**PROCESSING SENSOR DATA OF DAILY LIVING ACTIVITIES**

**Moldovan Vlad Madalin**

**CUPRINS:**

**1.Obiectivul temei**

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

**3. Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii , packages, algoritmi, interfata utilizator)**

**3.1 Decizii de proiectare**

**3.2 Diagrama UML**

**4.Implementare**

**4.1 Clasa MonitoredData**

**4.2 Clasa Tasks**

**4.3 Clasa FileWrite**

**4.4 Clasa MainClass**

**5.Concluzii**

**6.Bibliografie**

**1.Obiectivul temei**

Obiectivul principal al temei este de a implementa a aplicatie pentru analiza comportamentului unei personae inregistrate de un set de senzori instalati in casa pentru a rezolva cele 6 task-uri primite .Obiectivul secundar al temei este de a utiliza lambda expressions si stream processing pentru indeplinirea celor 6 task-uri , dar si generarea rezultatelor intr-un fisier .txt ( task\_number.txt ) .

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

Cerinta problemei este de a implemente cele 6 task-uri .

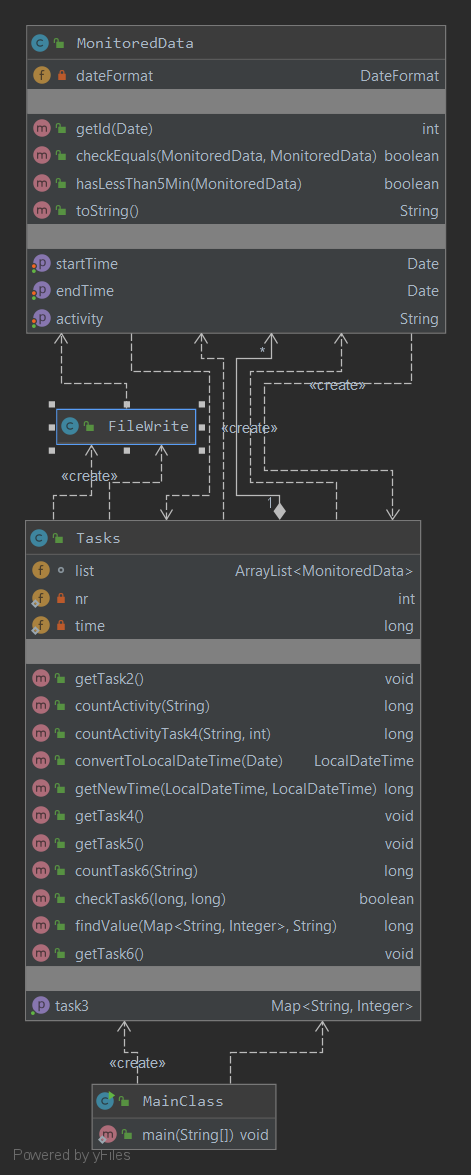
* Task 1 -> Definiti o clasa MonitoredDate cu 3 campuri : start time , end time and activity (string) . Cititi datele din fisierul Activity.txt si folositi streams si impartiti fiecare linie in 3 parti : start\_time , end\_time si activity\_label , si creati o lista de obiecte de tipul MonitoredDate.
* Task 2 -> Numarati zilele distincte care apar in fisierul dat.
* Task 3 -> Numarati de cate ori fiecare activitate apare pe intreaga durata de monitorizare ( Returnati o structura de tipul Map<String,Integer> care reprezinta fiecare activitate distincta si numarul ei de aparitii in fisier ; cheia structurii va fi reprezentata de un obiect String care corespunde cu numele activitatii , si valoarea structurii va fi reprezentata de un obiect de tipul Integer care corespunde numarului de aparitii a activitatii in fisierul dat) .
* Task 4 -> Numarati de cate ori apare fiecare activitate in fiecare zi pe durata de monitorizare ( Returnati o structura de tipul Map<Integer, Map<String,Integer>> care contine numarul de aparitii a fiecarei activitati intr-o zi ; cheia structurii va fi un obiect de tipul Integer care corespunde numarului zilei din fisierul dat , si valoarea structurii va fi o alta structura reprezentata de Map<String,Integer> unde cheia va fi un String care corespunde numelui activitatii , iar valoarea va fi un Integer care corespunde numarului de aparitii a activitatii in ziua precizata ).
* Task 5 -> Pentru fiecare activitate calculate durata totala pe perioada monitorizarii ( Returnati o structura de tipul Map<String , LocalTime> in care cheia va fi reprezentata de un obiect String care corepsunde numelui activitatii , iar valoarea va fi reprezentata de un obiect de tipul LocalTime care corespunde duratei totale in care a fost efectuata activitatea pe perioada monitorizarii ) .
* Task 6 -> Selectati activitatile care au mai mult de 90% dintre datele inregistrate cu durata mai scurta de 5 minute , colectati datele intr-un List<String> care contine numele distincte ale activitatilor si returnati lista .

**3.** **Proiectare (decizii de proiectare, diagrame UML, structure de date, proiectare clase, interfete, relatii , packages, algoritmi, interfata utilizator)**

**3.1 Decizii de proiectare**

Pentru implementarea aplicatiei au fost implementate 4 clase : MonitoredData , Tasks , FileWrite , MainClass . Aceste clase vor ajuta la executarea task-urilor primate in cerinta .In rezolvarea acestei aplicatii au fost folosite lambda expressions si stream processing conform cerintei . Prima clasa care a fost implementata a fost clasa MonitoredData , dupa care clasa Tasks pentru rezolvarea task-urile din cerinta , iar pentru afisarea rezultatelor in format .txt a fost implementata clasa FileWrite pentru scrierea datelor intr-un fisier .

**3.2 Diagrama UML**

****

**4.Implementare**

**4.1 Clasa MonitoredData**

Clasa MonitoredData este utilizata pentru stocarea informatilor din fisierul citit . Aceasta clasa contine 3 atribute : startTime ( de tipul Date ) care reprezinta data la care a inceput activitatea , endTime ( de tipul Date ) care reprezinta data la care s-a terminat activitatea , Activity ( de tipul String ) care reprezinta numele activitatii care se desfasoara .

Metode :

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

- reprezinta constructorul clasei care primeste ca parametrii : startTime ( de tipul Date ) , endTime (de tipul Date) , activity (de tipul string)

* public Date getStartTime()

- reprezinta un getter pentru obtinerea datei la care a inceput activitatea

* public void setStartTime(Date startTime)

- reprezinta un setter pentru setarea datei la care a inceput activitatea

* public Date getEndTime()

- reprezinta un getter pentru obtinerea datei la care s-a sfarsit activitatea

* public void setEndTime(Date endTime)

- reprezinta un setter pentru setarea datei la care s-a sfarsit activitatea

* public String getActivity()

- reprezinta un getter pentru obtinerea numelui activitatii unui obiect de acest tip

* public int getId(Date m)

- reprezinta o metoda pentru determinarea unui id unic pentru fiecare zi din an , aceasta va fi utila in rezolvarea task-urilor primate

* public boolean hasLessThan5Min(MonitoredData m)

- reprezinta o metoda care verifica daca un obiect de tipul MonitoredData are durata activitatii mai mica decat 5 minute , in caz afirmativ va returna true , iar in caz contrar va returna false . Aceasta metoda este utila pentru rezolvarea task-ului 6

**4.2 Clasa Tasks**

Clasa Tasks este utilizata pentru a rezolva cerintele problemei care constau in cele 6 task-uri . Rezolvarea acestora se va face prin metodele acestei clase . Aceasta clasa contine ca atribut o lista de MonitoredData care reprezinta toate datele citite din fisier .

Metode:

* public Tasks()

- reprezinta constructorul default al clasei

* public Tasks(String file) throws IOException

- reprezinta constructorul clasei care primeste ca parametru numele fisierului din care se vor citi datele si tot odata prin acesta se va rezolvaa doua parte a task-ului 1.

- Cerinta task-ului 1 este : Definiti o clasa MonitoredDate cu 3 campuri : start time , end time and activity (string) . Cititi datele din fisierul Activity.txt si folositi streams si impartiti fiecare linie in 3 parti : start\_time , end\_time si activity\_label , si creati o lista de obiecte de tipul MonitoredDate.

- se va prelucra fisierul primit prin impartirea fiecarei linii in 3 partii , prima parte va reprezenta startTime-ul , a doua parte va reprezenta de endTime , iar ultima parte va fi numele activitatii . Toate aceste informatii vor fi stocate in atributul clasei ArrayList<MonitoredData> list .

* public void getTask2() throws IOException

- reprezinta metoda prin care se va rezolva cel de al doilea task

- Cerinta : Numarati zilele distincte care apar in fisierul dat.

- Folosind stream processing se va parcurge lista si se va forma o noua lista care contine doar id-urile zilelor ( metoda getId prezentata mai sus in clasa MonitoredData ) distincte , rezultatul acestui task va fi size-ul listei noi create .

* public long countActivity( String a )

- reprezinta o metoda cu ajutorul careia se va gasi de cate ori este executata activitatea cu numele “a” ( primit ca paramentru ) in lista noastra de date .

* public Map< String , Integer > getTask3() throws IOException

- reprezinta metoda care ne va ajuta sa rezolvam cel de al treilea task al cerintei

- Cerinta : Numarati de cate ori fiecare activitate apare pe intreaga durata de monitorizare ( Returnati o structura de tipul Map<String,Integer> care reprezinta fiecare activitate distincta si numarul ei de aparitii in fisier ; cheia structurii va fi reprezentata de un obiect String care corespunde cu numele activitatii , si valoarea structurii va fi reprezentata de un obiect de tipul Integer care corespunde numarului de aparitii a activitatii in fisierul dat) .

- Pentru formarea structurii Map<String,Integer> se va parcurge lista cu datele din fisier si se va forma o noua lista care va contine o lista cu numele tuturor activitatilor din fisier ( distincta ) , iar acestea vor reprezenta cheia structurii , iar valoarea structurii ( numarul de aparitie a fiecarei activitatii ) va fi reprezentata prin apelarea functiei countActivity mentionate mai sus

* public long countActivityTask4(String a,int id)

- reprezinta o metoda care calculeaza numarul de aparitii a unei activitatii “a” intr-o anumita zi ( dupa id-ul id )

* public LocalDateTime convertToLocalDateTime(Date dateToConvert)

- reprezinta o metoda de convertire a unui obiect de tip Date intr-un obiect de tip LocalDateTime

* public long getNewTime(LocalDateTime a,LocalDateTime b)

- reprezinta o metoda care calculeaza diferenta dintre doua LocalDateTime , rezultatul acesteia va fi in secunde

* public void getTask4() throws IOException

- reprezinta metoda care ne va ajuta sa rezolvam cel de al paturela task al cerintei

- Cerinta : Numarati de cate ori apare fiecare activitate in fiecare zi pe durata de monitorizare ( Returnati o structura de tipul Map<Integer, Map<String,Integer>> care contine numarul de aparitii a fiecarei activitati intr-o zi ; cheia structurii va fi un obiect de tipul Integer care corespunde numarului zilei din fisierul dat , si valoarea structurii va fi o alta structura reprezentata de Map<String,Integer> unde cheia va fi un String care corespunde numelui activitatii , iar valoarea va fi un Integer care corespunde numarului de aparitii a activitatii in ziua precizata ).

- se va parcurge lista cu datele din fisier si se va forma o noua lista de tipul List<Integer> care va contine id-ul fiecarei zi distincte , dupa care aceasta noua lista va fi parcursa pentru a determina numarul de aparitii a fiecarei activitatii in ziua respective ( dupa id-ul acesteia ) .

* public void getTask5() throws IOException

- reprezinta metoda care ne va ajuta sa rezolvam cel de al cincilea task al cerintei

- Cerinta : Pentru fiecare activitate calculate durata totala pe perioada monitorizarii ( Returnati o structura de tipul Map<String , LocalTime> in care cheia va fi reprezentata de un obiect String care corepsunde numelui activitatii , iar valoarea va fi reprezentata de un obiect de tipul LocalTime care corespunde duratei totale in care a fost efectuata activitatea pe perioada monitorizarii ) .

- pentru rezolvarea acestei cerinte am implementat o structura de tipul Map<String,Integer> in care se va retine numele activitatii si durata acesteia in minute . Se va parcurge lista care contine datele din fisier si se va forma o noua lista de tipul List<String> in care se vor retine numele activitatilor distincte dupa care aceasta lista se va parcurge si se va calcula timpul pentru fiecare activitate in parte cu ajutorul metodei getNewTime() .

* public long countTask6(String a)

- reprezinta metoda care numara de cate ori o activitate este realizata in mai putin de 5 minute

* public boolean checkTask6(long a,long b)

- reprezinta metoda care verifica daca 90% dintre inregistrarile unei activitati s-au desfasurat in mai putin de 5 minute si va returna true daca este afirmativ sau false daca este negative

* public long findValue(Map<String,Integer> map,String a)

- reprezinta o metoda care returneaza valoarea listei dupa cheia data

* public void getTask6() throws IOException

- reprezinta metoda care ne va ajuta sa rezolvam cel de al 6 lea task al cerintei

- Cerinta : Selectati activitatile care au mai mult de 90% dintre datele inregistrate cu durata mai scurta de 5 minute , colectati datele intr-un List<String> care contine numele distincte ale activitatilor si returnati lista .

- pentru rezolvarea acestia au fost utilizate cele 3 metode precizate mai sus : countTask6(String a), long findValue(Map<String,Integer> map,String a) , checkTask6(long a,long b)

**4.3 Clasa FileWrite**

Aceasta clasa este utilizata pentru afisarea rezultatelor fiecarui task intr-un format task\_numer.txt

Metode:

* public FileWrite(ArrayList<MonitoredData> list, String id) throws IOException

- constructor folosit pentru afisarea rezultatului de la primul task

* public FileWrite(int x,String id) throws IOException

- constructor folosit pentru afisarea rezultatului de la al doilea task

* public FileWrite(Map<String,Integer> map, String id) throws IOException

- constructor folosit pentru afisarea rezultatului de la al treilea task si cel de al cincilea task

* public FileWrite(Map<Integer, Map<String, Integer>> map) throws IOException

- constructor folosit pentru afisarea rezultatului de la al patrulea task

* public FileWrite(List<String> list, String id)

- constructor folosit pentru afisarea rezultatului de la al saselea task

**4.4 Clasa MainClass**

Aceasta clasa reprezinta clasa main a aplicatiei care se va rula pentru executia programului .

**5. Concluzii**

Au fost implementate cele 6 task-uri din cerinta . Din aceasta tema am invata sa lucrez mai bine cu lambda expressions , dar si cu stream processing .

**6. Bibliografie**

1. <https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html>

2. <https://www.oracle.com/technical-resources/articles/java/ma14-java-se-8-streams.html>

3. <https://winterbe.com/posts/2014/07/31/java8-stream-tutorial-examples/>