**Documentatie Tema 5**

1. **Obiectivul temei**

Obiectivul acestei teme este citirea dintr-un fisier a unei liste de activitati si intervale de timp in care acestea se desfasoara si prelucrarea informatiei in functie de cerintele temei. Cerintele se rezolva utilizand streams.Fiecare task isi va face simtita prezenta prin creearea unui fisier txt si afisarea unui mesaj/ liste/ map in functie de cerinta problemei.

1. **Analiza problemei**
   1. **Cazuri de utilizare**

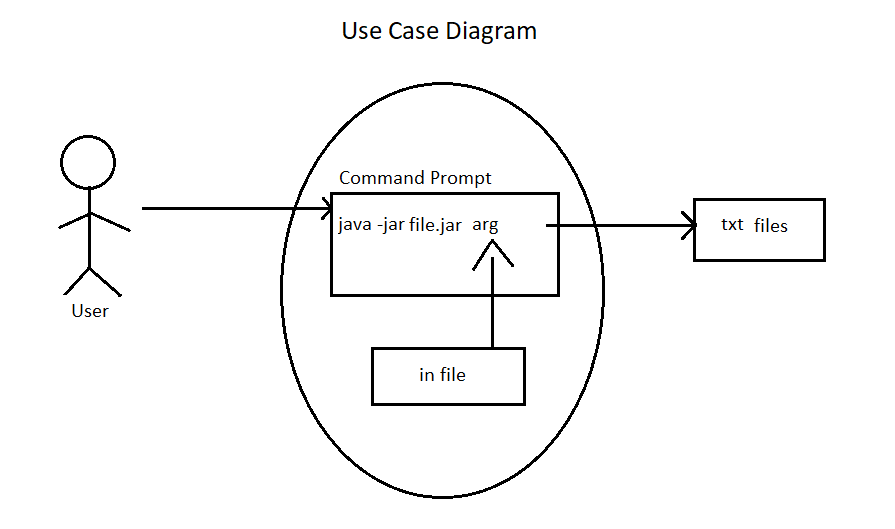
***Pasi de utilizare :***

1. Se deschide Command Prompt-ul in locatia fisierului.jar .
2. Pentru a rula se introduc urmatoarele: “ java –jar fisier.jar argument ” unde in loc de fisier se va scrie numele fisierului.jar, in cazul meu „ tasks.jar ”, iar ca argument va primi un fisier: fisierul din care se vor citi datele de intrare, fisier denumit „ Activities.txt ” .
3. Activities.txt contine 3 coloane importante: prima coloana reprezinta data si momentul de timp la care se incepe activitatea, a doua coloana reprezinta data si momentul de timp la care se incheie activitatea, iar ultima coloana contine numele activitatii. Pentru ca programul sa functioneze trebuie sa se respecte un anumit format. Data este scrisa sub forma ” yyyy-mm-dd ”, iar momentul de timp va fi separat de data printr-un spatiu si se va scrie sub forma „ hh:mm:ss ”. Aceleasi conditii se vor aplica atat pentr start time cat si pentru end time . Coloanele vor fi separate prin 2 sau mai multe spatii . Numele activitatii se va scrie cu litera mare, iar daca exista mai multe activitati de acelasi tip, se va respecta formatul ales initial .
4. Dupa ce s-a verificat formatul fisierului si ca numele acestuia a fost scris corect, se ruleaza programul .
5. Se urmaresc documentele txt generate si se verifica daca s-au indeplinit cerintele problemei in mod corespunzator .

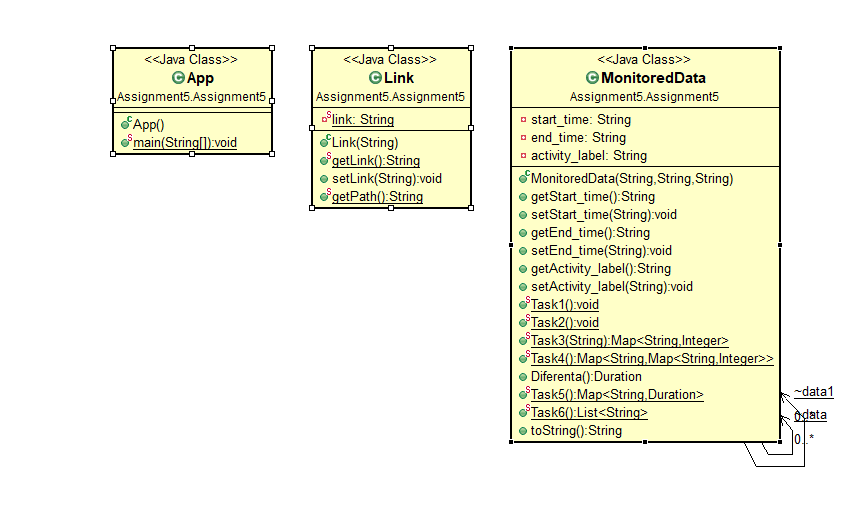
***Erori posibile :***

1. Una dintre erorile posibile este scrierea incorecta a numelui fisierului din care se doreste sa se citeasca.
2. O alta eroare posibila este scrierea incorecta a numelui fisierului .jar .

**2.2 Diagrama USE CASE**

****

**3. Proiectare**

**3.1. Diagrama UML**

**3.2. Explicatii clase**

* Clasa App : - este clasa care contine metoda main si in care au loc apelurile task-urilor.
* Clasa Link : - este clasa prin care se transmite argumentul primit ca parametru pentru a putea deschide fisierul “ Activities.txt ” .
* Clasa MonitoredData :- este clasa in care stocam start\_time, end\_time, activity\_label intr-o colectie de obiecte de tip MonitoredData si in care se afla toate metodele care executa task-urile ( 1, 2, 3, 4, 5, 6) .

**4. Explicatii clase**

* Clasa App : - contine metoda main. Aici deschidem fisierele.txt pentru a putea scrie in ele. Tot aici cream un obiect de tip Link pentru a putea transmite adresa fisierului “ Activities.txt “. Nu am creat niciun obiect de tip MonitoredData , deoarece metodele din clasa MonitoredData sunt statice. Dupa ce se scrie in fisier, acestea se vor inchide pentru a nu fi prezente memory leaks .
* Clasa Link :- are ca si variabila instanta un String in care se va retine argumentul primit ca parametru in urma rularii. Pe langa sett-eri si get-eri mai avem o metoda numita getPath() care returneaza un String, reprezentand calea spre fisier .
* Clasa MonitoredData :- in aceasta clasa avem ca si variabile instanta end\_data, start\_data si activity\_label . Am creat sett-eri, gett-eri si un constructor pentru variabile. Avem o variabila globala de tipul static ArrayList <MonitoredData> in care vom stoca informatia din fisier. Primul task este indeplinit de metoda Task1(). Task-ul 1 presupune citirea din fisier ,impartirea informatiei in 3 (start\_time, end\_time , activity\_time) si introducerea acestora in lista de obiecte de tip ArrayList <MonitoredData> pe care am declarat-o global. Mai intai pregatim fisierul pentru scriere, dupa care separam si introducem in lista elementele cu ajutorul instructiunii: rows.map( x -> x.split(“ \\s{2,} ” )).forEach( x -> *data*.add( **new** MonitoredData ( x[0] , x[1] , x[2] ))). Se scrie in fisier si se inchide. Task-ul numarul 2 presupune numararea zilelor distincte care se afla in fisier. Pentru a realiza acest lucru am creat o lista de String-uri in care am introdus doar data calendaristica pentru start\_time cu ajutorul instructiunii: *data*.stream().map( x -> x.getStart\_time( ).split( " [\\s](file:///\\s)+" ) ).forEach( x -> start.add( x[0] ) ). Apoi cu ajutorul aceste liste nou formate voi numara elementele distincte: start.stream( ).distinct( ).count( ), dupa care inchidem fisierul. Task-ul 3 cere sa numar de cate ori apare fiecare activitate pe parcursul intregului interval de timp din fisier. Aceasta metoda va returna o structura de tip Map<String,Integer>. Pentru inceput cream o noua lista de string - uri pentru a pune toate activitatile distincte in ea cu ajutorul instructiunii: *data*.stream( ).map( x-> x.getActivity\_label( ).replaceAll( [\\s+](file:///\\s+) , "" )).distinct( ).forEach( x -> activities.add( x )). Urmeaza o conditie, in functie e aceasta se ia o decizie. Daca string-ul dat ca si parametru este “count”, atunci se va executa instructiunea: activities.stream( ).forEach(x -> map1.put( x, ( **int** ) *data*.stream().filter(z-> z.getActivity\_label( ).replaceAll( [\\s+](file:///\\s+) , "" ).equalsIgnoreCase( x )).count( ))), care pentru fiecare activitate aduna 1 si pune suma in map. A doua ramura pentru if o voi explica la Task-ul 4 pentru ca acolo am folosit-o. Se scrie in fisier si se inchide. Al 4-lea task este asemantor cu al 3-lea, doar ca trebuie sa afisam pentru fiecare zi cate activitati s-au facut. Vom proceda ca la task-ul anterior. Vom creea acea lista de string-uri in care vom introduce aceleasi elemente. Aceasta metoda va returna o structura de forma Map< String ,Map < String ,Integer > >, unde primul String va reprezenta ziua, al doilea va reprezenta activitatea, iar intregul va reprezenta numarul de aparitii. Pentru fiecare zi vom alege activitatile distincte pe care le numaram. Ramura de pe else al if-ului din task-ul anterior va returna o structura de tip Map < String, Integer > care va reprezenta numele activitatii, respectiv numarul de aparatii in acea zi. Task-ul 5 va calcula perioda petrecuta pentru fiecare activitate pe tot parcursul parioadei din fisier. Pentru a realiza acest lucru m-am folosit de tipul de data Duration cu ajutorul careia am obtinut diferenta de timp dintre start\_time si and\_time : Duration duration = Duration.*between* ( LocalDateTime.*parse* ( start\_time,formatter ), LocalDateTime.*parse*( end\_time,formatter ) ). Clasa Diferenta() returneaza duration. Cu ajutorul instructiunii: *data*.stream( ).collect ( Collectors.*toMap*( MonitoredData::getActivity\_label, MonitoredData::Diferenta, Duration::plus)) am creat o structura de tip map care are ca String numele activitatii, iar ca duration suma tuturor timpilor pentru fiecare activitate. Ultimul task are ca cerinta: afisarea listei de activitati care se desfasoara in proportie de 90% in mai putin de 5 minute. Pentru acest task am selectat activitatile separate, apoi cele pe care le gaseam cu durata mai mica de 5 minute le adaugam intr-o lista, lista care urmeaza sa fie returnata de catre functie.

**5.Rezultate**

Avem fisierul “ Activities.txt ”, care are urmatorul continut:

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

2011-11-28 15:04:59 2011-11-28 15:06:29 Grooming

2011-11-28 15:07:01 2011-11-28 20:20:00 Spare\_Time/TV

2011-11-28 20:20:55 2011-11-28 20:20:59 Snack

2011-11-28 20:21:15 2011-11-29 02:06:00 Spare\_Time/TV

2011-11-29 02:16:00 2011-11-29 11:31:00 Sleeping

2011-11-29 11:31:55 2011-11-29 11:36:55 Toileting

2011-11-29 11:37:38 2011-11-29 11:48:54 Grooming

2011-11-29 11:49:57 2011-11-29 11:51:13 Showering

2011-11-29 12:08:28 2011-11-29 12:18:00 Breakfast

2011-11-29 12:19:01 2011-11-29 12:22:00 Grooming

2011-11-29 12:22:38 2011-11-29 12:24:59 Spare\_Time/TV

2011-11-29 13:25:29 2011-11-29 13:25:32 Snack

2011-11-29 13:25:38 2011-11-29 15:12:26 Spare\_Time/TV

2011-11-29 15:13:28 2011-11-29 15:13:57 Toileting

2011-11-29 15:14:33 2011-11-29 15:45:54 Lunch

2011-11-29 15:49:51 2011-11-29 15:50:54 Grooming

2011-11-29 15:52:04 2011-11-29 16:17:58 Spare\_Time/TV

2011-11-29 16:18:00 2011-11-29 16:31:27 Toileting

2011-11-29 16:34:17 2011-11-29 17:08:07 Spare\_Time/TV

2011-11-29 17:08:58 2011-11-29 17:09:29 Toileting

2011-11-29 17:43:00 2011-11-29 18:57:22 Spare\_Time/TV

2011-11-29 19:02:15 2011-11-29 20:23:38 Leaving

2011-11-29 20:28:00 2011-11-30 01:19:47 Spare\_Time/TV

2011-11-30 01:22:33 2011-11-30 10:07:31 Sleeping

2011-11-30 10:11:07 2011-11-30 10:13:59 Toileting

2011-11-30 10:14:45 2011-11-30 10:18:33 Showering

2011-11-30 10:22:59 2011-11-30 10:35:00 Breakfast

2011-11-30 10:39:30 2011-11-30 10:41:13 Grooming

2011-11-30 10:41:38 2011-11-30 13:04:59 Spare\_Time/TV

2011-11-30 13:05:27 2011-11-30 13:05:31 Snack

2011-11-30 13:05:38 2011-11-30 14:09:34 Spare\_Time/TV

2011-11-30 14:11:16 2011-11-30 14:11:48 Toileting

2011-11-30 14:13:45 2011-11-30 14:34:09 Spare\_Time/TV

2011-11-30 14:39:42 2011-11-30 15:00:24 Lunch

2011-11-30 15:11:48 2011-11-30 16:39:11 Spare\_Time/TV

2011-11-30 16:40:01 2011-11-30 16:40:24 Toileting

2011-11-30 16:41:05 2011-11-30 16:41:09 Snack

2011-11-30 16:41:15 2011-11-30 18:00:38 Spare\_Time/TV

2011-11-30 18:01:44 2011-11-30 18:01:47 Toileting

2011-11-30 18:01:54 2011-11-30 18:43:20 Spare\_Time/TV

2011-11-30 18:57:08 2011-11-30 18:57:34 Toileting

2011-11-30 18:57:39 2011-11-30 19:37:10 Spare\_Time/TV

2011-11-30 19:37:36 2011-11-30 19:38:01 Toileting

2011-11-30 19:46:44 2011-11-30 20:05:12 Leaving

2011-11-30 20:13:39 2011-11-30 20:13:48 Grooming

2011-11-30 20:14:03 2011-12-01 01:34:27 Spare\_Time/TV

2011-12-01 01:37:52 2011-12-01 10:50:45 Sleeping

2011-12-01 10:55:03 2011-12-01 10:55:32 Grooming

2011-12-01 10:58:37 2011-12-01 11:12:22 Showering

2011-12-01 11:17:05 2011-12-01 11:29:49 Breakfast

2011-12-01 11:30:06 2011-12-01 14:29:02 Spare\_Time/TV

.

.

.

* Primul Task presupune citirea, separarea coloanelor in String-uri si introducerea lor in

niste fisiere. In urma rularii fisierul “Task1.txt” este de forma :

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

2011-11-28 15:04:59 2011-11-28 15:06:29 Grooming

2011-11-28 15:07:01 2011-11-28 20:20:00 Spare\_Time/TV

2011-11-28 20:20:55 2011-11-28 20:20:59 Snack

2011-11-28 20:21:15 2011-11-29 02:06:00 Spare\_Time/TV

2011-11-29 02:16:00 2011-11-29 11:31:00 Sleeping

2011-11-29 11:31:55 2011-11-29 11:36:55 Toileting

2011-11-29 11:37:38 2011-11-29 11:48:54 Grooming

2011-11-29 11:49:57 2011-11-29 11:51:13 Showering

2011-11-29 12:08:28 2011-11-29 12:18:00 Breakfast

2011-11-29 12:19:01 2011-11-29 12:22:00 Grooming

2011-11-29 12:22:38 2011-11-29 12:24:59 Spare\_Time/TV

.

.

.

Se poate observa ca in urma separarii am putut modifica aranjarea informatiei in fisier.

* Task-ul numarul 2 prespune numararea zilelor distincte. In urma rularii fisierul arata asa:” In fisier apar 14 zile distincte”. Zilele pornesc din data de 28 noiembrie pana in data de 11 decembrie. Printr-un simplu calcul se deduce usor ca afirmatia rezultata este adevarata.
* Task-ul numarul 3 presupune afisarea activitatilor distincte si numarul de aparitii ale acestora pe tot parcursul perioadei. In urma rularii continutul fisierului este astfel:

Breakfast: 14

Toileting: 44

Grooming: 51

Sleeping: 14

Leaving: 14

Spare\_Time/TV: 77

Showering: 14

Snack: 11

Lunch: 9

Asta inseamna ca pe parcursul a 14 zile persoana respective a luat micul dejun de 14 ori, a dormit de 14 ori si asa mai departe.

* Task-ul numarul 4 presupune afisarea zilelor distincte si a numarului de activitati distincte realizate in ziua respective. In urma rularii, fisierul are continutul:

2011-12-06:{Breakfast=1, Toileting=3, Grooming=4, Sleeping=1, Leaving=0, Spare\_Time/TV=5, Showering=1, Snack=1, Lunch=1}

2011-12-05:{Breakfast=1, Toileting=5, Grooming=6, Sleeping=1, Leaving=2, Spare\_Time/TV=7, Showering=1, Snack=1, Lunch=1}

2011-12-04:{Breakfast=1, Toileting=4, Grooming=2, Sleeping=1, Leaving=1, Spare\_Time/TV=6, Showering=1, Snack=2, Lunch=0}

2011-12-03:{Breakfast=1, Toileting=2, Grooming=3, Sleeping=1, Leaving=1, Spare\_Time/TV=4, Showering=1, Snack=0, Lunch=0}

2011-12-09:{Breakfast=1, Toileting=2, Grooming=5, Sleeping=1, Leaving=2, Spare\_Time/TV=6, Showering=1, Snack=0, Lunch=0}

2011-11-29:{Breakfast=1, Toileting=4, Grooming=3, Sleeping=1, Leaving=1, Spare\_Time/TV=6, Showering=1, Snack=1, Lunch=1}

2011-12-08:{Breakfast=1, Toileting=1, Grooming=5, Sleeping=1, Leaving=1, Spare\_Time/TV=4, Showering=1, Snack=0, Lunch=0}

2011-11-28:{Breakfast=1, Toileting=3, Grooming=2, Sleeping=1, Leaving=1, Spare\_Time/TV=4, Showering=1, Snack=1, Lunch=1}

2011-12-07:{Breakfast=1, Toileting=6, Grooming=5, Sleeping=1, Leaving=1, Spare\_Time/TV=8, Showering=1, Snack=2, Lunch=1}

2011-11-30:{Breakfast=1, Toileting=6, Grooming=2, Sleeping=1, Leaving=1, Spare\_Time/TV=8, Showering=1, Snack=2, Lunch=1}

2011-12-02:{Breakfast=1, Toileting=3, Grooming=4, Sleeping=1, Leaving=0, Spare\_Time/TV=7, Showering=1, Snack=1, Lunch=1}

2011-12-01:{Breakfast=1, Toileting=2, Grooming=3, Sleeping=1, Leaving=1, Spare\_Time/TV=6, Showering=1, Snack=0, Lunch=1}

2011-12-11:{Breakfast=1, Toileting=2, Grooming=3, Sleeping=1, Leaving=0, Spare\_Time/TV=3, Showering=1, Snack=0, Lunch=1}

2011-12-10:{Breakfast=1, Toileting=1, Grooming=4, Sleeping=1, Leaving=2, Spare\_Time/TV=3, Showering=1, Snack=0, Lunch=0}

Este asemanator cu task-ul anterior doar ca de aceasta data facem acest lucru pentru fiecare zi in parte.

* Task-ul numarul 5 presupune calcularea perioadei petrecute pentru realizarea fiecarei activitati pe tot parcursul perioadei de timp. In urma rularii, fisierul arata astfel:

Leaving : 27 hours 44 minutes 44 seconds

Breakfast : 2 hours 58 minutes 8 seconds

Sleeping: 131 hours 3 minutes 31 seconds

Snack : 0 hours 6 minutes 1 seconds

Grooming : 2 hours 40 minutes 42 seconds

Showering : 1 hours 34 minutes 9 seconds

Spare\_Time/TV: 142 hours 28 minutes 55 seconds

Toileting : 2 hours 20 minutes 34 seconds

Lunch: 5 hours 13 minutes 31 seconds

Asta inseamna ca persoana respective a dormit in 14 zile timp de 131 de ore 3 minute si 31 de secunde.

* Task-ul numarul 6 presupune afisarea listei de activitati care se desfasoara in proportie de 90% in mai putin de 5 minute. In urma rularii programului, fisierul are continutul:

Toileting

Grooming

Snack

Showering

Spare\_Time/TV

Breakfast

6.Concluzii

Aceasta tema m-a facut sa realizez cat de usoara poate fi programare daca ai anumite cunostinte. Stream-urile au foarte multe functii de sortare, aranjare si codul este foarte compact. Mi-am fixat anumite cunostiinte legate de functia split(). Din punctual meu de vedere este mult mai elegant sa programezi cu stream-uri atunci cand este posibil acest lucru si sunt convins ca o sa folosesc si de acum incolo pentru urmatoarele proiecte in java.

7.Bibliografie

* <https://www.geeksforgeeks.org/stream-map-java-examples/>
* <https://docs.oracle.com/javase/8/docs/api/java/time/LocalDateTime.html>
* <https://docs.oracle.com/javase/tutorial/datetime/iso/period.html>
* https://examples.javacodegeeks.com/java-stream-how-to-use-java-8-streams/