Documentatie

PROCESSING SENSOR DATA OF DAILY LIVING ACTIVITIES



Tehnici de programare

Tema5

Realizat de Balc Horia-Ovidiu

Grupa 30227

Cuprins

1 . Cerinte Functionale

2 . Obiectivul temei

3 . Modelare

4 . Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

5 . Implementare

6. Testare

7 . Concluzie

8 . Bibliografie

# Cerinte Functionale

Proiectati si implementati un model de aplicatie care va avea rolul de a procesa informatii in legatura cu activitatile de zi cu zi ale unei persoane.

Programul:

* utilizarea unui fisier de tipul jar.
* citirea datelor din fisierul Activities.txt, unde fiecare linie este impartita in 3 sectiuni (cu ajutorul tastei tab ) prima reprezinta data (cu formatul acesta: anul, luna, ziua, ora, minutul,secunda ) de incepere a activitatii, a doua reprezinta data (cu formatul prezentat mai sus) de terminare a activitatii, iar cea de-a treia este numele activitatii care se desfasoara in acel interval de timp.
* scrierea in cate un fisier pentru fiecare cerinta (task), adica creeare a 6 fisiere de tip txt fiecare avand nume sugestiv (“Task1.txt”, “Task2.txt”, “Task3.txt”, “Task4.txt”, “Task5.txt”, “Task6.txt”).
* implementarea celor sase cerinte.

# 2Obiectivul Temei

2.1 Obiectivul principal

Obiectivul temei este crearea, in limbajul JAVA, a unui model de aplicatie care are rolul de a procesa informatii in legatura cu activitatile de zi cu zi ale unei persoane intr-un interval de timp in care acea persoana este monitorizata (cateva zile). Aceasta tema are practicabilitate in viata reala si are multe obiective importante care stau la baza programarii in limbajul JAVA.

2.2 Obiectivul secundare

Obiectivele secundare sunt: - folosirea programarii obiectuale prin utilizarea incapsularii

- scrierea variabilelor in stilul Camel Case

-utilizarea metodelor(maxim 30 de linii) si claselor( maxim 300 de linii)

-folosirea Stream-urilor la fiecare metoda ce implementeaza o cerinta, cu ajutorur carora se indeplinesc cerintele intr-un mod mai efficient si mai direct (reduce numarul de parcurgeri, deci mai putine linii de cod)

- utilizarea a lambda expressions (in Stream-uri) la fiecare metoda ce implementeaza o cerinta care necesita cel putin o verificare (Task3, Task4, Task5, Task6), cu ajutorur carora se indeplinesc cerintele intr-un mod mai efficient si mai direct (reduce numarul de verificari, deci mai putine linii de cod)

3. Modelare

In aceasta sectiune vom explica modelarea proiectului. Acesta este structurat pe baza a trei clase (Main, MonitoredData si Metode).

Clasa MonitoredData este utilizata pentru a formata mai usor linia din fisierul de tip txt care se citeste si se va stoca intr-un obiect de tipul MonitoredData.

Clasa Metode, dupa cum ii spune numele, reprezinta clasa ce are toate metodele programului adica ( cele 6 metode ce indeplinesc cele 6 cerinte date, o metoda pentru scrierea in cate un fisier pentru fiecare task si o metoda ,ajutatoare pentru metoda de citire (Task1), care are rolul de a transforma linia citita din fisierul Activities.txt intr-un obiect de tipul MonitoredData).

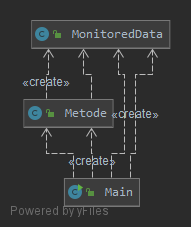
Clasa Main, dupa cum ii spune numele, reprezinta clasa statica si rulabila unde se instantiaza obiectele si se apeleaza metodele pentru toate cele 6 cerinte si pentru metoda de scriere.

4.Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

4.1Decizii de proiectare

Am decis sa modelez acest proiect cu ajutorul a 3 clase pentru a fi mai simplu de inteles si totodata de scris cod. Astfel, clasa MonitoredData reprezinta activitatea cu toate informatiile necesare (data de incepere, data de final si denumirea activitatii), clasa Metode reprezinta actiunile pe care le poate face programul, iar clasa Main face legatura intre celelalte clase, si aici se implementeaza toate task-urile, fiind singura clasa rulabila.

4.2Diagrame UML



5.Implementare

Proiectul este format din trei clase care vor fi descrise in mai jos in detaliu.

Clasa MonitoredData este o clasa ce are 3 variabile de tip private (startTime, endTime, activity), toate cele trei fiind obiecte de tip String. Constructorul acestei clase are ccca parametrii 3 stringuri. Primul string reprezinta data (cu formatul acesta: anul, luna, ziua, ora, minutul,secunda ) la care se incepe activitatea, al doilea reprezinta data (cu formatul acesta: anul, luna, ziua, ora, minutul,secunda ) la care se termina activitatea, iar al treilea reprezinta denumirea activitatii. Metodele pe care le are aceasta metoda vor fi explicate mai jos.

Metoda getStartTime este un getter ce nu primeste nici un argument ca parametru, aceasta returneaza un string ce reprezinta data de inceput a activitatii pe care o are obiectul de tip MonitoredData prin care sa apelat metoda.

Metoda getEndTime este un getter ce nu primeste nici un argument ca parametru, aceasta returneaza un string ce reprezinta data de sfarsit a activitatii pe care o are obiectul de tip MonitoredData prin care sa apelat metoda.

Metoda getActivity este un getter ce nu primeste nici un argument ca parametru, aceasta returneaza un string ce reprezinta numele activitatii pe care o are obiectul de tip MonitoredData prin care sa apelat metoda.

Metoda setStartTime este un setter ce primeste ca argument un obiect de tip String ce va fi setat ca fiind noul startTime al obiectului de tip MonitoredData prin care sa apelat metoda (care nu returneaza nimic , deoarece este de tip void).

Metoda setEndTime este un setter ce primeste ca argument un obiect de tip String ce va fi setat ca fiind noul endTime al obiectului de tip MonitoredData prin care sa apelat metoda (care nu returneaza nimic , deoarece este de tip void).

Metoda setActivity este un setter ce primeste ca argument un obiect de tip String ce va fi setat ca fiind noul activity al obiectului de tip MonitoredData prin care sa apelat metoda (care nu returneaza nimic , deoarece este de tip void).

Clasa Metodeeste o clasa ce nu are variabile, este facuta doar pentru a implementa metodele programului. Metodele sunt explicate in detaliu mai jos.

Metoda citire (Task1) este o metoda ce primeste ca parametru o lista (ArrrayList) de obiecte de tipul MonitoredData m, metoda returneaza o lista (ArrayList) de obiecte de tipul MonitoredData. Mai intai se initializeaza un string cu valoarea numelui fisierului de tip txt din care se citeste (Activities.txt) si se instantiaza o lista de stringuri numita list. Urmatorul pas este sa colecteze toate liniile din fisier in Lista list prin intermediul unui obiect de tip Stream<String> cu ajutorul funciei predefinite lines (din clasa default Files) si get (din clasa default Paths). Pasul descris precedent este incadrat intr-un bloc try catch in cazul in care se produce o exceptie de tipul IOException. In ultima parte a metodei se initializeaza o variabila int i cu valoarea 0, se verifica daca lista este goala in caz negative va intra intr-un foreach unde se parcurge lista list. Aici se incrementeaza variabila i, se initializeaza un obiect de tip MonitoredData mD, se apeleaza metoda impartire care va fi descrisa mai jos, dupa care se adauga mD in lista primita ca parametru. La finalul metodei se returneaza lista m (cea primita ca parametru).

Metoda impartire este o metoda ce primeste ca parametru un obiect de tip String s si altul de tip MonitoredData mD, si nu returneaza nimic (de tipul void). Aici folosim un sir de stringuri linie care ia valoarea sirului s impartit de tasta tab (cu ajutorul functiei predefinite split), dupa care se seteaza data de start, de final si numele activitatii a obiectului mD cu cele trei stringuri din linie (linie[0], linie[1], linie[2]).

Metoda count (Task2) este o metoda ce primeste ca parametru o lista de obiecte de tip MonitoredData mD si returneaza un int ce reprezinta numarul de zile distincte. Pentru inceput am initializat o lista de stringuri numita lista si am parcurs cu un forEach lista mD, unde am extras din stringul startTime doar data si am adaugat Stringul in lista lista. La final initializam o variabila (c) de tip long care ia valoarea numarului de zile distincte, cu ajutorul functiilor de stream aplicate listei de stringuri ( distinct si count), si returnam variabila c dup ace o convertim la int (cast).

Metoda Task3 este o metoda ce primeste ca parametru o lista de de obiecte de tip MonitoredData mD si returneaza un obiect de tip HashMap <String, Integer>. La inceput instantiem un HashMap <String, Integer> map,si o lista de Stringuri lista si se face o parcurgere cu foreach a listei primate ca parametru, unde se adauga in lista de stringuri numele activitatii. Dupa care se mai parcurge cu foreach lista inca o data unde se numara pentru fiecare activitate de cate ori apare, si numarul se salveaza intr-o variabila long (count). Numararea se face cu ajutorul streamurilor si a lambda expressions prin metodele predefinite (stream,filtrer (unde este conditia, adica lambda expression), count), dupa care se adauga in map prin metoda put (numele activitatii si a valorii lui count castata la int). L afinal se iese din parcurgere si se returneaza map.

Metoda Task4 este o metoda ce primeste ca parametru o lista de de obiecte de tip MonitoredData mD si returneaza un obiect de tip HashMap < Integer, HashMap <String, Integer>>. Dupa instantierea a unui obiect de tipul HashMap <Integer, HashMap <String, Integer>> mapaMapei si a unei liste lista1, se parcurge lista primita ca parametru unde se extrage doar data din startTime si se adauga stringul in lista1. Pasul urmator este o parcurgere a listei lista1, unde se instantiaza un obiect de tip HashMap < Integer, String> si se face o parcuegere a liste mD. Aici se retine intr-un obiect de tipul String (str) data din variabila startTime a obiectului m, daca str si string sunt egale atunci se initializeaza un obiect de tip Sting t si se numara cu ajutorur stremurilor si lambda expression de cate ori s-a practicat activitatea in acea zi. In variabila count se va transmite valoarea cu ajutorul metodelor predefinite (stream, filtrer si count), dupa care se adauga in map Stringul t si variabila count castata la int. Dupa iesirea din primul for se adauga un obiect si in mapaMapei (tot cu ajutorul metodei default put), format din sizeul lui mapaMapei +1 si din map. La finalul metodei se returneaza variabila mapaMapei.

Metoda Task5 este o metoda ce primeste ca parametru o lista de de obiecte de tip MonitoredData mD si returneaza un obiect de tip HashMap <String, Integer>. Dupa instantierea a unui obiect de tipul HashMap <String, Integer> map, se parcurge lista primita ca parametru unde se extrage doar Activity care se adauga in Stringul t si se utilizeaza o lista de MonitoredData numita list care cu ajutorul streamului si a lambda expressions retine doar MonitoredData-urile ce au Activity-ul egal cu t. Astfel, daca nu exista stringul in map se parcurge lista list in care se calculeaza timpul total obtinut la pentru fiecare activitate. Acest lucru este datorat functiilor de convertire din string intr-un obiect de tip Data acestea fiind prinse intr-un bloc try catch in cazul in care se arunca o ParseException. Inainte de iesirea din ultimul for se insereaza in map (prin metoda default put) srtingul t si s ce reprezinta numarul total de sec (dup ace l-am impartit cu 1000 pentru a nu fi in milisecunde), iar la final se returneaza map.

Metoda Task6 este o metoda ce primeste ca parametru o lista de de obiecte de tip MonitoredData mD si returneaza un obiect de tip List <String>. Dupa instantierea a unui obiect de tipul HashMap <String, Integer>> map care ia valoare map-ului returnat de catre taskul 3 si a unei liste de Stringuri lista se parcurge lista primita ca parametru unde se extrage doar Activity-ul si se adauga in stringul string. Cu ajutorul streamului si a lambda expressions se initializeaza listaDate, o lista de Monitored data ce are variabila activity egala cu string. Urmatorul pas este initializarea variabilei locale activity de tip int cu valoare din map (prin metoda get), si a variabilei count de tip int cu 0 si in cazul in care lista lista nu contine stringul string se parcurge lista listaDate. Aici se verifica daca timpul pentru o activitate este mai mic decat 300 sec (5 minute) caz in care se incrementeaza count-ul. De asemenea calculul a fost simplificat de functiile de convertire din string intr-un obiect de tip Data acestea fiind prinse intr-un bloc try catch in cazul in care se arunca o ParseException. Inainte de a iesi din for-ul mare se verifica daca count este mai mare decat 90% din activitate, in caz afirmativ in lista lista se adauga stringul string (prin metoda default add), iar la finalul metodei se returneaza lista.

Metoda Scriere este o metoda ce ca parametru un int i si o lista de de obiecte de tip MonitoredData mD si arunca exceptia de tipul IOException. Primul pas este de a creea un fisier prin clasa FileWriter (predefinita) care va avea numele in functie de i (Taski.txt). Daca i=1 se scrie rezultatele pentru primul task in fisier, daca i=2 pentru al 2-lea, daca i=3 pentru al 3-lea, daca i=4 pentru al 4-lea, daca i=5 pentru al 5-lea, daca i=6 pentru al 6-lea, dupa care se inchide writerul.

Clasa Main este clasa ce contine metoda static rulabila a programului. Astfel aici am initializat un obiect de tip MonitoredData, o lista (ArrayList<MonitiredData>)si un obiect de tip Metode. Variabila mD primeste valoarea returnata de metoda citire explicate mai sus. Intr-un bloc try catch, deoarece metoda scriere arunca un IOException, am apelat metoda de scriere pentru toate cele 6 task-uri.

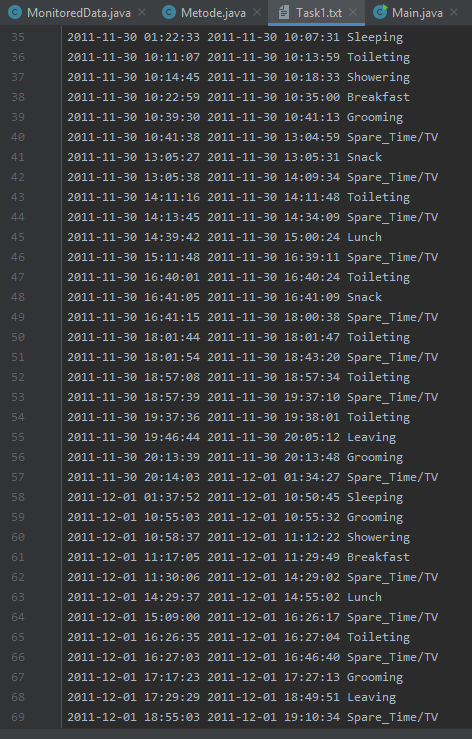
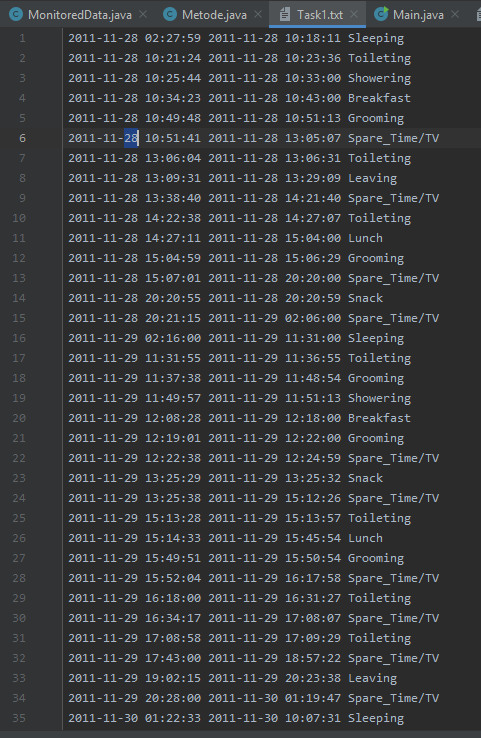
6.Testare

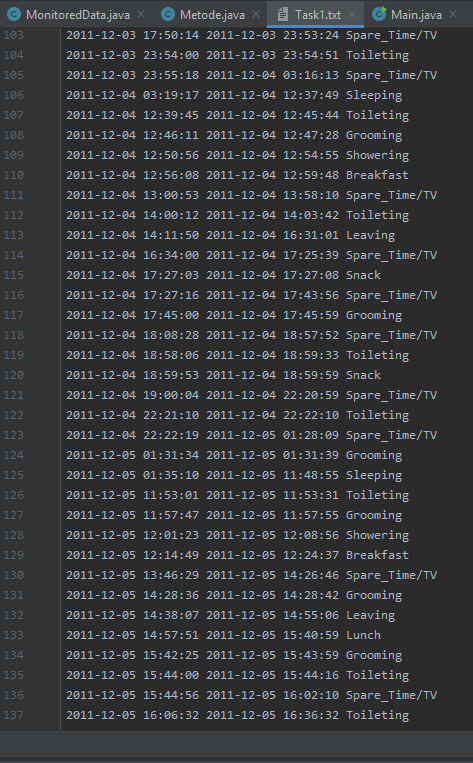
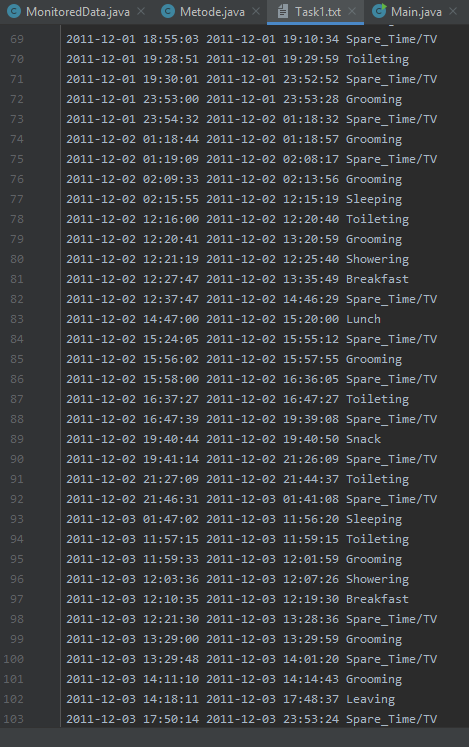
In urma instructiunii de mai jos se afiseaza pentru fiecare task rezultatele.

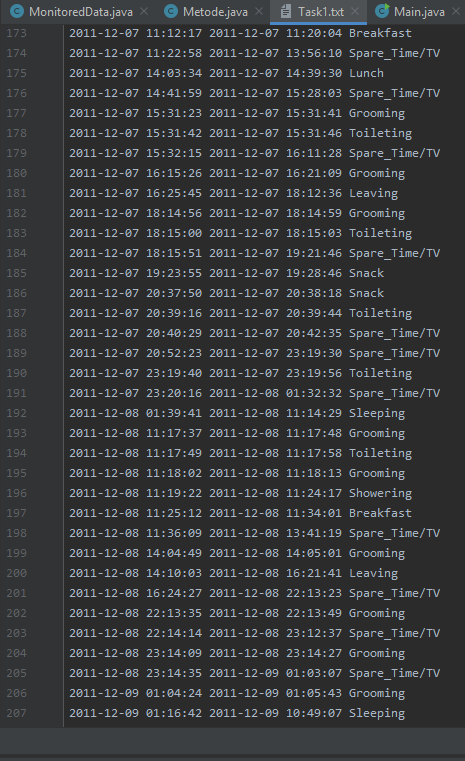
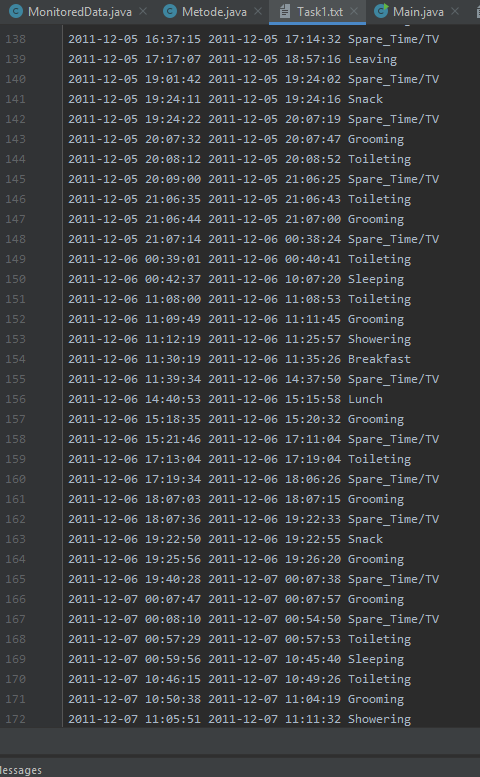


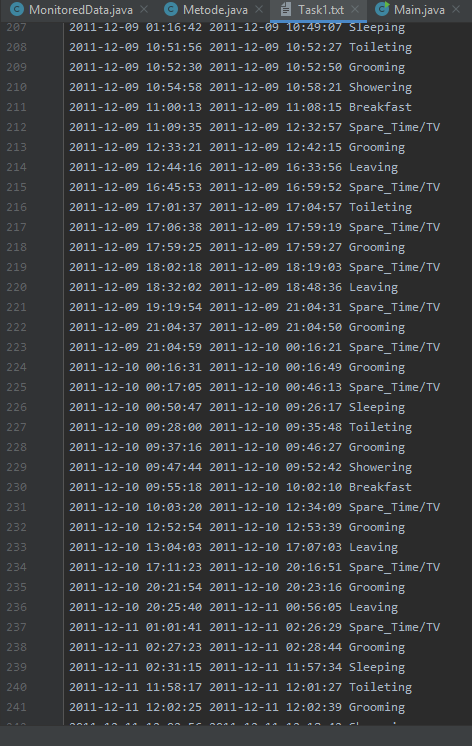
Rezultatele pentru fiecare task sunt aratate mai jos.

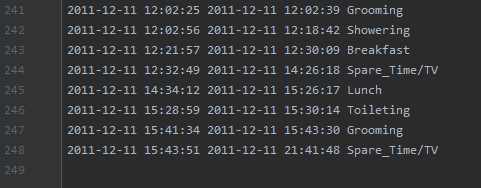
Task1



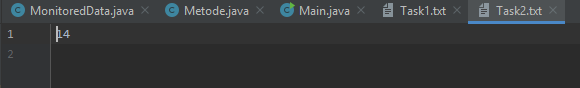




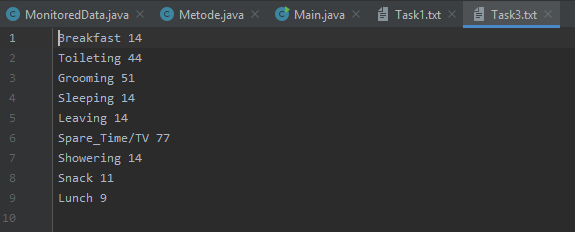




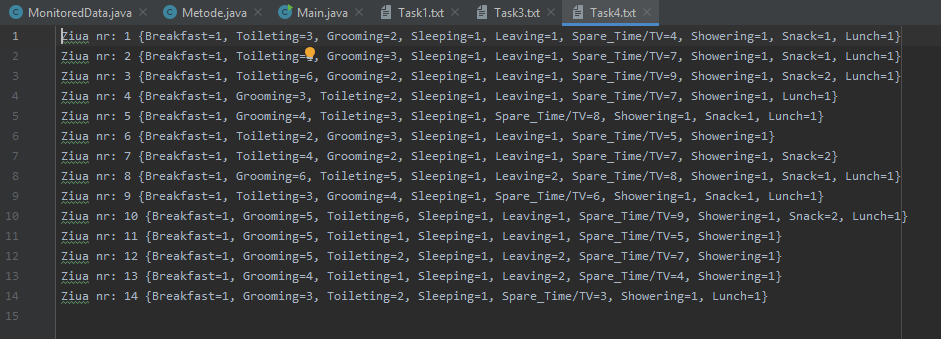
Task2



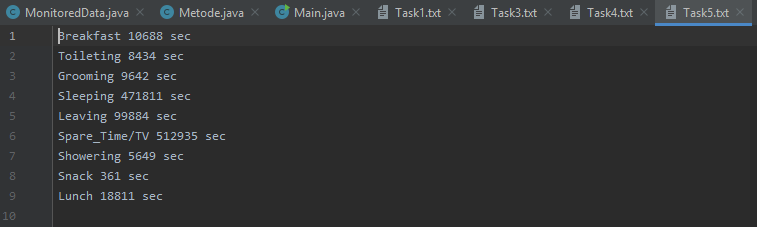
Task3



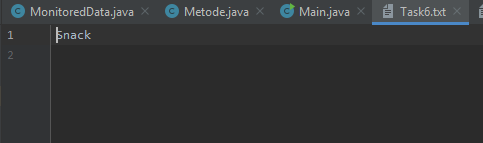
Task4



Task5



Task6



7 . Concluzie

In concluzie, am realizat proiectul dat cu toate cerintele cerute. Implementarea acestui proiect m-a invatat sa lucrez cu streamuri si cu lambda expressions.

8 . Bibliografie

M-am inspirat la folosirea lambda expression-urilor a streamurilor si a obiectelor de tip data:

<https://stackoverflow.com/>

<https://www.geeksforgeeks.org/>

<https://docs.oracle.com/en/>