultat

```
public HashMap<String, Duration> createTask5 ()
```

- pentru rezolvarea celui de-al cincilea task, se creeaza un HashMap, durations, avand cheia un String, reprezentand numele activitatii, iar valoarea un obiect de tipul Duration, reprezentand durata totala a activitatii pe intreaga perioada

- se itereaza lista de String-uri activitati, la fiecare astfel de iteratie intr-o lista de elemente de tipul Long, duration, se colecteaza stream-ul care are ca sursa lista monitoredData, mapat dupa elementele de tipul Long returnate de metoda getDuration (), filtrat dupa urmatoarea conditie: String-ul returnat de metoda getActivity() trebuie sa fie egal cu String-ul iterat al listei activitati; este iterat apoi lista duration, fiecare element fiind adaugat la o variabila de tipul long, time; variabila time este parsata apoi catre un obiect de tipul Duration, dur, folosind metoda ofMillis; in HashMap-ul durations se pune o intrare noua: cheia acestei intrari reprezinta String-ul iterat al listei activitati, iar valoarea intrarii reprezinta obiectul de tipul Duration, dur

- se parcurge HashMap-ul durations si datele se salveaza intr-un String, fiecare durata fiind scrisa sub forma HH:mm:ss, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul "task_5.txt", numele fisierului

- la final, se returneaza HashMap-ul durations obtinut ca rezultat

```
public List<String> createTask6 ()
```

- pentru rezolvarea celui de-al saselea task, se creeaza o lista de String-uri, activities

- se itereaza lista de String-uri activitati, la fiecare astfel de iteratie intr-o lista de elemente de tipul Long, duration, se colecteaza stream-ul care are ca sursa lista monitoredData, mapat dupa elementele de tipul Long returnate de metoda getDuration (), filtrat dupa urmatoarea conditie: String-ul returnat de metoda getActivity() trebuie sa fie egal cu String-ul iterat al listei activitati; intr-o variabila de tipul long, nr, se salveaza numarul elementelor din stream-ul care are ca sursa lista duration, filtrat dupa urmatoarea conditie: valoarea elementelor impartita la 6000 sa fie mai mica decat 5; intr-un intreg, nrTotal, se pune dimensiunea listei duration, care reprezinta numarul total de aparitii a unei activitati pentru ziua respectiva; in cazul in care variabila nr reprezinta mai mult decat 90% din variabila nrTotal, se adauga la lista activities String-ul iterat al listei activitati

- se parcurge lista activities si datele se salveaza intr-un String, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul "task_6.txt", numele fisierului

- la final, se returneaza lista activities obtinuta ca rezultat

Clasa Main

```
public static void main (String[] args)
```

- este clasa main a proiectului

- se creeaza un obiect nou de tipul TasksFiles, tasks, in parametrul constructorului se scrie numele fisierului de input: "Activities.txt", iar apoi prin obiectul creat se apeleaza metodele prin care se rezolva cerintele celor 6 task-uri

4.2. Testare

In urma executiei comenzii:

```
java -jar PT2020_30227_Bakk_Cosmin-Robert_Assignment_5.jar
```

vor fi create urmatoarele fisiere:

 task_1

 task_2


 task_3

 task_4

 task_5


 task_6

avand continuturile:

 task_1 - Notepad


File Edit Format View Help

```
Start time: 2011-11-28 02:27:59, End time: 2011-11-28 10:18:11, Activity: Sleeping
Start time: 2011-11-28 10:21:24, End time: 2011-11-28 10:23:36, Activity: Toileting
Start time: 2011-11-28 10:25:44, End time: 2011-11-28 10:33:00, Activity: Showering
Start time: 2011-11-28 10:34:23, End time: 2011-11-28 10:43:00, Activity: Breakfast
etc
```

 task_2 - Notepad

File Edit Format View Help

Distinct days that appear in the monitoring data: 14

 task_3 - Notepad

File Edit Format View Help

Number of times each activity has appeared over the monitoring period:

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


Number of times each activity has appeared for each day over the monitoring period:

```
Ziua 1
Breakfast 1
Toileting 3
Grooming 2
Sleeping 1
Leaving 1
Spare_Time/TV 4
Showering 1
Snack 1
Lunch 1
```

```
Ziua 2
Breakfast 1
Toileting 4
Grooming 3
Sleeping 1
Leaving 1
Spare_Time/TV 7
Showering 1
Snack 1
Lunch 1
```

```
Ziua 3
Breakfast 1
```

etc

 task_5 - Notepad

File Edit Format View Help

Entire duration of each activity over the monitoring period:

```
Breakfast 2:58:08
Toileting 2:20:34
Grooming 2:40:42
Sleeping 131:03:31
Leaving 27:44:44
Spare_Time/TV 142:28:55
Showering 1:34:09
Snack 0:06:01
Lunch 5:13:31
```


Activities that have more than 90% of the monitoring records with duration less than 5 minutes:

Snack

5. Concluzii

Acest proiect functioneaza conform cerintelor task-urilor prezentate, insa poate fi imbunatatit prin adaugarea unor noi task-uri, sau chiar prin adaugarea mai multor fisiere de intrare pentru a se putea face o statistica pentru diferite persoane. Proiectul m-a ajutat sa invat cum se utilizeaza stream-urile si expresiile lambda, acestea fiind utile pentru a crea un cod mult mai compact.

6. Bibliografie

1. <https://www.geeksforgeeks.org/stream-in-java/>
2. <https://www.baeldung.com/java-8-streams-introduction>
3. http://coned.utcluj.ro/~salomie/PT_Lic/4_Lab/Assignment_5/Assignment_5.pdf
4. <http://tutorials.jenkov.com/java-date-time/parsing-formatting-dates.html>
5. <https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html>