**- TEMA V -**

**ANALYZING THE BEHAVIOR OF A PERSON RECORDED BY A SET OF SENSORS**

Marc Andrei – Aurel

Grupa 30224

Laborator – Tehnici de Programare

Cuprins

1. Obiectivul temei
2. Analiza problemei
3. Proiectare
4. Implementare
5. Rezultate
6. Concluzii
7. Bibliografie

Obiectivul temei

Stream API – ul este folosit pentru procesarea unor colectii de obiecte. Un stream este o secventa de obiecte care suporta mai multe metode ce pot fi inlantuite (pipelined) pentru a produce rezultatul dorit. Stream-ul nu este o structura de date, in schimb ia input de la colectii, array-uri si canale input / output. Acestea nu schimba structura de date originala, ci doar furnizeaza rezultatul dorit prin metodele inlantuite. Fiecare operatie este executata in particular si returneaza un stream. Operatiile terminale marcheaza sfarsitul stream-ului si returneaza rezultatul. Unele operatii intermediare sunt: map, care este folosita pentru a mapa obiectele din colectie la alte obiecte, luand in considerare Predicate-ul pasat ca argument; filter, care este folosita pentru selectia elementelor, luand in considerare Predicate-ul pasat ca argument; sorted, care este folosita pentru sortarea stream-urilor. Unele operatii terminale sunt: collect, care este folosita pentru returnarea rezultatului operatiilor intermediare efectuate in stream; forEach, care este folosita pentru iterarea prin fiecare element al stream-ului; reduce, folosit pentru reducerea elementelor stream-ului la o singura valoare si are ca parametru un BinaryOperator.

Lambda Expressions sunt introduse din Java 8 si sunt primul pas al Java spre programarea functionala. Lamda Expression este, asadar, o functie care poate fi creata fara sa apartina unei clase si poate fi pasata ca si cum ar fi un obiect si poate fi executata. Sunt folosite, de obicei, pentru implementarea event listener-ilor sau in programarea functionala impreuna cu stream-urile.

Obiectivul temei este dezvoltarea unei aplicatii care analizeaza comportamentul unei persoane inregistrate de un set de senzori. Log-ul activitatilor unei persoane este stocat dupa urmatoarele criterii: timpul de inceput al activitatii, timpul de sfarsit al activitatii si activitatea propriu – zisa. Pentru a le stoca, ales sa folosesc Date pentru timpul de inceput si sfarsit al activitatii, respectiv String pentru numele activitatii. Printre activitatile persoanei monitorizate, se regasesc: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner, Snack, Spare\_Time/TV, Grooming. Datele sunt impartite pe mai multe zile, fiecare activitate avand randul ei propriu in fisierul Activities.txt. Cerintele temei sunt urmatoarele:

1. Definirea clasei MonitoredData, cu 3 campuri: start time, end time si activitatea propriu – zisa. Citirea datelor din fisierul dat, Activities.txt folosind stream-uri si impartirea fiecarei linii in 3 parti, fiecare corespunzand la cate un camp din clasa MonitoredData, si crearea unei liste de MonitoredData din fisierul dat.
2. Sa se numere cate zile apar in fisier.
3. Sa se numere de cate ori apare fiecare activitate pe durata intregii perioade de monitorizare. Ca rezultat, se va returna un map cu activitatile si numarul de aparitii a fiecareia.
4. Sa se numere de cate ori apare fiecare activitate pentru fiecare zi a perioadei de monitorizare.
5. Sa se calculeze durata fiecarei activitati (fiecarei linii).
6. Sa se calculeze durata totala a fiecarei activitati, pe intreaga perioada de monitorizare.
7. Sa se filtreze activitatile care au 90% din istoricul monitorizarii, o durata de mai putin de 5 minute.

Analiza problemei

Tema consta in implementarea unei aplicatii ce foloseste stream-uri si lambda expressions, pentru a analiza comportamentul si activitatile unei persoane, inregistrate si scrise intr-un fisier. Pentru fiecare cerinta, vom incerca rezolvarea ei utilizand stream-uri si lambda expressions. Pentru a rezolva cerintele, vom crea clasa MonitoredData, care va contine campurile scrise si in fisier, si anume data si timpul la care incepe, respectiv se termina activitatea, si activitatea propriu – zisa. Acestea le vom defini ca atribute, de tipul Date, respectiv String. Ca metode, vom avea getter-e si setter-e pentru a avea acces la campurile obiectului de tip Monitored Data, atributele sale fiind private, conform principiilor OOP. De asemenea, vom implementa si metodele toString() si compareTo().

Pentru rezolvarea cerintelor, vom crea un pachet nou, iar in acest pachet, vom crea o clasa noua pentru fiecare cerinta, pentru a fi mai usor de deosebit. Clasa MonitoredDataRetriever realizeaza prima cerinta, adica parsarea fisierului. DaysCounter numara zilele care sunt incluse in perioada de monitorizare (cerinta a doua). ActivityCounter numara de cate ori apare fiecare activitate pe intreaga durata a monitorizarii (cerinta a treia). Clasa ActivitiesPerDayCounter numara de cate ori apare fiecare activitate pe fiecare zi a perioadei de monitorizare (cerinta a patra). Clasa DurationCalculator calculeaza durata fiecarei activitati (cerinta a cincea). Clasa EntireActivityDurationCalculator calculeaza durata totala a unei activitati, pe intreaga perioada a monitorizarii (cerinta a sasea). Clasa ActivityFilter filtreaza activitatile care au 90% din istoricul monitorizarii, o durata mai mica de 5 minute (cerinta a saptea).

Analiza problemei si implementare

Tema consta in implementarea unei aplicatii ce foloseste stream-uri si lambda expressions, pentru a analiza comportamentul si activitatile unei persoane, inregistrate si scrise intr-un fisier.

In clasa MonitoredData, am definit clasa in care vom stoca datele luate din fisier. Am definit trei atribute: timpul de inceput al activitatii, de tip Date, timpul de sfarsit al activitatii, de tip Date, si numele activitatii, de tip String. Ca metode, am generat getter-e si setter-e si am implementat si suprascris metoda toString si metoda compareTo.

In clasa MonitoredDataRetriever, am implementat rezolvarea primei cerinte. Am deschis fisierul si am luat fiecare linie si am impartit-o in trei si am luat campurile necesare, apoi salvandu-le intr-o lista cu elemente de tip Monitored Data.

In clasa DaysCounter, am implementat rezolvarea cerintei a doua. Am utilizat metoda collect, am grupat zilele dupa data si apoi am luat size-ul acelui map pentru a returna numarul zilelor monitorizate.

In clasa ActivityCounter am implementat rezolvarea cerintei a treia. Am utilizat metoda collect, am grupat dupa numele activitatii si am numarat cu counting(), returnand un Map < String, Long >.

In clasa ActivitiesPerDayCounter am implementat rezolvarea cerintei a patra. Am utilizat metoda collect si am grupat dupa anul, luna si ziua pentru a nu tine cont si de ora la care se intampla activitatea respectiva.

In clasa DurationCalculator am implementat rezolvarea cerintei a cincea. Am luat fiecare linie a fisierului si am parsat datele de pe linia respectiva. Apoi, am creat o noua data care este egala cu diferenta dintre datele date in fisier, si am pus-o in map cu index-ul liniei.

In clasa EntireActivitiesDurationCalculator am implementat rezolvarea cerintei a sasea. Am colectat toate activitatile si pentru fiecare am calculat diferenta, pe rand, dintre fiecare activitate. Am returnat un Map < String, Time >.

In clasa ActivityFilter am implementat rezolvarea cerintei a saptea.

Diagrama Use Case, care prezinta operatiile ce pot fi efectuate de un user, si anume sa ruleze fiecare cerinta, introducand index-ul, si sa vizualizeze analizele comportamentului pe perioada monitorizata.

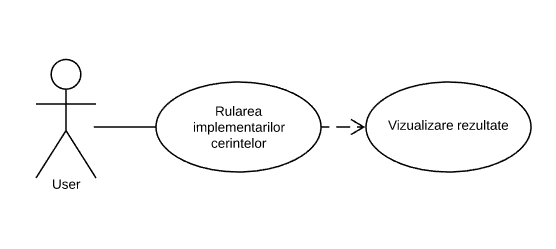
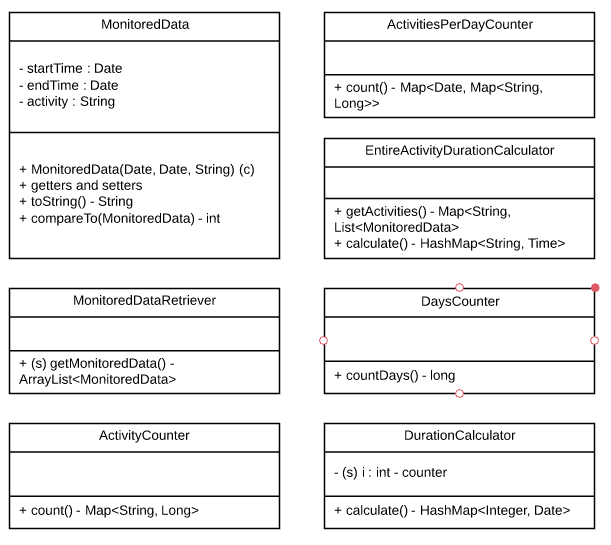


Diagrama UML, care prezinta structura claselor din proiect. Fiecare clasa, in afara de MonitoredData, indeplinesc o cerinta din tema. Sarcina fiecarei clase este prezentata mai sus in documentatie, la analiza problemei si implementare.



Rezultate

Pentru testarea aplicatiei, am printat rezultatele obtinute si am analizat cateva activitati la fiecare cerinta, pentru a garanta corectitudinea rezultatelor. La implementarea fiecarei cerinte, am testat amanuntit functionalitatea sa.

Concluzii

In aplicatie s-a realizat implementarea unui sistem ce analizeaza comportamentul uneoi persoane, inregistrate de un set de senzori. A fost nevoita folosirea stream-urilor si a expresiilor lambda, aceasta tema fiind foarte utila pentru cunoasterea acestor aspecte importante ale programarii orientate pe obiecte si a limbajului de programare Java. Ca dezvoltare ulterioara, poate fi luata in considerare posibilitatea adaugarii altor activitati in log, direct din aplicatie, precum si o interfata mai prietenoasa pentru utilizator.

Bibliografie

<https://www.lucidchart.com> – pentru desenarea diagramei use case si pentru desenarea diagramei UML a claselor

<https://www.geeksforgeeks.org/stream-in-java/>

<https://www.tutorialspoint.com/java8/java8_lambda_expressions.htm>

<https://stackoverflow.com/>

<http://tutorials.jenkov.com/java/lambda-expressions.html>

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