**TEHNICI DE PROGRAMARE FUNDAMENTALE**

**ASSIGNMENT 5**

**PROCESSING SENSOR DATA OF DAILY LIVING ACTIVITIES**

**DOCUMENTATIE**

Bakk Cosmin-Robert

Grupa 30227

Cuprins

1. Obiectivul temei

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

3. Proiectare

3.1. Diagrama UML

3.2. Proiectare clase

4. Implementare si testare

4.1. Metode

4.2. Testare

5. Concluzii

6. Bibliografie

1. Obiectivul temei

Obiectivul acestei teme de laborator este de a proiecta, implementa si testa o aplicatie pentru a analiza comportamentul unei persoane inregistrate de un set de senzori instalati in casa acesteia. Datele despre activitatea persoanei pe un interval de timp vor fi salvate intr-un fisier .txt, Activities.txt. Pe fiecare rand din fisierul de intrare vor fi scrise numele unei activitati facute de persoana, impreuna cu timpii de inceput si de sfarsit a activitatii, in formatul “yyyy-MM-dd HH:mm:ss yyyy-MM-dd HH:mm:ss Activity name ”. Numele activitatilor este unul dintre urmatoarele: Leaving, Toileting, Showering, Sleeping, Breakfast, Lunch, Snack, Spare\_Time/TV, Grooming. Pe baza datelor extrase din fisierul de intrare, se vor rezolva urmatoarele task-uri, iar rezultatele vor fi puse in cate un fisier .txt, specific fiecarui task:

Task 1: Scrierea tuturor datelor citite din fisier.

Task 2: Numarul de zile distincte in care s-a realizat inregistrarea activitatii.

Task 3: Numarul de aparitii a fiecarei activitati pe intreaga perioada a inregistrarii activitatii.

Task 4: Numarul de aparitii a fiecarei activitati pentru fiecare zi din perioada inregistrarii activitatii.

Task 5: Durata totala a fiecarei activitati.

Task 6: Filtrarea activitatilor pentru care mai mult de 90% din aparitii au durata mai mica de 5 minute.

Obiectivele secundare sunt: respectarea paradigmei programarii orientate pe obiect, folosirea functiilor pe stream-uri, folosirea expresiilor lambda si crearea unei arhive de tipul .jar pentru portabilitate si testarea mai usoara a programului.

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

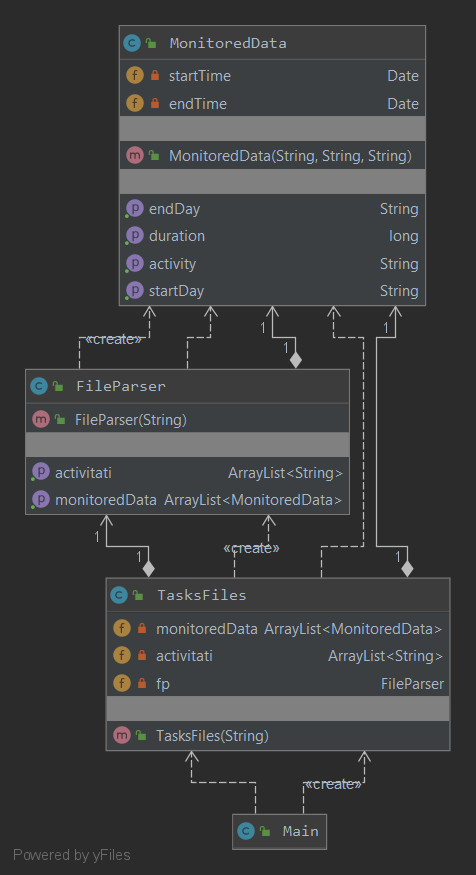
La executarea programului, datele despre activitatea persoanei sunt citite din fisierul de intrare si informatiile despre fiecare activitate vor fi salvate intr-o lista de obiecte de tipul MonitoredData, care are un String activity si doua variabile de tipul Date, startTime si endTime. Cu ajutorul acestei liste, vor fi implementate cele 6 task-uri. Pentru implementarea celor 6 task-uri, colectiile au fost convertite in stream-uri, un stream fiind o secventa de elemente pe care se pot aplica operatii agregate secventiale si paralele. O operatie aplicata unui stream poate fi vazuta ca un query pe acel stream. Majoritatea operatiilor pe stream-uri accepta parametri, in care se pot folosi expresii lambda. O expresie lambda apeleaza o metoda care are un nume sau creeaza o metoda anonima care ofera un comportament parametrizat. Elementul din stanga sagetii (->) este parametrul expresiei lambda, iar elementul din dreapta sagetii este corpul expresiei lambda, locul unde este implementata logica expresiei.

Pentru a impartii metodele proiectului, am creat cele 4 clase: clasa MonitoredData, care defineste informatiile despre fiecare activitate care apare in fisierul de intrare, clasa FileParser, care se ocupa cu citiriea din fisierul de intrare si cu scrierea in fisierele de iesire, pentru fiecare task, clasa TasksFiles, unde sunt implementate metodele pentru rezolvarea task-urilor si clasa Main, care contine metoda main a proiectului.

Rezultatele obtinute in urma executarii celor 6 task-uri vor fi salvate in task\_1.txt, task\_2.txt, task\_3.txt, task\_4.txt, task\_5.txt si task\_6.txt.

3. Proiectare

3.1. Diagrama UML



3.2. Proiectare clase

***Clasa MonitoredData***: aceasta clasa defineste conceptul de activitate care a fost inregistrata pe intreaga perioada. Campurile clasei sunt: startTime, de tipul Date, reprezentand timpul de start al activitatii, endTime, de tipul Date, reprezentand timpul de final al activitatii si activity de tipul String, reprezentand numele activitatii.

***Clasa FileParser***: aceasta clasa se ocupa cu citiriea din fisierul de intrare si cu scrierea in fisierele de iesire. Campurile clasei sunt: monitoredData, de tipul ArrayList <MonitoredData>, reprezentand lista completa de activitati care au fost realizate pe intreaga perioada si activitati, de tipul ArrayList <String>, reprezentand lista de nume unice ale activitatilor care apar in lista monitoredData.

***Clasa TasksFiles***: aceasta clasa se ocupa cu implementarea metodelor prin care se rezolva cerintele celor 6 task-uri. Campurile clasei sunt: monitoredData, de tipul ArrayList <MonitoredData>, reprezentand lista completa de activitati care au fost realizate pe intreaga perioada, activitati, de tipul ArrayList <String>, reprezentand lista de nume unice ale activitatilor care apar in lista monitoredData si fp, de tipul FileParser. reprezentand obiectul cu ajutorul caruia se apeleaza metodele clasei FileParser.

***Clasa Main***: aceasta clasa este clasa main a proiectului. In aceasta clasa se creeaza un obiect nou de tipul TasksFiles, in parametrul constructorului se scrie numele fisierului de input: “Activities.txt”, iar apoi prin obiectul creat se apeleaza metodele prin care se rezolva cerintele celor 6 task-uri.

4. Implementare si testare

4.1. Metode

***Clasa MonitoredData***

public MonitoredData (String startTime, String endTime, String activity)

- constructorul clasei, campul activity al clasei primeste String-ul activity primit ca parametru, iar pentru startTime si endTime, conversia din tipul String in tipul Date se realizeaza cu ajutorul metodei parse aplicata pentru un nou obiect de tipul SimpleDateFormat, cu pattern-ul “yyyy-MM-dd HH:mm:ss”

public String toString ( )

- metoda suprascrisa toString, pentru a afisa un obiect de tipul MonitoredData

public String getStartDay ( )

- metoda care returneaza doar datele despre zi a campului startTime (formatul “yyyy-MM-dd”) sub forma de string

public String getEndDay ( )

- metoda care returneaza doar datele despre zi a campului endTime (formatul “yyyy-MM-dd”) sub forma de string

public long getDuration ( )

- metoda care returneaza durata unei activitati in milisecunde, aceasta fiind egala cu diferenta dintre cele doua campuri de tipul Date, endTime si startTime, pe care a fost aplicata metoda getTime ( )

public String getActivity ( )

- metoda care returneaza campul activity

***Clasa FileParser***

public FileParser (String path)

- constructorul clasei, in acesta se apeleaza metoda read, avand ca parametru String-ul path

public void write (String path, String s)

- metoda care primeste 2 String-uri ca parametrii, primul fiind path-ul, numele fisierului .txt in care se va scrie, iar al doilea fiind string-ul care se va scrie in fisier

- scrierea in fisier se realizeaza cu ajutorul metodei write apelata printr-un obiect de tipul PrintWriter

public void read (String path)

- metoda care primeste un String ca parametru, acesta reprezentand path-ul, numele fisierului .txt din care se vor citi datele

- liniile fisierului de intrare sunt salvate intr-un Stream de obiecte de tip String, acest stream fiind apoi colectat intr-o lista de string-uri, cu ajutorul metodei collect

- se parcurge lista de string-uri, fiecare element al listei fiind impartit prin regex-ul “\t+”, iar string-urile rezultate (vor fi in numar de 3), sunt folosite pentru constructorul unui obiect nou de tipul MonitoredData, care este apoi adaugat in lista monitoredData; de asemenea, daca al 3-lea string care rezulta din impartirea elementului iterat (numele activitatii) nu se gaseste in lista activitati, atunci va fi adaugat listei – astfel, in lista respectiva vor fi salvate toate numele activitatilor care apar in fisierul de intrare

public ArrayList <MonitoredData> getMonitoredData ( )

- metoda care returneaza lista monitoredData a clasei

public ArrayList <String> getActivitati ( )

- metoda care returneaza lista activitati a clasei

***Clasa TasksFiles***

public TasksFiles (String path)

- constructorul clasei, primeste ca parametru String-ul path, numele fisierului, care este folosit pentru a initializa campul fp al clasei, iar apoi campurile monitoredData si activitati primesc valorile returnate de cele doua metode ale obiectului fp, getMonitoredData ( ), respectiv getActivitati ( )

public void createTask1 ( )

- pentru rezolvarea primului task, se parcurge lista monitoredData, iar fiecare obiect iterat este concatenat la un string

- la final, string-ul este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul “task\_1.txt”, numele fisierului

public void createTask2 ( )

- pentru rezolvarea celui de-al doilea task, intr-o variabila de tipul long se salveaza numarul elementelor distincte din stream-ul care are ca sursa lista monitoredData, mapat dupa String-urile returnate de metoda getStartDay ( ) aplicata fiecarui element din expresia lambda

- la final, rezultatul este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul “task\_2.txt”, numele fisierului

public HashMap <String, Integer> createTask3 ( )

- pentru rezolvarea celui de-al treilea task, se creeaza un HashMap, activities, avand cheia un String, reprezentand numele activitatii, iar valoarea un Integer, reprezentand numarul de aparitii al activitatii pe intreaga perioada

- intr-o lista de String-uri, allActivities, se vor colecta toate String-urile returnate de metoda getActivity ( ), dupa care a fost mapat stream-ul care are ca sursa lista monitoredData

- pentru fiecare element al listei de String-uri activitati a clasei, care contine numele activitatilor distincte, se va pune in HashMap-ul activities o noua intrare: cheia acestei intrari va fi String-ul iterat din lista activitati, iar valoarea intrarii va fi numarul de elemente al stream-ului care are ca sursa lista allActivities, filtrat astfel incat sa contina doar String-urile egale cu String-ul iterat din lista activitati

- se parcurge HashMap-ul activities si datele se salveaza intr-un String, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul “task\_3.txt”, numele fisierului

- la final, se returneaza HashMap-ul activities obtinut ca rezultat

public HashMap <Integer, HashMap<String, Integer>> createTask4 ( )

- pentru rezolvarea celui de-al patrulea task, se creeaza un HashMap, activityCountDays, avand cheia un Integer, reprezentand numarul zilei, iar valoarea un HashMap care are cheia un String, reprezentand numele activitatii, iar valoarea un Integer, reprezentand numarul de aparitii al activitatii pe durata zilei

- se creeaza o lista de String-uri, allStartDays, in care se adauga toate valorile returnate de metoda getStartDay( ) a elementelor din lista monitoredData

- se creeaza o lista de String-uri, startDays, in care se adauga elementele unice ale listei allStartDays

- se itereaza lista startDays, pentru fiecare iteratie se creeaza un HashMap nou, activityCount, avand cheia un String, reprezentand numele activitatii, iar valoarea un Integer, reprezentand numarul de aparitii al activitatii in ziua respectiva; se itereaza apoi lista de String-uri activitati a clasei, care contine numele activitatilor distincte, pentru fiecare element se pune o noua intrare in HashMap-ul activityCount: cheia acestei intrari reprezinta elementul iterat al listei activitati, iar valoarea intrarii reprezinta numarul de elemente al stream-ului care are ca sursa lista monitoredData, filtrat de urmatoarea conditie: String-ul care reprezinta ziua curenta sa fie egal fie cu cel returnat de metoda getStartDay ( ), fie de metoda getEndDay ( ), si String-ul care reprezinta numele activitatii sa fie egal cu cel returnat de metoda getActivity ( ); dupa ce se itereaza toata lista activitati, se pune o noua intrare in HashMap-ul activityCountDays: cheia acestei intrari reprezinta numarul zilei curente (calculat ca numarul de elemente din acest HashMap + 1), iar valoarea intrarii reprezinta HashMap-ul activityCount

- se parcurge HashMap-ul activityCountDays si datele se salveaza intr-un String, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul “task\_4.txt”, numele fisierului

- la final, se returneaza HashMap-ul activityCountDays obtinut ca rezultat

public HashMap <String, Duration> createTask5 ( )

- pentru rezolvarea celui de-al cincilea task, se creeaza un HashMap, durations, avand cheia un String, reprezentand numele activitatii, iar valoarea un obiect de tipul Duration, reprezentand durata totala a activitatii pe intreaga perioada

- se itereaza lista de String-uri activitati, la fiecare astfel de iteratie intr-o lista de elemente de tipul Long, duration, se colecteaza stream-ul care are ca sursa lista monitoredData, mapat dupa elementele de tipul Long returnate de metoda getDuration ( ), filtrat dupa urmatoarea conditie: String-ul returnat de metoda getActivity( ) trebuie sa fie egal cu String-ul iterat al listei activitati; este iterat apoi lista duration, fiecare element fiind adaugat la o variabila de tipul long, time; variabila time este parsata apoi catre un obiect de tipul Duration, dur, folosind metoda ofMillis; in HashMap-ul durations se pune o intrare noua: cheia acestei intrari reprezinta String-ul iterat al listei activitati, iar valoarea intrarii reprezinta obiectul de tipul Duration, dur

- se parcurge HashMap-ul durations si datele se salveaza intr-un String, fiecare durata fiind scrisa sub forma HH:mm:ss, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul “task\_5.txt”, numele fisierului

- la final, se returneaza HashMap-ul durations obtinut ca rezultat

public List <String> createTask6 ( )

- pentru rezolvarea celui de-al saselea task, se creeaza o lista de String-uri, activities

- se itereaza lista de String-uri activitati, la fiecare astfel de iteratie intr-o lista de elemente de tipul Long, duration, se colecteaza stream-ul care are ca sursa lista monitoredData, mapat dupa elementele de tipul Long returnate de metoda getDuration ( ), filtrat dupa urmatoarea conditie: String-ul returnat de metoda getActivity( ) trebuie sa fie egal cu String-ul iterat al listei activitati; intr-o variabila de tipul long, nr, se salveaza numarul elementelor din stream-ul care are ca sursa lista duration, filtrat dupa urmatoarea conditie: valoarea elementelor impartita la 6000 sa fie mai mica decat 5; intr-un intreg, nrTotal, se pune dimensiunea listei duration, care reprezinta numarul total de aparitii a unei activitati pentru ziua respectiva; in cazul in care variabila nr reprezinta mai mult decat 90% din variabila nrTotal, se adauga la lista activities String-ul iterat al listei activitati

- se parcurge lista activities si datele se salveaza intr-un String, care este trimis ca al doilea parametru al metodei write a obiectului fp, primul parametru fiind string-ul “task\_6.txt”, numele fisierului

- la final, se returneaza lista activities obtinuta ca rezultat

***Clasa Main***

public static void main (String[] args)

- este clasa main a proiectului

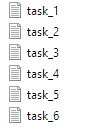
- se creeaza un obiect nou de tipul TasksFiles, tasks, in parametrul constructorului se scrie numele fisierului de input: “Activities.txt”, iar apoi prin obiectul creat se apeleaza metodele prin care se rezolva cerintele celor 6 task-uri

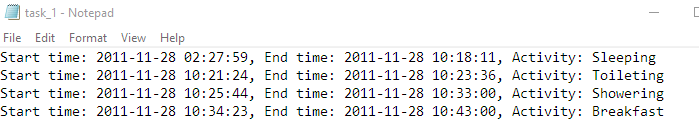
4.2. Testare

In urma executiei comenzii:

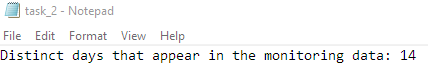


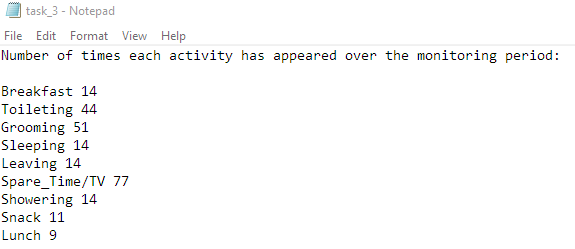
vor fi create urmatoarele fisiere:

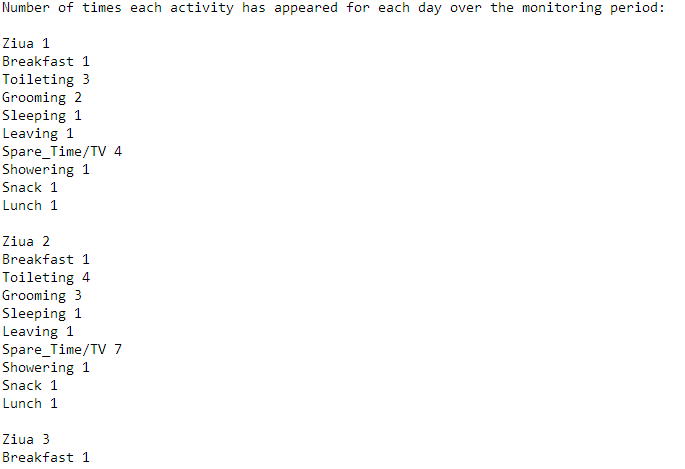


avand continuturile:

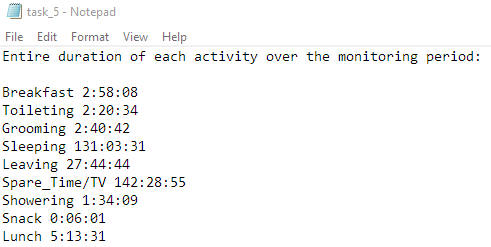
etc

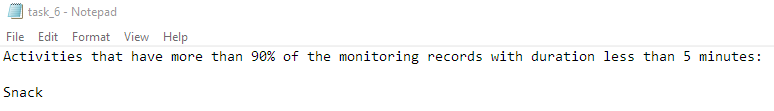






etc





5. Concluzii

Acest proiect functioneaza conform cerintelor task-urilor prezentate, insa poate fi imbunatatit prin adaugarea unor noi task-uri, sau chiar prin adaugarea mai multor fisiere de intrare pentru a se putea face o statistica pentru diferite persoane. Proiectul m-a ajutat sa invat cum se utilizeaza stream-urile si expresiile lambda, acestea fiind utile pentru a crea un cod mult mai compact.

6. Bibliografie

1. https://www.geeksforgeeks.org/stream-in-java/

2. https://www.baeldung.com/java-8-streams-introduction

3. http://coned.utcluj.ro/~salomie/PT\_Lic/4\_Lab/Assignment\_5/Assignment\_5.pdf

4. http://tutorials.jenkov.com/java-date-time/parsing-formatting-dates.html

5. https://docs.oracle.com/javase/8/docs/api/java/time/Duration.html