Documentatie Tema 5

Tehnici de programare Lambda Expressions si Stream Processing

Nume: Balint

Prenume: Simina Monica

Grupa: 30226

Univestitatea Tehnica din Cluj-Napoca,

Facultatea de Automatica si Calculatoare.

Cuprins:

Obiectivul temei

Luati in considerare sarcina de a analiza comportamentul unei persoane inregistrate de un set de senzori. Jurnalul istoric al activitatii persoanei este stocat ca tuple (start time, end time, activity label), unde start time si end time reprezinta data si ora in care fiecare activitate a inceput si s-a incheiat in timp ce eticheta de activitate reprezinta tipul de activitate realizat de persoana: plecare, toaletare, dus, somn, mic dejun, pranz, cina, gustari, timp liber / TV, ingrijire. Scrieți un program Java 1.8 folosind expresii lambda și procesarea fluxurilor pentru a face sarcinile definite de mai jos:

Definiti o clasa Monitored Data cu 3 campuri: ora de incepere, ora de incheiere și activitatea ca string. Cititi datele din fisierul Activity.txt folosind fluxurile și impartiti fiecare rând în 3 părți: start time, end time si eticheta de activitate si sa creati o lista de obiecte de tip Monitored Data.

Aflati cate zile de date monitorizate apar în jurnal.

Numarati de cate ori a aparut fiecare activitate pe intreaga perioada de monitorizare. Returnati un map de tip <String, Int> reprezentand maparea activitatilor la numarul lor.

Generati o structura de date de tipul Map <Integer, Map <String, Integer >> care contine numarul de activitati pentru fiecare zi a jurnalului (sarcina numărul 2 aplicata pentru fiecare zi a jurnalului) si scrieti rezultatul intr-un fisier text.

Determinati o structura de date a formularului Map <String, Date Time> pe care o mapati pentru fiecare activitate durata totala calculata in perioada de monitorizare. Filtrați activitatile cu o durata totala mai mare de 10 ore. Scrieti rezultatul într-un fisier text.

Filtrati activitatile care au 90% din probele de monitorizare cu durata mai putin de 5 minute, colectati rezultatele intr-o lista <String> care contine numai nume distincte de activitate si scrieti rezultatul intr-un fișier text.

Analiza problemei

Programul in sine nu este unul complicat, mai ales avand la dispozitie toate resursele necesare, iar obiectivul acestui proiect este acela de a intelege mult mai bine utilizarea unor tehnici de programare de tipul lambda expressions si streams processing. Abilitatea unei clase Java de a examina si manipula datele chiar din interiorul sau, nu pare sa sune a a fi ceva extrem de complicat sau elaborat, dar in alte limbaje de programare aceasta trasatura lipseste sau chiar nu exista. De exemplu, intr-un program care utilizeaza limbaje precum: Pascal, C, C++ este imposibila obtinerea unor informatii referitoare la functile sau procedurile definite in interiorul programului.

Aplicatia este una simiplista si functioneaza in felul urmator: apelata, metoda task1 afiseaza de cate ori a aparut fiecare activitate pe durata monitorizarii; metoda task2 numara de cate ori a aparut fiecare activitate pe intreaga perioada de monitorizare si o scrie in fisierul task_2; metoda task3 genereaza o structura de date de tipul Map <Integer, Map <String, Integer >> care contine numarul de activitati pentru fiecare zi a jurnalului (sarcina numarul 2 aplicata pentru fiecare zi a jurnalului). O scrie in fisierul task_3; metoda task4 genereaza o structura de date a formatului Map <String, Date Time> care e mapata pentru fiecare activitate durata totala calculata in perioada de monitorizare. Sunt filtrate activitatile cu o durata totala mai mare de 10 ore. O va scrie in fisierul task_4.

Erori: -se folosesc blocuri try catch in cazul in care un fisier in care se doreste scrierea nu exista sau nu se poate crea; de asemenea se folosesc blocuri try catch pentru deschiderea fisierului din care se citesc activitatie monitorizate.

Proiectare si implementare

Structuri de date:

-se folosesc structurile de date cerute: List < MonitoredData >, Map <String, Integer>, Map < Integer, Map <String, Integer> >, Map <String, DateTime>, List <String >;

Proiectare clase: -aplicatia este alcatuita dintr- un pachet si 4 clase:

Clasa Monitored Data: reprezinta clasa de baza a aplicatiei, are ca attribute ora si data la care a inceput activitatea, ora si data la care s-a sfarsit activiatea si activitatea care s-a efectuat in acest interval; are metode de get si set si suprascrierea metodei to String;

Clasa Operations: clasa in care se implementeaza rezolvarile cerintelor de mai sus: Metoda task1: returneaza cate zile de date monitorizate apar in jurnal. Metoda task2: numara de cate ori a aparut fiecare activitate pe intreaga perioada de monitorizare. Returneaza un map de tip <String, Int> reprezentand maparea activitatilor la numarul lor. Metoda task3: genereaza o structura de date de tipul Map <Integer, Map <String, Integer >> care contine numarul de activitati pentru fiecare zi a jurnalului (sarcina numarul 2 aplicata pentru fiecare zi a jurnalului). Metoda task 4: returneaza o structura de date Map <String, Date Time> pe care o mapeza pentru fiecare activitate durata totala calculata in perioada de monitorizare. Filtreaza activitatile cu o durata totala mai mare de 10 ore.

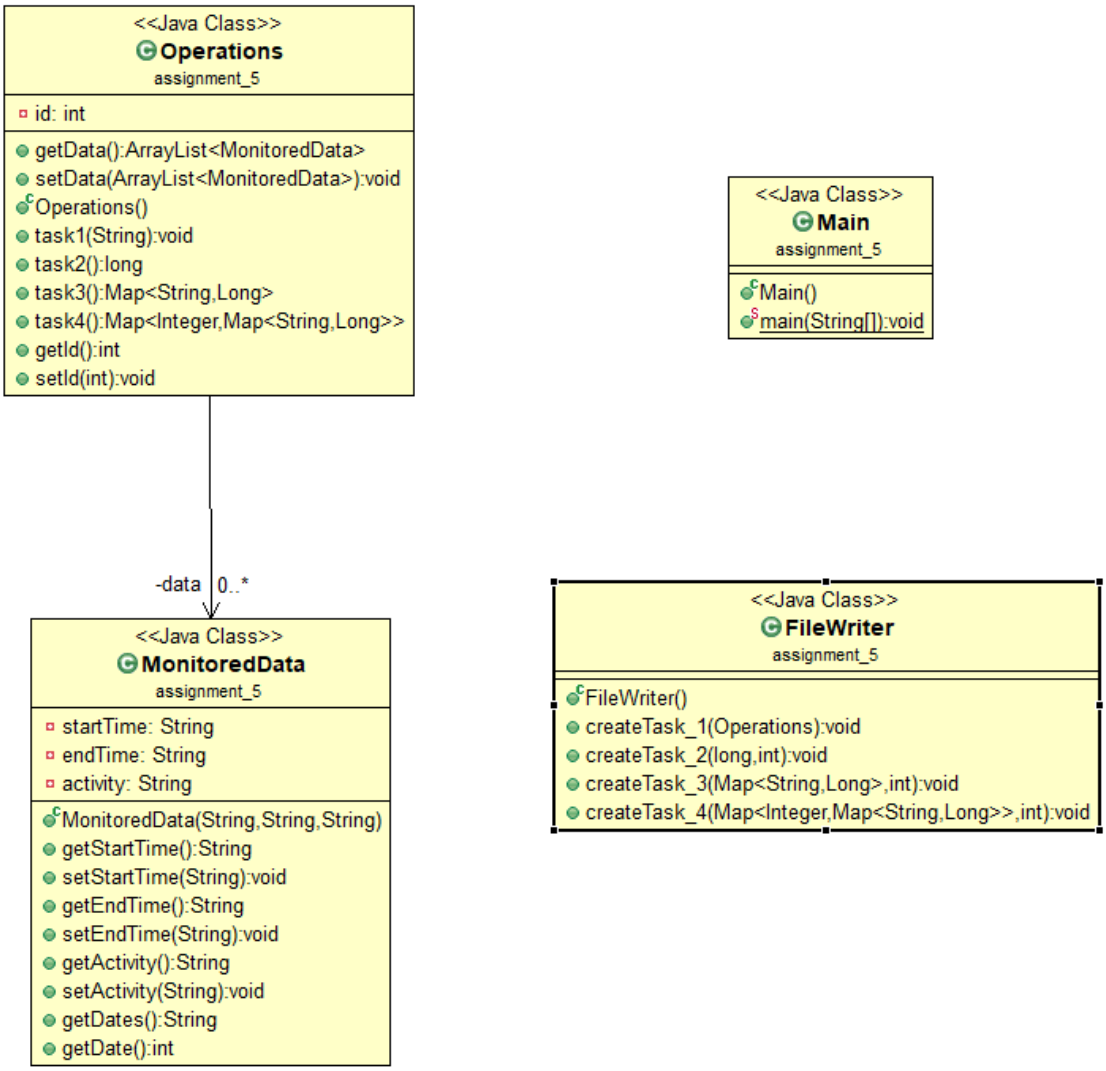
Clasa FileWriter unde se afla 4 metode care sunt apelate pentru scrierea in fisier a rezultatelor celor 4 task-uri prezentate mai sus. In clasa FileWriter se utilizeaza tipul de date print writer pentru implementarea celor 4 metode care vor fi apelate in clasa Main:

- `createTask_1(Operations o);`
- `createTask_2(long i,int id);`
- `createTask_3(Map<String, Long> result, int id);`
- `createTask_4(Map<Integer, Map<String, Long>> result, int id);`

Clasa Main: clasa care contine metoda static main care porneste aplicatia;

Am ales utilizarea colectiilor in rezolvarea unor cerinte. Fiecare utilizator are la dispozitie mai multe optiuni pe care le poate alege oricand.

In continuare voi prezenta diagrama UML a proiectului:



Implementare:

-metoda task1 se ocupa de parsarea fisieului de intrare si de separarea servicilor.

-metoda task2: List<String> dates =data.stream().map(MonitoredData::getDays).collect(Collectors.*toList*()): salveaza in lista dates toate zilele citite din fisier cu ajutorul metodei get Days din clasa Monitored Data, care returneaza cu ajutorul metodei sub string;

-metoda task3: Map<String, Long>result = data.stream().collect(Collectors.*groupingBy* (MonitoredData: :getActivity, Collectors.*counting*())): grupeaza pentru fiecare activitate luata cu ajutorul metodei get Activity din clasa Monitored Data, iar pentru fiecare activitate se mapeaza numarul de aparitii in fisierul text;

-metoda task4: Map<Integer, Map<String, Long>> result = data.stream().collect(Collectors. *groupingBy*(MonitoredData::getDay, Collectors.*groupingBy*(MonitoredData::getActivity, Collectors.*counting*()))): mapeaza pentru fiecare zi aflata in fisier o structura de tipul map, care reprezinta activitatea si un numar care reprezinta de cate ori acea activitate s-a efectuat in acea zi;

Concluzii:

-aplicatia e capabila sa obtina informatii despre activitatile unei persoane pe durata mai multor zile si filtrarea acestor dupa anumite criterii;

Ce s-a invatat:

-aprofundarea cunostintelor legate de Lambda expressions si Streams;

Dezvoltari ulterioare:

-imbunatatirea metodelor;

Bibliografie:

https://stackoverflow.com/

https://www.geeksforgeeks.org/

http://www.mkyong.com/tutorials/java-8-tutorials/