

**FACULTATEA DE AUTOMATICA SI CALCULATOARE**

**DEPARTAMENTUL CALCULATOARE**

**DISCIPLINA TEHNICI DE PROGRAMARE**

**Documentatie Tema 5**

**Aplicatie pentru procesarea**

**stream-urilor utilizand expresii lambda**

Pavel Madalina Adriana

Grupa 30224

**CUPRINS**

**1. OBIECTIVUL TEMEI ……………………………………………………………………..3**

**2. ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE …...4**

**3. PROIECTARE (DECIZII DE PROIECTARE, DIAGRAME UML, STRUCTURI DE DATE, PROIECTARE CLASE, INTERFETE, RELATII, PACKAGES, ALGORITMI, INTERFATA UTILIZATOR) …………………………..……………………………………5**

**4. IMPLEMENTARE …………………………………………………………………………6**

**5. REZULTATE …………………………………….…………………………………………7**

**6. CONCLUZII .……………………………………………………………………………....10**

**7. BIBLIOGRAFIE …………………………………………………………………………..10**

**1.OBIECTIVUL TEMEI**

Enuntul temei este urmatorul :

Se cere implementarea unei aplicatii pentru analizarea comportamentului unei persoane, datele privind acest comportament fiind inregistrate de un set de senzori.

Obiectivul principal al acestei teme este de a implementa cerintele utilizand expresii lambda si procesarea stream-urilor.

Obiectivele secundare sunt urmatoarele :

* Dezvoltarea de use-case-uri si scenarii (Capitolul 2)
* Alegerea structurilor de date (Capitolul 3)
* Impartirea pe clase, utilizarea interfetelor, evidentierea relatiilor (Capitolul 3)
* Dezvoltarea algoritmilor utilizati (Capitolul 4)
* Testarea solutiei (Capitolul 5)

**2.ANALIZA PROBLEMEI, MODELARE, SCENARII, CAZURI DE UTILIZARE**

Aplicatia trebuie sa analizeze comportamentul unei persoane, datele privind acest comportament fiind inregistrat de un set de senzori. Istoricul activitatii unei persoane este stocat in tuple : data si ora inceperii activitatii (start\_time), data si ora incheierii activitatii (end\_time) si o eticheta (activity\_label). Eticheta reprezinta activitatea pe care persoana o desfasoara si poate avea una din valorile : “Leaving”, “Toileting”, “Showering”, “Sleeping”, “Breakfast”, “Lunch”, “Snack”, “Spare\_Time/TV”, “Grooming”. Acest istoric este salvat in fisierul Activities.txt.

**3. PROIECTARE (DECIZII DE PROIECTARE, DIAGRAME UML, STRUCTURI DE DATE, PROIECTARE CLASE, INTERFETE, RELATII, PACKAGES, ALGORITMI, INTERFATA UTILIZATOR)**

Datele privind activitatea persoanei sunt stocate intr-un fisier, din care urmeaza sa fie citite si procesate prin intermediul stream-urilor, utilizand expresii lambda.

Clasele utilizate in realizarea acestei teme sunt :

1. **Clasa MonitoredData** – contine ca variabile instanta 3 obiecte de tipul String : start time-ul, end time-ul si eticheta activitatii; si un obiect de tipul Duration care retine durata totala a activitatii. Pe langa constructor, clasa contine o metoda de get pentru eticheta activitatii, suprascrie metoda toString() si contine implementarea metodelor necesare in realizarea cerintelor temei.
2. **Clasa TestTask1** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 1.
3. **Clasa TestTask2** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 2.
4. **Clasa TestTask3** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 3.
5. **Clasa TestTask4** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 4.
6. **Clasa TestTask5** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 5.
7. **Clasa TestTask6** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 6.
8. **Clasa TestTask7** – nu contine variabile instanta si nici constructor. Clasa contine doar metoda main, in cadrul careia se instantiaza un obiect de tipul MonitoredData, se realizeaza citirea din fisier, crearea listei de obiecte de tipul MonitoredData si se apeleaza metoda care indeplineste cerinta task-ului 7.

Toate clasele prezentate fac parte din pachetul ro.tuc.pt.assig\_5.

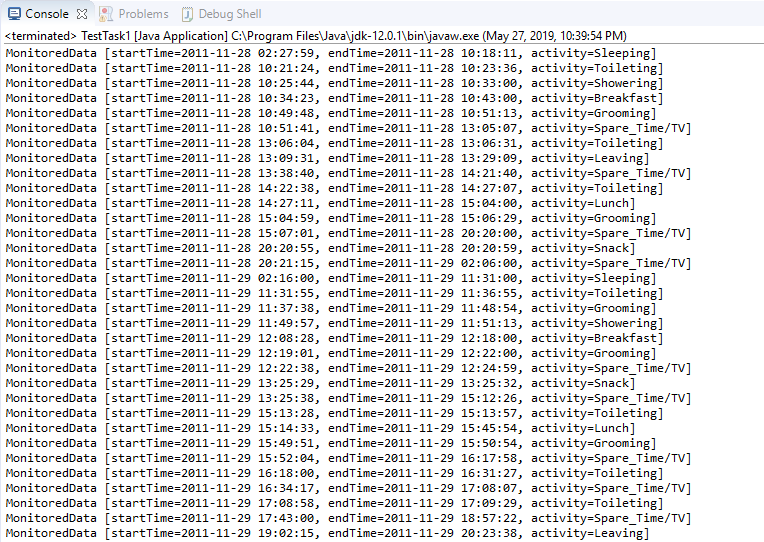
**4. IMPLEMENTARE**

Metodele implementate din clasa MonitoredData sunt :

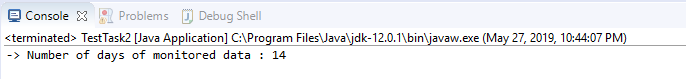
* public List<String> readFile() – realizeaza citirea din fisierul Activity.txt si returneaza o lista care contine ca elemente liniile din fisier.
* public List<MonitoredData> createList(List<String> data) – transforma lista data transmisa ca parametru intr-un stream, pe care il transforma intr-o lista, apoi proceseaza fiecare element, prin extragerea datelor de interes din fiecare linie (start time, end time si activity label). Avand aceste date, creeaza un obiect de tipul MonitoredData, pe care il introduce intr-o lista de obiecte de tipul MonitoredData, lista ce este in final returnata.
* public int countDays(List<MonitoredData> monitoredData) – transforma lista transmisa ca parametru intr-un stream si extrage din fiecare element data (an-luna-zi) din campurile pentru start time si end time, pe care le introduce intr-un set, pentru a asigura unicitatea aparitiei acelei date; metoda returneaza size-ul set-ului.
* public Map<String, Long> countHowManyTimesForEachActivity(List<MonitoredData> monitoredData) – transforma lista transmisa ca parametru intr-un stream si returneaza o mapare in care cheia este reprezentata de eticheta activitatii, iar valoarea este numarul de aparitii al activitatii in fisier.
* public String getDayFromStartTime() – returneaza data (an-luna-zi) dintr-un start time.
* public Map<String, Map<String, Long>> countHowManyTimesEachActivityEachDay(List<MonitoredData> monitoredData) - transforma lista transmisa ca parametru intr-un stream si returneaza o mapare in care cheia este reprezentata de eticheta activitatii, iar valoarea este o mapare in care cheia este reprezentata de data (an-luna-zi) din start time si valoarea este numarul de aparitii al activitatii in ziua respectiva.
* public Duration computeDuration() – calculeaza durata activitatii ca diferenta dintre end time si start time.
* public List<String> computeDurationForActivities(List<MonitoredData> monitoredData) – transforma lista transmisa ca parametru intr-un stream si returneaza o lista de String, iar elementele acestei liste contin numele activitatii si durata (pentru fiecare linie din fisier).
* public Map<String,String> computeTotalDurationForEachActivity(List<MonitoredData> monitoredData) – creeaza o mapare in care cheia este eticheta activitatii, iar valoarea este reprezentata de lista de obiecte de tipul MonitoredData care contin eticheta respectiva. Pentru fiecare pereche cheie-valoare, metoda calculeaza durata totala de pe fiecare lista. Eticheta activitatii si durata totala corespunzatoare sunt apoi adaugate intr-o mapare ce este returnata.
* public List<String> filterActivities(List<MonitoredData> monitoredData) – transforma lista transmisa ca parametru intr-un stream, filtrand informatia in functie de durata (durata trebuie sa fie mai mica de 5 minute) si creeaza o mapare in care cheia este reprezentata de eticheta activitatii, iar valoarea de numarul de activitati care au durata mai mica de 5 minute. Daca acest numar reprezinta cel putin 90% din numarul total de aparitii al activitatii respective, eticheta activitatii este adaugata intr-o lista. Metoda returneaza in final aceasta lista.

**5. REZULTATE**

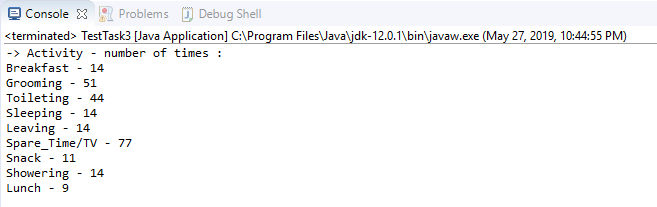
5.1.pentru task-ul 1 (doar o parte din afisare) :



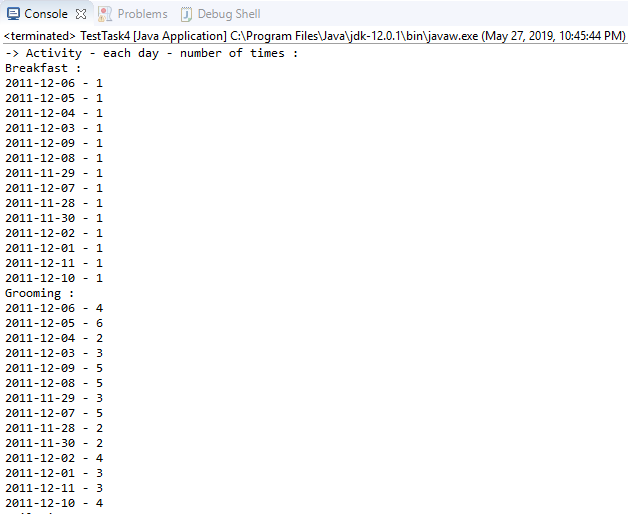
5.2.pentru task-ul 2 :



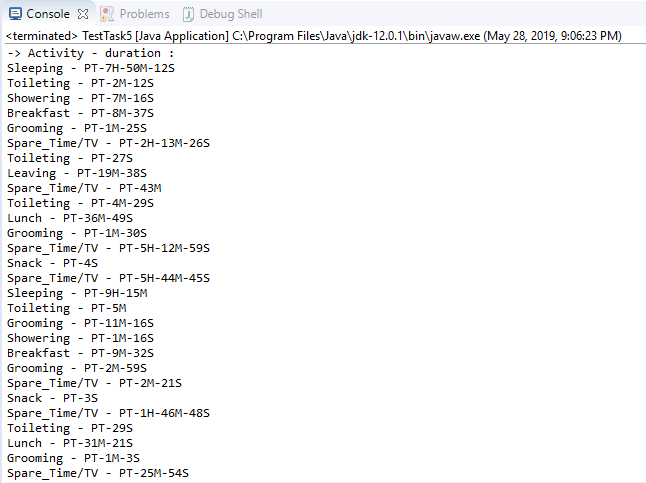
5.3.pentru task-ul 3 :



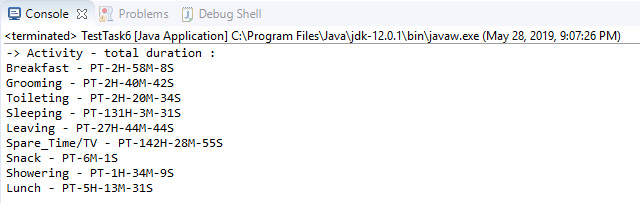
5.4.pentru task-ul 4 (doar o parte din afisare):



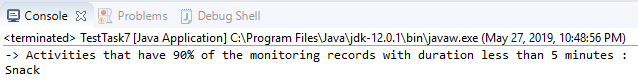
5.5.pentru task-ul 5 (doar o parte din afisare):



5.6.pentru task-ul 6 :



5.7.pentru task-ul 7 :



**6. CONCLUZII**

In concluzie, aceasta tema m-a ajutat sa imi imbunatatesc cunostiintele legate de procesarea stream-urilor si de utilizarea expresiilor lambda.

**7. BIBLIOGRAFIE**

* cursuri PT
* <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split>
* <https://docs.oracle.com/javase/8/docs/api/java/util/stream/Collectors.html>
* <https://www.baeldung.com/java-date-difference>
* <https://www.mkyong.com/java8/java-8-how-to-convert-string-to-localdate/>