

Scutaru Lucian

Gr. 30225

1. Obiectivul temei:

Se va prezenta obiectivul principal al temei printr-o fraza si un tabel sau o lista cu obiectivele secundare. Obiectivele secundare reprezinta pasii care trebuie urmati pentru indeplinirea obiectivului principal. Fiecare obiectiv secundar va fi descris si se va indica in care capitol al documentatiei va fi detaliat.

Obiectivul temei este implementarea si testare unei aplicatii pentru analizarea comportamentului unei personae, inregistrat cu ajutorul unor senzori instalati in propria casa. Rezultatul fiecarui task vor aparea intr-un fisier text.

Obiective secundare:

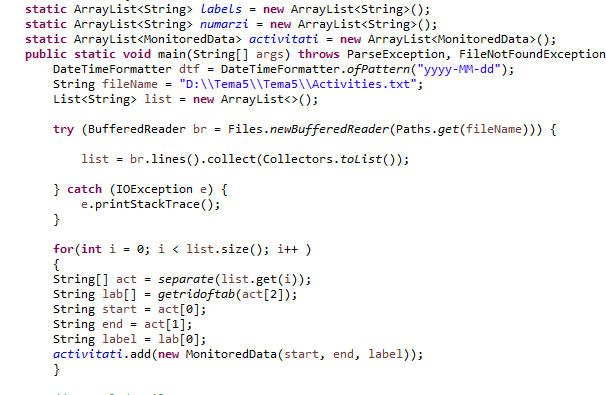
1. Implementarea clasei MonitoredData cu 3 field-uri, fiecare reprezentand momentul inceperii, finalul si numele activitatii. - 2
2. Citirea activitatilor din fisierul text si impartirea fiecarei linii in 3 parti. - 2
3. Numararea fiecarei zile din fisierul text. - 4
4. Numararea si adaugarera intr-un hashmap a activitatilor care apar si a numarului de aparitii ale acestora. - 4
5. Numararea duratei cumulate a fiecarei activitati. - 4
6. Adaugarea rezultatelor in fisiere text. – 4
7. Analiza problemei, modelare, scenarii, cazuri de utilizare

Se va prezenta cadrul de cerinte functionale formalizat si cazurile de utilizare ca si diagrame si descrieri de use-case. Descrierile use-case-urilor se vor face sub forma unui flow-chart ori sub forma unei liste continand pasii executiei fiecarui use-case.

Am decis sa abordez problema astfel:

Am citit din fisierul text cu ajutorul unui stream si am impartit fiecare linie citita in 3 field-uri(start, stop, label). Dupa citirea din text am creat o clasa care defineste fiecare activitate si am creat obiectele cu ajutorul detaliilor citite din fisierul text apoi le-am adaugat intr-un arraylist.

Task-urile 2 3 si 5 le voi detalia la capitolul 4.



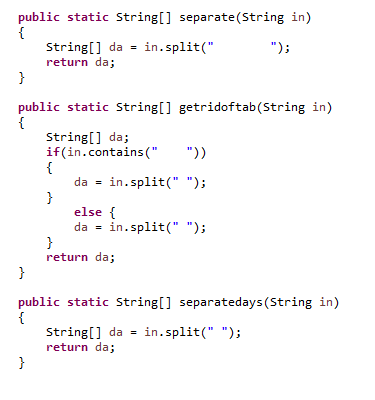
1. Proiectare (decizii de proiectare, diagrame UML, structuri de date, proiectare clase, interfete, relatii, packages, algoritmi, interfata utilizator)

Se va prezenta proiectarea OOP a aplicatiei, diagramele UML de clase si de pachete, structurile de date folosite, interfetele definite si algoritmii folositi (daca e cazul)

Structurile de date folosite sunt ArrayList-uri si HashMap-uri. ArrayList-ul este folosit pentru a salva activitatile(obiectele) pe care le creez dupa ce citesc din fisierul text. Am ales sa folosesc ArrayList pentru a putea insera in fisierul text ulterior mai multe evenimente.

HashMap-urile le folosesc pentru a implementa task-urile 3 si 5.

Am implementat 3 metode insa in esenta acestea sunt aceleasi, singurul lucru care difera este o singura conditie. Metodele sunt folosite pentru a structura/imparti fiecare linie citita din fisieul text in 3 campuri diferite(cu ajutorul .split).



1. Implementare

Se va descrie fiecare clasa cu campurile si cu metodele importante. Se va descrie implemantarea interfetei utilizator.

Singura clasa pe care am folosit-o este cea care defineste activitatile. Aici am implementat un simpu constructor si setters si getters.

In clasa monitored data am 3 field-uri, fiecare reprezentate de cate un String, datile urmand sa fie transformate in obiecte Date/FateTime.

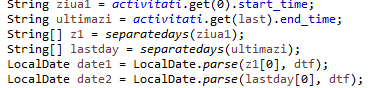
In clasa Test(cea cu main-ul) am implementat cele 3 metode reprezentate mai sus si structurile de date de care am avut nevoie(pe langa clasa main).

Abordare task-uri:

Pentru Task2 am calculat initital un int care imi returneaza numarul ultimei zile din fisierul cu activitati.



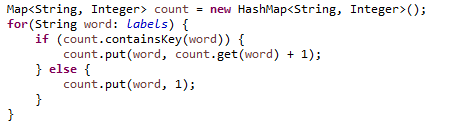
Am luat din ArrayList-ul cu activitati stringurile cu data de inceput a primei activitati si cea de final a ultimei activitati. Cu ajutorul unei metode prezentate mai sus am separat string-urile de mai sus in 2 campuri(data/ora) si am creart cate un obiect LocalDate pentru prima si ultima activitate.

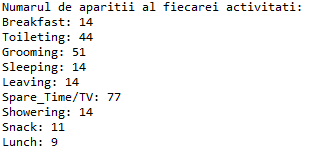


Cu ajutorul Duration.betwen am calculate un long care reprezinta numarul de zile dintre cele 2 date create mai sus. Am scris apoi in fisier cu ajutorul unui PrintWriter numarul zilelor.



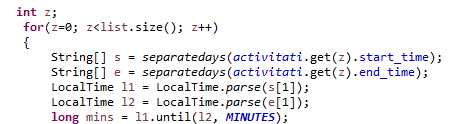
Pentru Task3 am creat un nou ArrayList de String-uri si am adaugat in el toate numele activitatilor luate din Arraylist-ul cu activitati. Am luat apoi un HashMap<String, Integer>, am parcurs ArrayList-ul cu numele activitatilor si am pus 2 conditii. Daca HashMap contine deja un nume de activitate, atunci Integer-ul asociat acesteia se incrementeaza, daca nu atunci label-ul este adaugat in Map iar Integerul este egal cu 1.



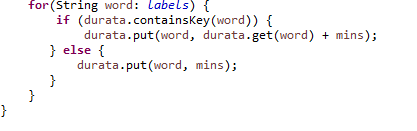


La Task5 am declarant un HashMap<String, Long> in care voi adauga mai apoi numele fiecarei activitati si numarul minutelor cumulate pentru fiecare din acestea.

Am parcurs tot ArrayList-ul cu ajutorul unui for. Pentru fiecare activitate am luat cate un string cu ora de incepere si cea de terminare. Am creat cate un obiect LocalDate cu cele 2 dati (ore) iar apoi am calculate un long cu numarul minutelor dintre cele 2 dati cu ajutorul .until.



Asemanator cu Task3, am adaugat in HashMap tipul de activitate si numarul de minute cumulate ale acestora.

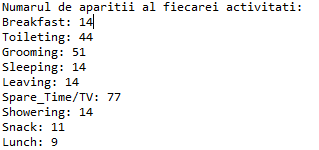


1. Rezultate

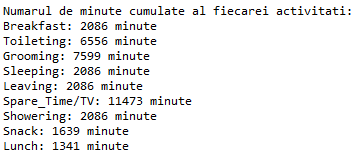
Rezultate Task 2:



Rezultate Task 3:



Rezultate Task 5:



1. Concluzii

In concluzie am reusit sa implementez 4 task-uri din cele 6 prezentate in Tema5, cu ajutorul unor structure de date adecvate si a metodelor ce manipuleaza liniile citite din fisierul text.

Din tema 5 am invatat lucrul cu stream si split, lucrul cu HashMap, citirea si manipularea liniilor citite din fisier text, scrierea in fisier text.

1. Bibliografie
2. <https://stackoverflow.com/>
3. <https://www.reddit.com/>
4. <https://www.w3schools.com/>
5. <https://www.callicoder.com/>