DOCUMENTATIE TEMA 5

PROCESSING SENSOR DATA OF DAILY LIVING ACTIVITIES

Baleanu Sorina-Diana

Grupa 302210

Profesor laborator: Stoica Anda

Cuprins:

1.Obiectivul temei

2.Analiza problemei

3.Proiectare

4.Implementare

5.Rezultate

6.Concluzii

7.Bibliografie

1.Obiective

1.1.Obiectivul principal

Obiectivul principal al acestui proiect este proiectarea, implementarea și testarea unei aplicatii pentru analiza comportamentului unei persoane. Persoana este inregistrata de un set de senzori instalat în casa sa. Jurnalul istoric al activitatiilor efectuate de este stocat sub formă de tuple care contin timpul de incepere si terminare al fiecarei activitati, alaturi de eticheta activitatii care reprezintă tipul de activitate desfasurat de persoană: Plecarea, Toaletarea, Dusul, Dormirea, Micul dejun,Pranz, Cina, gustare, Timp liber / TV, Toaleta.

1.2.Obiectivele secundare

Obiectivele secundare constau in pasii urmati pentru atingerea obiectvelor principale.

|  |  |  |
| --- | --- | --- |
| Obiectiv secundar | Descriere | Capitol |
| Dezvoltarea de scenarii si cazuri de utilizare | In computere, un scenariu este o naratiune a interactiunilor dintre roluri ( actori in UML ) si sistem. In cazul nostru rolul - un utilizator al aplicatiei.  În inginerie software și sisteme, un caz de utilizare este o listă de acțiuni sau pași de eveniment care definesc de obicei interacțiunile dintre un rol (cunoscut în Unified Modeling Language (UML) ca actor) și un sistem pentru a atinge un obiectiv. Actorul poate fi un sistem uman sau alt sistem extern. | 2 |
| Structuri de date | Alegerea structurilor de date care au fost folosite pentru indeplinirea obiectivului principal | 2 |
| Proiectare | Proiecatrea este facuta folosind programarea functionala in Java cu expresii Lamda si procesare cu Streams. | 3 |
| Algoritmi | Algoritimi folositi pentru implementarea celor 6 task-uri. | 3 |
| Implementarea solutiei | Se va descrie fiecare clasa alaturi de toate metodele implementate in aceasta | 4 |
| Testare | Testarea corectitudinii operatiilor efectuate. | 5 |

2.Analiza problemei ,modelare, scenarii si cazuri de utilizare

2.1.Analiza problemei

In urma primirii datelor sub forma unui fisier text, programul va analiza comportamentul persoanei . Task-urile programului sunt urmatoarele:

* Prelucrarea datelor din fiser cu ajutorul streamu-urilor, creerea unei liste de obiecte de tip MonitoredData si afisarea datelor in fiserul task1.txt.
* Numararea zilelor distincte care apar in datele trimise inspre prelucrare. Afisarea rezultatului in fiserul task2.txt.
* Numararea aparitei fiecarei activitatii in datele transmise inspre prelucrare. Afisarea rezultatului in fisierul task3.txt
* Numararea aparitiei fiecarei activitati in fiecare zi in datele transmise inspre prelucrare. Afisarea rezultatului in fisierul task4.txt.
* Contorizarea duratei fiecarei activitati aparute in datele transmise inspre prelucrare. Afisarea rezultatului in fisierul task5.txt.
* Filtrearea activitatilor care au peste 90% din inregistrarile de monitorizare cu durata mai putin de 5 minute. Afisarea rezultatului in fisierul task6.txt

2.2 .Scenarii si cazuri de utilizare

Use-Case: Rularea fisierului .jar

Primary Actor: user

Cazuri de succes:

Utilizarea aplicatiei presupune:

Pasul 1 : Creerea unui fisier text Activities.txt, pe baza caruia programul va realiza task-urile necesare. In fisierul text trebuie introduse datele dupa urmatorul format:

* Pe fiecare linie vom avea timpul de incepere al unei actiuni, timpul de finalizare al actiunii si label-ul actiunii, despartite fiecare prin doua taburi
* Timpul de incepere si finalizare trebuie sa respecte formatul: yyyy-mm-dd hh:mm:ss

Pasul 2 : Se ruleaza aplicatia .jar din terminal.

Pasul 3: Aplicatia va realiza cele 6 task-uri si va afisa rezultatele in fisierele task1.txt, task2.txt, task3.txt, task4.txt task5.txt si task6.txt din folderul curent. Se vor afisa pe ecran mesajul „ Task\_no done!” in caz de reusita.

Scenarii de esec:

* Timpul de inceput si timpul de final nu sunt introduse conform formatului precizat.
* Nu este respectata spatierea de doua taburi intre datele introduse in fisier.

3.Proiectare

3.1.Decizii de proiectare

Una din principalele decizii de proiectare este utilizarea Java Streams si Java Lambda Expression disponibile incepand cu Java 8.

Expresiile lambda Java sunt noi în Java 8. Expresiile Java lambda sunt primul pas în programarea functionala Java. O expresie lambda Java este astfel o funcție care poate fi creata fara a aparține vreunei clase. O expresie lambda Java poate fi transmisă ca si cum ar fi un obiect si executata la cerere. Expresiile lambda Java sunt utilizate în mod obișnuit pentru a implementa ascultatori / apeluri de evenimente simple sau în programarea funcțională cu API Streams Java. API-ul Stream este utilizat pentru procesarea colectiilor de obiecte. Un flux este o secventa de obiecte care accepta diferite metode care pot fi canalizate pentru a produce rezultatul dorit. Caracteristicile principale ale Java stream sunt :

* Un Stream nu este o structura de date, ci are intrare din canalele Collections, Arrays sau I / O.
* Stream-urile nu schimba structura de date originală, acestea furnizează doar rezultatul în conformitate cu metodele canalizate.

Aceste caracteristici principale ale Stream-urilor au fost foarte utile in rezolvarea temeie doarece am putut obtine rezultatele dorite fara a modifica structura de date obtinuta prin procesarea fisierului text si fara a utiliza prea multe structuri de date intermediare.

Fiecare operație intermediară este executata si returnează un stream , prin urmare, diverse operatii intermediare pot fi canalizate.

3.2.Diagrame UML

Unified Modeling Language (prescurtat UML) este un limbaj standard pentru descrierea de modele si specificatii software. Diagrama de clase UML Este folosită pentru reprezentarea vizuală a claselor și a interdependențelor, taxionomiei și a relațiilor de multiplicitate dintre ele. Diagramele de clasă sunt folosite și pentru reprezentarea concretă a unor instanțe de clasă, așadar obiecte și a legăturilor concrete dintre acestea.

A screenshot of a cell phone

Description automatically generated

3.3.Structuri de date

Structura principala folosita in acest proiect este Map pentru stocarea rezultatelor asociate task-ului 3, task-ului 4 si task-ului 5. Un obiect de tip Map este un obiect care mapeaza cheile valorilor. Un obiect de tip Map nu poate conține chei duplicate: Fiecare cheie poate face o mapare la cel mult o valoare. Modeleaza abstractizarea functiei matematice. Interfata Map include metode pentru operațiunile de baza basic (cum ar fi put, get, remove, containsKey, containsValue, size, si empty), operatii de tip bulk(cum ar fi putAll si clear), si operatii pe colectie(cum ar fi keySet, entrySet, si values).

Alta structura folosita a fost LocalDateTime, in aceasta structura am convertit String-urile corespunzatoare timpului de incepere si finalizare a actiunii pentru a putea calcula durata fiecarei actiuni. LocalDateTime este un obiect de dată imuabil care reprezintă o data, de multe ori vizualizata ca an - luna - zi- ora -minut - secund. Timpul este reprezentat de precizia nanosecundelor.

Durata unei actiuni a fost calculata cu ajutorul clasei Duration. Aceasta clasa modeleaza o cantitate de timp în termeni de secunde si nanosecunde. Poate fi accesat folosind alte unități bazate pe durată, cum ar fi minute și ore. În plus, unitatea DAYS poate fi utilizată și este tratată ca fiind exact egală cu 24 de ore, ignorând astfel efectele de vară. Pentru practic, durata este stocată cu constrângeri similare cu Instant. Durata folosește rezoluția nanosecundelor cu o valoare maximă a secundelor care poate fi menținută într-un timp lung. Aceasta este mai mare decât vârsta actuală estimată a universului. Intervalul unei durate necesită stocarea unui număr mai mare decât cel lung. Pentru a obține acest lucru, clasa stochează o secundă de reprezentare lungă și o int care reprezintă nanosecunda de secundă, care va fi întotdeauna între 0 și 999,999.999. Modelul are o durată direcționată, ceea ce înseamnă că durata poate fi negativă. Durata este măsurată în „secunde”, dar acestea nu sunt neapărat identice cu definiția științifică „SI secundă” bazată pe ceasuri atomice.

Clasa Calendar este o clasă abstracta care ofera metode pentru conversia intre un moment specific in timp si un set de campuri calendaristice precum YEAR, MONTH, DAY\_OF\_MONTH, HOUR si asa mai departe, și pentru manipularea campurilor din calendar, cum ar fi obținerea datei din săptămâna viitoare. Am folosit un obiect de tip calendar pentru a obtine din data de inceput a actiunii doar ziua lunii.

3.4.Proiectare clase si relatii intre clase

Clasa MonitoredData

Clasa MonitoredData reprezinta datele furnizate de catre senzori in legatura cu o actiune executata de persoana. Un obiect de acest tip este reprezentat de un String startTime si un String endTime care corespund timpului de incepere si finalizare al actiunii respective.

Clasa Task

Clasa Task este reprezentata de o lista de activitati care sunt obiecte de tip MonitoredData. Aceasta lista este initalizata in constructorul clasei prin procesarea activitatilor din fisierul de intrare Activities.txt. Fiecare task corespunde unei metode impelementata in clasa Task, fiecare metoda este responsabila de scrierea rezultatului in fisierul corespunzator Task-ului.

3.5.Algoritmi

Task1

Pentru indeplinirea primului task am atribuit unui stream de String-uri, liniile fisierului de intrare, urmand sa impartim continutul linei in mai multe string-uri la intalnirea a doua tab-uri , cu ajutorul acestor string-uri am creeat un nou stream prin apelarea metodei map cu functia (o->new MonitoredData(o[0],o[1],o[2]) dupa care l-am transformat intr-o lista de obiecte de tip MonitoredData care va fi retinut in clasa Task si cu care vor lucra toate celelalte Task-uri. Aceste operatii au fost realizate in constructorul clasei Task. Pentru finalizarea primului task este necesara si apelarea metodei task 1 care se ocupa cu scrierea in fisierul task1.txt a informatiilor corespunzatoare obtinute.

Task2

Pentru indeplinirea celui de-al doilea task am folosit lista de obiecte din clasa Task. Am folosit aceasta colectie drept stream si am creeat un nou stream de activitati prin apelarea metodei map cu functia (d->d.getActivity) dupa care am apelat metoda distinct pentru a pastra fiecare actvitate o singura data dupa care am apelat metoda count pentru a numara elemenetele din stream si a obtine rezultatul dorit .Dupa obtinerea rezultatului am efectuat scrierea acestuia in fisierul task2.txt.

Task3

Pentru indeplinirea celui de-al treilea task am folosit lista de obiecte din clasa Task. Am folosit aceasta colectie drept stream si am colectat informatiile dorite apeland metoda de collect. Datele au fost grupate dupa activitati si la fiecare intalnirea a unei activitati s-a calculat numarul aparitei acesteia in datele furnizate de fiserul de intrare Actvities.txt. Am obtinut ca rezultat o structura de tip Map < String , Integer>, cheile structurii reprezentatnd numele activitatii , iar valoarea din map corespunzatoare fiecarei chei este numarul de aparitii al activitatii respective. Dupa obtinerea rezultatului se face scrierea acestuia in fisierul task3.txt si se afiseaza u mesaj de sucees.

Task4

Pentru indeplinirea celui de-al patrulea task am folosit lista de obiecte din clasa Task. Am folosit aceasta colectie drept stream si am colectat informatiile dorite apeland metoda de collect. Datele au fost grupate dupa ziua lunii in care au fost realizate , zi obtinuta prin apelarea metodei getDistinctDay din clasa MonitoreData si la fiecare intalnirea unei zile noi s-a realizat si o grupare dupa activitai si numarul aparitiei acestora. S-a obtinut ca rezultat o structura de tipul Map < Integer , Map <<String , Integer>>, cheile structurii reprezentatndzilele in care au inceput activitatile , iar valoarea din map corespunzatoare fiecarei chei este un Map cu anumele actvitaii drept cheie si numarul de aparitii al activitatii respective drept valoare . Dupa obtinerea rezultatului se face scrierea acestuia in fisierul task4.txt si se afiseaza u mesaj de sucees.

Task5

Pentru indeplinirea celui de-al ciniclea task am folosit lista de obiecte din clasa Task. Am folosit aceasta colectie drept stream si am colectat informatiile dorite apeland metoda de collect. Datele au fost grupate dupa numele activtatii, si la fiecare intalnirea fiecarei activitati s-a calculat durata sa si a fost adunata cu durata celoalte activitati de acelasi fel inatalnite apelanduse metoda plus a clasei Duration. S-a obtinut o structura de tipul Map<String ,Duration> unde cheia este reprezentata de numele activtatii si valoarea reprezinta durata totala a acestei activitati pe intreaga perioada monitorizata de senzori. Dupa obtinerea rezultatului se face scrierea acestuia in fisierul task5.txt si se afiseaza u mesaj de sucees.

Task6

Pentru indeplinirea celui de-al sasela task am folosit lista de obiecte din clasa Task. Am folosit aceasta colectie drept stream si am colectat informatiile dorite apeland metoda de collect. Datele au fost grupate dupanumele activtatii, si pentru fiecare actvitate s-a caluclat media duratei acesteia . Am obtinut o structura de tip Map< String , Double> unde cheia reprezinta numele activitatii si valoarea este media in minute a acesteia , am folosit aceasta structura pe post de stream si am utilizat metoda filter pentru a pastra doar intrarile din Map care au media mai mica de 5.55(adica 90% din aparitii au durata mai mica decat 5 minute, rezultatul a fost obtinut utilizand regula de 3 simpla). Dupa filtare am creat un nou stream cu actvitatile care indeplinesc conditiile, iar apoi l-am transformat intr-o lista. Dupa obtinerea rezultatului se face scrierea acestuia in fisierul task6.txt si se afiseaza u mesaj de sucees.

5.Implementare

Clasa MonitoredData

* Getteri si setteri corespunzatori
* Metoda getDistinctDay: converteste Stringul startTime intr-o data pe care o seteaza intr-un obicect de tip Calendar, urmand sa returneze ziua lunii drept intreg cu ajutorul obiectului de tip Calendar
* Metoda getDuration: converteste String-urile startTime si endTime in obiecte de tip LocalDateTime conform formatului de data „yyyy-mm-dd hh:mm:ss” dupa care calcuelaza durata folosind metoda between implementata in clasa Duration.

Clasa Task

* Metoda task1
* Metoda task2
* Metoda task3
* Metoda task4
* Metoda task5
* Metoda task6

6.Rezultate

Rezultatele obtinute pot fi consultate in fiserele task1.txt, task2.txt, task3.txt, task4.txt, task5.txt, task6.txt atasate proiectului.

7.Concluzii

In concluzie realizarea acestui proiect a fost foarte utila , am reusit pentru prima data sa introducem programarea functionala in Java. Am lucrat pentru prima data cu Java Stream si Java Lamda Expressions ce au fost introduse incepand cu Java 8. Am descoperit astfel metode mult mai usoare de a obtine rezultatele dorite fara a folosi foarte multe structuri intermediare apeland pur si simplu la programarea functioanala.

Un alt plus a fost lucrarea cu timpul in Java , am invatat sa manipulez mult mai bine datele , duratele si timpul ,utilizand instante ale claselor LocalDateTime , Calendar, Duration.

8.Bibliografie

* <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>
* <https://docs.oracle.com/javase/tutorial/java/javaOO/methodreferences.html>
* <https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html>
* <https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>