*Universitatea Tehnică Cluj-Napoca*
*Facultatea de Automatică și Calculatoare*

# FUNDAMENTAL PROGRAMMING TECHNIQUES

*Tania Loreana Gabor*
*Grupa:30224*

# 1. Obiectivul principal

O casă inteligentă are un set de senzori care pot fi folosiți pentru a înregistra comportamentul unei persoane care locuiește în aceasta. Jurnalul istoric al activității persoanei este stocat sub formă de tuple(startTime, endTime, activity), unde startTime și endTime reprezintă data și ora la care fiecare activitate a început și s-a încheiat în timp ce activitatea reprezintă tipul de activitate desfășurată de persoana respectivă: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Dinner, Snack, Spare_Time/TV, Grooming. Fișierul jurnal atașat Activities.txt conține un set de înregistrări de activități într-o anumită perioadă de timp. A trebuit să definim o clasă MonitoredData având startTime, endTime și activity ca variabile de instanță și să citim datele fișierului de intrare în structura de date MnitoredData de tip List. Folosind tehnici de procesare stream și expresii lambda introduse de Java 8,a trebuit să scriem următorul set de programe scurte pentru procesarea obiectului MonitoredData:

1. Citiți datelor corect din fișierul de input.
2. Numărați zilele distincte care apar în datele de monitorizare.
3. Determinați o structură map ,care mapează pentru fiecare tip de acțiune distinctă numărul de apariții în jurnal
4. Generați o structură de date de tip <Map> care conține numărul de activități pentru fiecare zi a jurnalului (sarcina numărul 2 aplicată pentru fiecare zi a jurnalului)
5. Determinați o structură de date a formularului Map care mapează pentru fiecare activitate durata totală calculată în perioada de monitorizare.
6. Filtrați activitățile care au 90% din eșantioanele de monitorizare cu o durată mai mică de 5 minute, colectați rezultatele într-o listă care conține doar numele activităților distincte.

# 2. Analiza problemei

Datele mari se definesc ca fiind niște seturi de date gigantice atât de complexe încât aplicațiile software de procesare a datelor clasice nu pot gestiona aceste seturi într-un timp convenabil și nu sunt capabile să le facă față. Procesele care sunt aplicate pe aceste seturi sunt: colectarea, stocarea sau depunerea, analizarea și procesarea, curațarea datelor, căutarea și analizarea seturilor, partajarea lor, transferul, vizualizarea, interogarea și actualizarea și nu în ultimul rând măsuri de securitate și măsuri de confidențialitate a datelor . Pentru efectuarea acestor acțiuni pe structuri de date mari s-au dezvoltat diferiți algoritmi.Pentru analiza datelor există testarea A / B ,pentru procesarea limbilor naturale sau învățarea automată. Stocarea în cloud și și lucrul cu baze de date sunt soluții mari de stocare a datelor dezvoltate și încă în curs de dezvoltare. Un exemplu de acest tip de date mari ar putea fi faptul că în această lume există până la 5 miliarde de abonamente la telefoane mobile. Gestionarea istoricului apelurilor, a mesajelor și a tuturor activităților acestor abonamente la

telefoane mobile necesită algoritmi avansați. De asemenea, până la 2 miliarde de persoane accesează zilnic internetul. Înregistrarea și verificarea acestor activități necesită, de asemenea, calcule la nivel înalt și gestionarea aplicațiilor. Internetul aplicațiilor I O T este rețeaua de dispozitive fizice, clădiri, vehicule și orice alt tip de dispozitive. Să luăm în considerare un exemplu, pentru a înțelege mai bine cazurile de utilizare ale acestei aplicații. Să zicem că am cumpărat doar un nou sistem integrat de case inteligente cu o colecție foarte mare de senzori, capabil să înregistreze în timp real ce activitate prelucrați la un anumit moment și să salvați toate aceste activități într-un jurnal, cu ora de început, data de începere, denumirea activității ora și data de încheiere. Am cumpărat acest sistem pentru a îmbunătăți gestionarea timpului pe care îl utilizăm și pentru a optimiza modul în care organizăm ziua.

De aceea, am avea nevoie de o aplicație software care să gestioneze și să trateze datele primite de la sistemul hardware și să calculăm un set de analize asupra datelor, oferind ca rezultat un set de statistici precum numărul de zile în înregistrări, pe activități cu durata de un anumit timp sau numărul de apariții ale unei anumite activități într-o zi sau un alt tip de date care ne-ar putea interesa. Acest sistem de rețea permite acestor dispozitive să comunice între ele și să facă schimb de date, pentru a îmbunătăți activitatea acestor dispozitive. Problema pe care ne focusăm este gestionarea datelor primite de la un set de senzori într-un sistem de automatizare a casei IOT, care înregistrează activitățile zilnice ale proprietarului. Am putea folosi colecții din Java, dar începând cu Java 8 există o API actualizată care folosește streamuri.Streamutile funcționează pe seturi de date utilizând procesarea cu mai multe nuclee, chiar dacă programatorul nu trebuie să utilizeze o singură linie de cod multiprocesat sau multi-thread. Această caracteristică aduce un întreg set de acțiuni care pot fi efectuate pe orice colecție, cum ar fi filtre, instrucțiuni de mapare, sortare, însumare, colectare instrucțiuni pentru a aduce elemente filtrate din stream într-o colecție.Pentru realizarea temei, s-a precizat că trebuie implementată cu ajutorul streamurilor si expresiilor lambda.

Introdus în Java 8, API-ul Stream este utilizat pentru procesarea colecțiilor de obiecte. Un stream este o secvență de obiecte care acceptă diferite metode care pot fi utilizate pentru a produce rezultatul dorit.

Caracteristicile streamului Java sunt :

Un stream nu este o structură de date, ci are intrare din canalele Collections, Arrays sau I / O.

Streamurile nu schimbă structura de date originală, acestea furnizează doar rezultatul în conformitate cu metodele canalizate.

Fiecare operație intermediară este executată încet și returnează un stream ,prin urmare, diverse operații intermediare pot fi utilizate. Operațiunile terminalului marchează sfârșitu lstreamului și returnează rezultatul.

# 3. Modelarea datelor

Procesul de modelare este definit ca fiind procesul de modulare a unei mari probleme în unele mai mici, care sunt mai ușor de înțeles și depanat. Acest lucru contribuie, de asemenea, la clarificarea unei idei abstracte. În dezvoltarea software, modelarea este esențială pentru a construi o aplicație care are o structură puternică de bază. După cum s-a menționat mai sus, trebuie să prelucrăm intrările unui fișier care reprezintă un jurnal și să citim datelor corect din fișierul de input,număram zilele

distincte care apar în datele de monitorizare ,determinăm o structură map ,care mapează pentru fiecare tip de acțiune distinctă numărul de apariții în jurnal , generăm o structură de date de tip <Map> care conține numărul de activități pentru fiecare zi a jurnalului (sarcina numărul 2 aplicată pentru fiecare zi a jurnalului) ,determinăm o structură de date a formularului care mapează pentru fiecare activitate durata totală calculată în perioada de monitorizare,filtrăm activitățile care au 90% din eșantioanele de monitorizare cu o durată mai mică de 5 minute și să colectăm rezultatele într-o listă care conține doar numele activităților distincte.

Arhitectura aplicației este într-adevăr simplă, deoarece conține o clasă de înregistrări de date monitorizate , o unitate principală Main în care se scrie în fișierele de iesire corespunzătoare și o clasă Task în care sunt implementate cele 6 metode.

# 4. Cazurile de utilizare

Să luăm în considerare un exemplu, pentru a înțelege mai bine cazurile de utilizare ale acestei aplicații. Să zicem că am cumpărat doar un nou sistem integrat de case inteligente cu o colecție foarte mare de senzori, capabil să înregistreze în timp real ce activitate se execută la un anumit moment și să salvați toate aceste activități într-un jurnal, cu ora de început, data de începere, denumirea activității, ora și data de încheiere. Am cumpărat acest sistem pentru a îmbunătăți gestionarea timpului pe care îl utilizăm și pentru a optimiza modul în care ne organizăm ziua. De aceea, am avea nevoie de o aplicație software care să gestioneze și să trateze datele primite de la sistemul hardware și să facem un set de analize asupra datelor, oferind ca rezultat un set de statistici precum numărul de zile din înregistrări, pe activități cu durata într-un anumit timp sau numărul de apariții ale unei anumite activități într-o zi sau un alt tip de date care ne-ar putea interesa.

# 5. Implementare

Voi descrie în continuare cele mai esențiale metode prezente în cerere împreună cu utilizarea și semnificația lor.

## MonitoredData.java

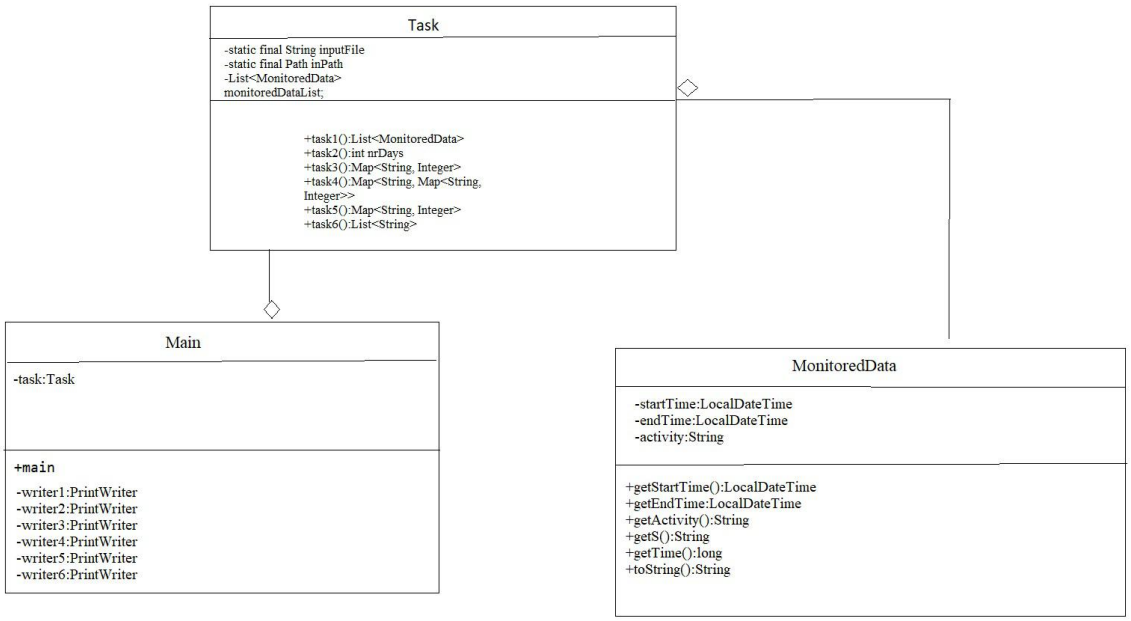
O instanță a acestei clase reprezintă o linie de date monitorizate din fișierul Activities.txt. Fișierele sunt descrise în cerința problemei: startTime este o variabilă LocalDateTime care indică ora și data la care a început activitatea, endTime este o variabilă LocalDateTime care indică ora și data la care activitatea s-a încheiat și activitatea este un String care păstrează numele a activității care a fost efectuată între cele două date anterioare înregistrate.În această clasă am getteri și setteri , dar și metodele:getTime() prin care returnează diferența în minute dintre două date, metoda getS() prin care convertește din LocalDateTime în String o anumită dată, dar și metoda toString.

## Task.java

În această clasă am implementat metodele prin care am realizat cele șase taskuri care trebuiau implementate.Pentru realizarea acestora am folosit expresii lambda .

## Main.java

În această clasă am creat fișiere de tipul txt cu numele task_nrtask.txt , am creat obiectul de tip Task si am apelat metodele din această clasă.




Cele mai importante aspecte privind modul în care a fost folosit Java 8 este clasa Collectors care implementează fișiere utilizate frecvent pentru clasa Stream. Collect este o operație extrem de utilă a terminalului pentru a transforma elementele fluxului într-un fel diferit de rezultate, de ex. o listă Set sau Map. Collect acceptă un colecționar care constă din patru operațiuni diferite: un furnizor, un acumulator, un combinator și un fișier. Aceasta sună la început super complicat, dar partea bună este că Java 8 acceptă diverși colectori integrați prin clasa Collectors. Deci, pentru cele mai comune operații, nu trebuie să implementați singur un colector.

# 6.Rezultatul

Rezultatul dezvoltării acestei aplicații, , este un software ușor de utilizat, ceea ce facilitează utilizatorului să gestioneze câteva activități ale unui jurnal primit de la un sistem IOT de automatizare casnică și generarea de statistici pe câteva subiecte: numărarea zilelor distincte care apar în datele de monitorizare, determină o structură de tip map , care mapează la fiecare tip de acțiune distinct numărul de apariții în jurnal, scrie structura rezultată într-un fișier text, generează o structură de date de tip <Map> care conține numărul de activități pentru fiecare zi a jurnalului (sarcina numărul 2 aplicată pentru fiecare zi a jurnalului) și scrie rezultatul într-un fișier text. Determină o structură de date Map care mapează pentru fiecare activitate durata totală calculată în perioada de monitorizare. Filtrează activitățile și scrie rezultatul într-un fișier text. Filtrează activitățile care au 90% din eșantioanele de monitorizare cu o durată mai mică de 5 minute, colectează rezultatele într-o listă care conține doar numele de activitate distincte și scrieți rezultatul într-un fișier text.Astfel,aplicația finală este capabilă să rezolve toate cerințele pentru această atribuire folosind procesarea stream Java 8 și expresiile lambda. Folosirea programării funcționale și declarative pentru această sarcină specifică a fost o muncă extrem de solicitantă care mi-a crescut abilitățile de lucru în Java.

# 6. Concluzii

Am învățat lucrând la acest proiect:
O mai bună utilizare a Intellij
- O mai bună utilizare a limbajului de codare Java
- O mai bună înțelegere și utilizare a paradigmelor de programare orientate pe obiect
- Cod mai organizat și clar
- Programarea pentru prima dată, așa cum nu am mai făcut asta niciodată în viața mea.
- Am aflat despre streamuri
- Aflați despre Java 8 Streams și cum pot fi utilizate în gestionarea seturilor mari de date.
- O introducere dar și o mai bună înțelegere a expresiilor lambda.

## 8.Posibile îmbunătățiti

- Ar putea fi generate mai multe date statistice pe fișierele jurnal primite de la sistemele hardware.
- Cu un sistem de automatizare a casei mai complex, aplicația ar putea mări gama de subiecte pentru a studia și analiza activitatea utilizatorului
- Aplicația ar putea oferi utilizatorului sfaturi despre cum să îmbunătățească diferite elemente în comportamentul zilnic, cum ar fi să mănânce mai bine, să facă mai mult sport.

-Aplicației i s-ar putea adăuga o interfață grafică.
-Aplicația ar putea monitoriza activitatea mai multor persoane.

## 9. Bibliografie

https://www.baeldung.com/java-blogs
https://beginnersbook.com/java-io-tutorial-with-examples/
Lucrarea de laborator de pe siteul http://www.coned.utcluj.ro/~salomie/
https://www.geeksforgeeks.org/java-8-collectors-counting-with-examples/