# Expresii Lambda

# Tehnici de programare

# Tema 5

Alexandru Nistor Botolan

An 2 Seria B Grupa 30227

Facultatea de Automatica si Calculatoare

Sectia Calculatoare si Tehnologia Informatiei

## 1.Obiectivul temei

Obiectivul principal al temei este de a crea o aplicatie care se ocupa cu procesarea unui input de informatie cu ajutorul expresiilor lambda si de a determina anumite cerinte referitoarea la inputul respective.

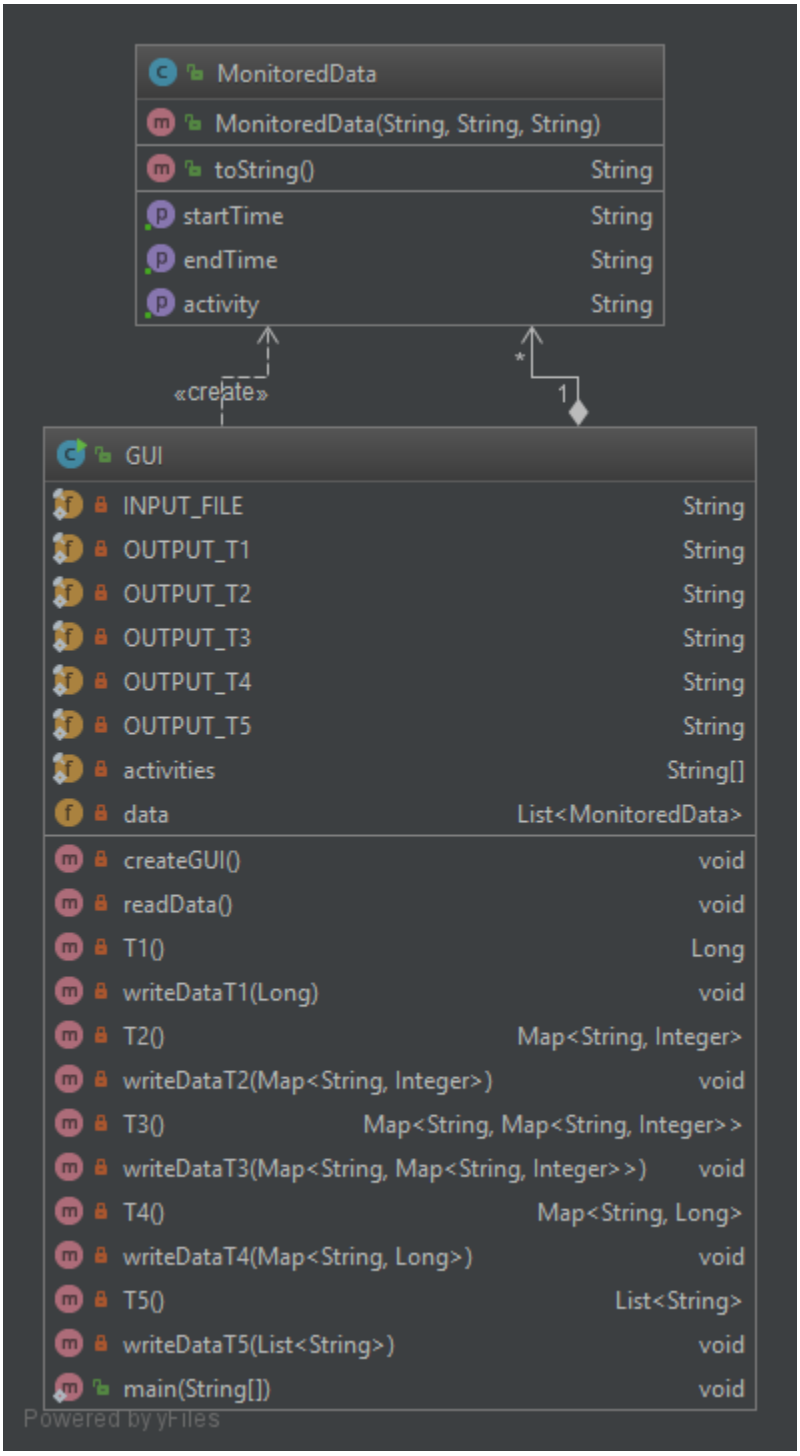
Obiectivul secundar este de a utiliza doar streamuri si expresii lambda pentru a procesa acest input.

## 2.Analiza problemei, asumptii, modelare, scenario, cazuri de utilizare, erori

Aceasta aplicatie trebuie doar sa utilizeze streamuri si expresii lambda, astfel programul va fii compact, utilizand doar o clasa care va memora datele citite din fisier si o alta clasa care va prezenta o interfata minimalisa cu ajutorul careia vom realiza cerintele respective.

Problema propusa este alcatuita din 5 cerinte: numararea zilelor diferite din lista, numararea aparitiei unei activitati in fisier, numarar5ea de cateori apare o activitate intr-o zi, sa se gaseasca cate activitati au o durata mai mare de 10 ore in total, si sa se afiseze activitatiile care apar intr-o proportie de 90 % cu o durata mai mica de 5 minute.

3. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator, modul de tratare a erorilor)



In problema propusa este in paritita in 2 clase: clasa MonitoredData se ocupa cu stocarea datelor citite din fisier, iar a doua clasa se ocupa cu prezentarea interfetei minimaliste, dar si executarea operatiunii dorite, cum ar fi cele prezentate mai sus, dar si scrierea si citirea din fisier.

## 4.Implementare

Implementarea problemei consta in clasa GUI, in aceasta clasa, la inceputul programului se executa metoda readData() care citeste datele din fisier si cu ajutorul expresiilor lambda le proceseaza si le salveaza sub forma unei liste in memorie de elemente de tip MonitorData. Dupa aceasta, se poate selecta optiunea care sa fie executata. Pentru task-ul 1, care presupune numararea zilelor pe durata simularii descrisa in fisier, am folosit expresii lambda care transforma lista intr-un stream, iar apoi, cu ajutorul unor instructiuni cum ar fi map, luam de la fiecare element startTime, iar cu ajutorul metodei distinct() ne asiguram ca luam aceiasi data o singura data, iar la final, cu metoda count() numaram cate astfel de elemente avem si returnam valoarea rezultata. Pentru taskul 2, ne cere sa afisam fiecare activitate de cate ori a fost gasita in fisierul rezultat. Pentru inceput am transformat lista de activiati intr-un stream, iar apoi cu ajutorul metodei toMap() transformam streamul intr-un Map<String, Integer>. Cheia map-ului va fi numele activiatii, iar valoarea va fi de cate ori aceasta a aparut. Pentru a nu avea duplicate, in cazul in care cheia a mai aparut o data, ne vom utiliza de expresia (a, b) -> a + 1. Aceasta expresie, in cazul in care a si b au aceiasi valoare, inseamna ca avem aceiasi activiate, deci valoarea ei va fi incrementata cu 1. Pentru task-ul numarul 3, ne cere ca pentru fiecare zi sa se creeze o structura de tip Map<String, Map<String, Integer>>. Pentru inceput se transforma lista intr-un stream, iar apoi mapam map-ul mare dupa valoarea zilei. Valoarea map-ului va fi reprezentat de un alt map care va avea ca si cheie ziua respectiva, iar ca valoare un intreg care va reprezenta de cate ori a aparut in ziua respectiva. Pentru aceasta vom filtra a doua mapare dupa conditia ca ziua din map-ul al doilea sa fie aceiasi cu cea din primul map, astfel instructiunea care ne ofera aceasta solutie este filter, iar in interiorul acesteia testam egalitatea dintre cele doua sirur de caractere. Dupa ce aceasta conditie a fost satisfacuta, ne vom utiliza de toMap sa se creeze a doua mapare, cheia fiind activitatea, iar valoarea de cate ori a aparut activitatea in ziua respectiva. Ca sa

eliminam duplicatele, vom utiliza, ca mai sus (a, b) -> a + 1. Pentru cea dea doua problema, cand avem duplicarea timpului de start, in cazul in care apar doua zile identice, vom folosi doar una dintre ele. La finalul executiei acestei metode, vom returna un map care are structura Map<String, Map<String, Integer>>.

Pentru task-ul 4 avem nevoie sa returnam care activitati au avut o durata totala mai mare de 10 ore. Pentru aceasta ne vom folosi de tipul Date din java, si vom utiliza formatul " yyyy-MM-dd HH:mm:ss" cu ajutorul caruia vom parsa data pe care o avem stocata intr-un tip Date. Pentru inceput vom grupa lista transformata in stream dupa numele activiatii. Aceasta o realizam cu ajutorul metodei groupingBy(). Apoi, pentru a determina valoarea tuplei din tabela, vom utiliza metoda summingLong(), cu aceasta vom face suma duratei unei activitati. Utilizand metoda parse si getTime() transformam sirul de caractere care reprezinta data si ora intr-o valoare numarica de tip-ul long, iar diferenta timpului de start si timp-ului de finish ne va da valoarea duratei acelei activitati. Dupa ce avem acest map, il vom procesa, transformand-ul intr-un stream si il vom filtra dupa conditia ca valoarea de tipul logn sa fie mai mare decat 10 * 3600 * 1000, valoarea aceasta reprezinta 10 ore transformate in milisecunde. La final, vom returna un map care va avea ca si cheie numele activitatii, iar ca valoare care ore a durat aceasta, valoare tranformata din milisecunde in ore, cu ajutorul expresii getValue() / 3600 / 1000.

Pentru task-ul 5 avem nevoie sa returnam o lista care a avut in proportie de 90 % din numarul de aparitii in fisier, o durata mai mica decat 5 minute. Pentru aceasta ne vom folosi de doua map-uri. Primul map va retine activitatiile si cate dintr-e aceste aparitii ale activiatii au avut o durata mai mica de 5 minute. Pentru aceasta vom utiliza metoda filter si vom pune conditia ca durata evenimentului sa fie mai mic decat 5 * 60 *1000. Dupa incheierea acestei instructiuni, ne vom utiliza de metoda de la task-ul 2 si vom lua de cate ori a aparut o activiate pe durata simularii. Astfel, vom procesa primul map si vom filtra dupa existenta activiatii si vom impune conditia suplimentara care va consta in valoarea din primul map sa fie mai mare sau egala decat 0.9 * valoarea din map-ul de la task-ul 2. In cazul in care aceasta conditie este satisfacuta, vom fi siguri ca in 90 % din cazuri activitatea a avut o durata mai mica de 5 minute. Valoarea rezultata va fi o lista care contine numele tuturor activiatiilor care au indeplinit conditia.

## 5.Concluzii

Aceasta aplicatie este buna pentru procesarea unui volum mare de date, si de analiza lui si de a lua decizii cu privire la aceste evenimente. Din aceasta tema am invatat sa utilizez expresiile lambda, dar si sa utilizez streamurile pentru procesarea datelor dar si citirea si scrierea lor in fisier cu ajutorul acestora.