**Lambda Expressions**

Balus Dan

Grupa 30227

# Obiectivul temei

Homework 5 Stream Processing using Lambda Expressions

Description :

A smart house features a set of sensors that may be used to record the behavior of a person living in the house. The historical log of the person’s activity is stored as tuples (startTime, endTime, activityLabel), where startTime and endTime represent the date and time when each activity has started and ended while the activity label represents the type of activity performed by the person: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner, Snack, Spare\_Time/TV, Grooming. The attached log file Activities.txt contains a set of activity records over a certain period of time.

Define a class MonitoredData having startTime, endTime and activityLabel as instance variables and read the input file data into the data structure monitoredData of type List. Using stream processing techniques and lambda expressions introduced by Java 8, write the following set of short programs for processing the monitoredData.

1. Count the distinct days that appear in the monitoring data

2. Determine a map of type that maps to each distinct action type the number of occurrences in the log. Write the resulting map into a text file.

3. Generates a data structure of type Map> that contains the activity count for each day of the log (task number 2 applied for each day of the log) and writes the result in a text file.

4. Determine a data structure of the form Map that maps for each activity the total duration computed over the monitoring period. Filter the activities with total duration larger than 10 hours. Write the result in a text file.

5. Filter the activities that have 90% of the monitoring samples with duration less than 5 minutes, collect the results in a List containing only the distinct activity names and write the result in a text file. Design by Contract Programming Techniques, Design Patterns

REFERENCES:

[1] Ordóñez, F.J.; de Toledo, P.; Sanchis, A. Activity Recognition Using Hybrid Generative/Discriminative Models on Home Environments Using Binary Sensors. Sensors 2013, 13, 5460-5477.

[2] Available online at https://archive.ics.uci.edu/ml/datasets/Activities+of+Daily+Living+(ADLs)+Recognition+Using+Binary+Se nsors

# Analiza problemei, modelare, scenarii, cazuri de utilizare

## 2.1 Analiza problemei

O casa inteligenta sau automatizata reprezinta un stil de viata imbunatatit, deservind nevoilor fundamentale ale omului de confort, functionalitate si siguranta.

Cand propunem si integram un sistem de casa inteligenta, urmarim in primul rand sa simplificam si sa imbunatatim confortul, astfel, o simpla apasare a unui buton sa fie de ajuns pentru ca sa poti cobori storurile, sa inchizi luminile din toata casa, sa reglezi temperatura pentru o noapte friguroasa, sa armezi sistemul antiefractie; lucruri care te vor scuti de cateva drumuri, iti vor economisi timpul si iti vor mari cu siguranta gradul de confort.

Unele dintre facilitate ar putea fi:

-  Controlul  temperaturii

- Controlul securitatii

- Controlul audio/video

- Sistemul de Automatizare al Portilor

- Sistemul de Irigare al gazonului

- Accesul vizitatorilor prin intermediul Sistemelor de [Interfonie / VideoInterfonie](http://www.aviseco.ro/securitate/interfonie-videointerfonie-81.html)

- Sistemul de Acoperire(Prelata) al piscinei

 - Alte sisteme sau echipamente de uz casnic

**2.2 Structuri de date**

In acest proiect am folosit mai multe structuri de date, in funcție de necesitățile dezvoltării operațiilor. Aceste structuri sunt: TreeMap, ArrayList, List. Aceste structuri de date ajuta la gestionarea datelor citite din fișier.

Activitățile preluate din fișierul log dat sunt inițial salvate intr-un ArrayList. Am ales salvarea acestora intr-o astfel de structura deoarece oferă o buna gestionare a datelor. Din aceasta structura datele sunt modelate si filtrate in funcție de cerințele metodelor.

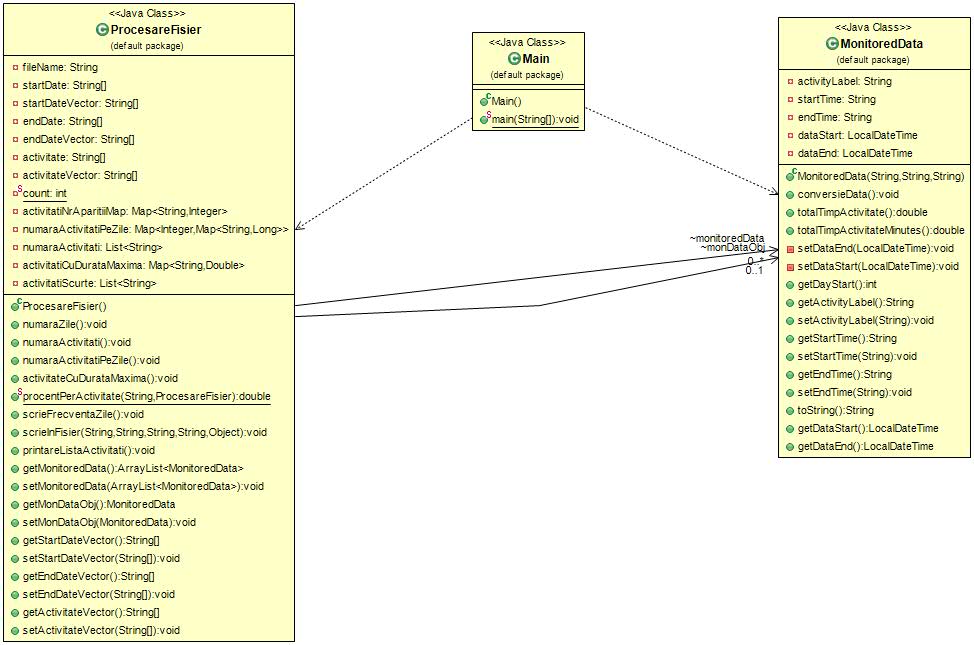
HashMap-ul este folosit pentru memorarea diferitelor date si a activităților aferente acestora. Programul conține mai multe tipuri de HashMap- uri cu elemente diferite. Elementele conținute de HashMap sunt rezultate din procesarea datelor conținute de ArrayList- ul menționat anterior. Am ales aceasta structura de date deoarece operațiile de inserare si ștergere se efectuează foarte eficient, crescând așadar eficienta întregului program.

**2.3 Scenarii, cazuri de utilizare**

Aplecatia se ruleaza, fara necesitatea unei interfete grafice ce interactioneaza cu operatorul, dar rezultatul progromului va fi scris in mai multe fisiere .txt, dupa rularea acestuia

**3.Proiectare**

**3.1 Diagrame UML**



**3.2 Designul Claselor si Algoritmi**

In proiecte este implementat un singur pachet si 3 clase

a)**MonitoredData**

- aici este clasa ce se ocupa de monitorizarea activitatilor

**Constructori**:

- exista un singur constructor ce primeste ca parametru timpul de inceput(startTime), timpul de sfarsit(endTime) si activitatea(activityLabel).

**Metode**:

-**public** **double** **totalTimpActivitate**() returneaza cu ajutorul ChronoUnit diferenta in ore de la data de start pana la data de sfasit

-**public** **double** **totalTimpActivitateMinutes**()returneaza cu ajutorul ChronoUnit diferenta in minute de la data de start pana la data de sfasit

- **public** **int** **getDayStart**() returneaza ziua din data primita, cu ajutorul metodei getDayOfMonth al LocalDataTime class

- **public** **String** **toString**() unde se suprascrie metoda toString pentru a afisa data de inceput, de final si activitate

- get-ere + set-ere ce vor lua/scrie atributele clasei

b)**ProcesareFisier**

**-** modeleaza o inregistrarea a unei actiuni in fisierul activities.txt ce contine toate activitatile ce s-au efectuat in casa inteligenta

**Constructori**:

- In constructorul acestei metode vom citi cu ajutorul metodeo forEach apelata de un obiect de tip stream din fisierul dat data de start, de end si activitatea, constuind tot aici si obiectul de tip monitored data. De asemenea, se pune obiectul intr un ArrayList.

**Metode:**

**- public** **void** **numaraZile()**

aceasta metoda rezolva primul task. Cu ajutorul stream-urilor lista monitoredData menționată anterior este parcursa, sunt salvate intr-o alta lista doar zilele distincte regăsite in monitoredData si sunt numărate prin metoda count specifica stream-urilor.

- **public** **void** **numaraActivitati()**() (

)aceasta metoda rezolva al doilea task . Cu ajutorul stream-urilor lista monitoredData menționată anterior este parcursa si modelata in așa fel încât sa rezulte un Map ce conține fiecare activitate cu numărul de apariții in fișierul dat. Pentru realizarea operației s-au folosit metodele următoare specifice stream-urilor : filter, map, distinct, collect si count.

-- **public** **void** **numaraActivitatiPeZile()** ()

aceasta metoda rezolva al treilea task. Vom stoca intr-un Map cu ajutorul motodelor count, stream si collect, vom colecta datele si le vom grupa dupa activitate si vom numara de cate ori apar.

- **public** **void** **activitateCuDurataMaxima()** ()

aceasta metoda rezolva al patrulea task. Se vor colecta datele grupate dupa activitate, iar cu ajutorul metodei reducing vom face suma duratei activitatilor cu ajutorul mtodei totalTimpActivitate si le vom filtra daca au suma mai mare decat 10 ore, colectandu-le intr-un Map.

-  **public** **static** **double** **procentPerActivitate(**(**String** activity, **ProcesareFisier** procFisier))

aceasta metoda ajuta la rezolvarea ultimului task. Cu ajutorul acestei metode ce primeste ca parametrii activitatea si un obiect de tipul clasei, vom afla procentajul activitatii respective,care va fi rezultatul dintre numarul activitatilor care dureaza mai putin de 5 minute \* 100 / toate activitatile.

-**public** **void** **scrieFrecventaZile**()

aceasta metoda rezolva task-ul 5. Metoda foloseste si ea filtrarea, maparea, metoda distinct si collect. Daca procentajul este mai amre de 90 % atunci se pune in lista activitatea

- **public** **void** **printareListaActivitati**(aceasta metoda scrie la consola toate obiectele de tip monitoredData

- **public** **void** **scrieInFisier** se va scrie in fisier

- get-ere + set-ere ce vor lua/scrie atributele clasei

**c)Main**

Metoda main se ocupa de generalizarea tuturor metodelor, aici fiind apelate si testate.

**4. Implementare si Testare**

Aplicatia a fost testata doar pe fisierul activities.txt pus la dispozitie odata cu cerinta problemei.

**5. Rezultate, concluze, dezvoltari ulterioare**

Dupa scrierea acestei aplicatii, dar si in timp ce scriam mi-am dat seama ca oricat de simpla poate parea o aplicatie(din punct de vedere al implementarii) atunci cand o implementezi vei intampina mai multe dificultati, care consta in acele cazuri de exceptie care la prima vedere nu sunt chiar atat de vizibile. In cazul acestei aplicatii dificultatea a constat in studierea si intelegerea unor concepte specifice Java8 si anume stream processing tehniques si lambda expression.

Lambda expressions sunt introduse in Java 8 si sunt considerate a fi cel mai important feature adus in plus.Acestea faciliteaza programarea functionala si simplivica dezvoltarea aplicatiiolor foarte mult.

**5.1 Informatii adunate**

Realizand aceasta aplicatie am invatat mai multe lucruri noi:

-am invatat sa folosesc stream processing tehniques:

• Scrierea respectiv citirea dintr-un fisier este mult simplificata

• Filtrarea elementelor dintr-o colectie de orice tip(List,Map,Set etc) este cu mult usurata si mult mai rapida atat din punct de vedere a vitezei de executie cat si din punct de vedere a scrierii codului(mult mai putin cod scris decat o rezolvare clasica a problemelor)

-am invatat sa folosesc lambda expression

**5.2 Dezvoltari ulterioare**

O posibila dezvoltare ulterioara ar putea fi creearea unei interfete cu care utilizatorul ar puta comunica mia usor cu programul.

Adaugarea unpr elemente de securitate a datelor.

Operatii de manioulare a datelor, de exemplu editarea lor

**6. Bibliografie**

http: // stackoverflow.com/

https: // docs.oracle.com

<http://winterbe.com/posts/2014/03/16/java-8-tutorial/>

<http://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>

www.google.com