Tehnici de programare

Documentatie tema 5

Miklós Balázs

Grupa 30224

UTCN, Calculatoare

Monitoring a person’s behavour

1. Obiectivul temei

Programul este folosit pentru a prelucra datele dint-un fisier txt. In fisierul apar activitatile unei persoane. Prima chestie este start time, dupa aia end time si in final ActivityLabel. Sarcina este de a citi datele din fisier, a separa pe ele si a calcula niste detalii in legatura cu ele.

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

Cerinta problemei este de a folosi stremuri penru a citi datele din fisier, si a folosi lambda expressions pentru a calcula niste date in plus in legatura cu datele citie mai anterior.

Pentru a separa cele trei parti importante a unei linii din fisier, am luat o clasa numit MonitoredData, unde stochez timpul initial, timpul final si numele activitatii in cate un string. Dupa ce am luat o lista de MonitoreData ca ArrayList pentru a stoca date pentru fiecare linii.

Impartirea unei linii am facut cu functia split, care este foarte utila daca este vorba despre impartirea unui String la mai multe subStringuri, care sunt separate cu acelasi segmenta de caractere. Dupa aceasta, m-am dat seama, ca tot cu functia split pot sa fac impartirea datei si orei in fiecare begin si end time a sirului impartit mai anterior. In Datele, anul, ziua si ora a fost impartita cu caracterul „-” iar ora, minutul si secondele au fost impartite cu „:”.

Am creat o noua clasa DetailedMonitoredData unde am stochat toata datele impartite anterior in cate un int, ca sa pot sa folosesc in cerinta care zice sa calculam durata fiecarei activitate in mai multe feluri.

Pentru toate calculele importante, ce se afla in cerinta, am folosit lambda expressions. Pentru a folosi lambda expressions trebuie sa folosim java 1.8 si am declaarat o interfata separata pentru fiecare dintre functiile. (antetul expresiilor lambda). In interfata este definit anetul functiei, adica ce parametrii ar trebui sa primeasca si tipul returnat.

In continuare am scris fucntii pentru a calcula numarul total de zile, cate ori a aparut fiecare activitate, cate ori a aparturt fiecare acivitate in fiecare zi, cat timp a durat fiecare activitate pe linie, cat timp a durat fiecare activitate in total etc.

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

Programul contine doua pachete, una pentru interfete si una pentru clase, fiindca nu a fost cerut nici interfata utilizator de grafica (GUI), nici nimic dupa care am simtit ca trebuie sa folosesc mai multe pacheturi inutile.

Pentru a calcula datele cerintei, am folosit niste algoritmi dupa capul meu, ce o sa explic in continuare.

Pentru a calcula numarul de zile in toata perioada de monitoring, am initializat cu 0 o variabila cnt, am parcursi luniile cu un for, in prima luna am scazut numarul de zile din numarul total de zile a lunii, dupa aia am adaugat pur si simplu numarul de zile in fiecare luna la variabila cnt.

Pentru a calcula de cate odata apare fiecare activitate in toata perioada,am luat niste variabile initializat la 0 in functie de numele activitatii, am parcurs toata activitatiile si am incrementat fiecare variabila cu unu, in functie de activitate. Am returnat un map de String si integer unde stochez datele calculate.

Pentru a calcula durata activitaii in fiecare linie, de fapt am scazut timpul initial din timpul final, sigur ca nu a fost asa de simplu scaderea. Daca ora initiala a fost mai mic decat ora finala pur si simplu am scazut din final initialul, invers am scazut din 24 ora initiala si am adaugat la ora finala. Si tot asa au iesit si minutele si secundele pentru fiecare linie. In final rezultatul am calculat in secunde. Am tranformate si minutele si orele in secunde si am returnat un sir de numere intregi cu secundele calculate pentru fiecare activitate.

Pentru a calcula durata intreaga a activitatiilor in toata monitoring time, am parcurs sirul parcurs anterior si in functie de ce activitate sunt ele, am adaugat la un alt sir de activitati cu numari intregi care au fost initializat la 0.

Pentru a calcula de cate ori a aparut fiecare activitate intr-o zi, am luat un sir de ActivityCountForEachDay cu dimensiunea nrDays. Am initializat variabila day cu 28-30 respectiv de la 1-11. Dupa ce am parcurs lista de DetailedMonitoredData si sirul de zile. Am zis daca ziua de inceput sau ziua de final este egala cu variabila day de la sirul de ActivityCountForEachDay, atunci incrementez cu 1 activitatea curenta de la ziua respectiva. In final am returnat sieul de ActivityCountForEachDay.

Pentru a filtra activitatile dintre care 90% s-a terminat mai putin decat 5 minute, am luat un sir cu numele activitatilor, si am parcurs. Am initializat un counter la 0 si la Map-ul de activitati am gasit numarul care arata de cate ori a fost facut activitatea in total. Dupa ce am parcurs cu un for durata fiecarei activitate pe linie. Daca numele activitatii a fost egala cu numele activitatii curent din sir, si daca durata ei a fost mai putin decat 300 de secunde atunci am incrementat counterul. In afara forului, in final daca counter a fost mai mare decat 9/10*deCateOriAAparutActivitateaInTotal atunci am adaugat numele lui la lista pe care returnez in final.

4. Implementare

Pachtelul interfaces contine urmatoare interfete:

Interfata **ActivityCountForEachDay** este folosita pentru a calcula numarul de ativitati in fiecare zi . Clasa are urmatoarele metode:

-publicActivitiesEachDay[]countActivitiesForEachDay(ArrayList<DetailedMonitoredData>datas,

int nrOfDays)- returneaza un sir de ActivitiesEachDay. ActivitiesEachDay este o clasa in care este stocat numarul zilei si numarul activitatilor facute in ziua respectiva.

Interfata **CountActivities** este folosita pentru a calcula de cate ori a fost facuta fiecare activitate in total. Are metoda:

-public Map<String,Integer> countActivityApparance(ArrayList<MonitoredData> datas)- returnneaza un Map unde cheia este numele activitatii la care este asociat un numar intreg, de cate ri a fost facut activitatea respectiva.

Interfata **Operations** contine metoda

-public int countDays(ArrayList<DetailedMonitoredData> datas)- care returneaza un int, numarul total a zielelor in intreaga durata de activitatii.

Interfata **RecordDurationForEachLine** contine metoda:

-public int[] calculateDuration(ArrayList<DetailedMonitoredData>datas)- calculeaza durata activitatiior in fiecare linie, si returneaza un sir de numar intregi cu timpurile calculate.

Interfata **RecordDurationOfEachActivity** contine metoda:

-public int[] durationOfActivities(int[] durations, ArrayList<MonitoredData> datas)- returneaza un sir de intregi, unde este stocat ca fiecare acivitate cat timp a durat in total.

Interfata **LastTask_90** contine metada:

-public ArrayList<String> Less_5_mins_90(Map<String, Integer> activityApparance, int[] durations,String[] names, ArrayList<MonitoredData>lines)- metoda filtreaza activitatile dintre care 90% s-a terminat mai putin decat 5 minute. Returneaza un ArrayList din numele activitatilor care satisfac conditia.

Pachetul model este folosit pentru a crea Clasele necesare folosite pentru prelucrarea datelor citite din fisier.

Clasa **ActivitiesEachDay** are urmatoarele variabile de instanta: int day, int month, int leaving=0,toiletting = 0, showering=0, sleeping=0, breakfast=0, launch=0, dinner=0, snack=0, Spare_TimeTV=0, grooming=0. Clasa este folosita pentru a calcula de cate ori a fost facuta fiecare activtate in fiecare zi. Are urmatoare metode:

-public ActivitiesEachDay(int month, int day)- constructorul clasei.

-getters si setters pentru fiecare variabile de instanta

Clasa **DetailedMonitoredData** are urmatoarele variabile de instanta: int beginYear, int beginMonth, int beginDay, int beginHour, int beginMins, int beginSecs, int endYear, int endMonth, int endDay, int endHour, endMins, endSecs, String activityName. Clasa este folosita pentru a stoca fiecare date importante pe fiecare linie. Are urmatoare metode:

-public DetailedMonitoredData(int beginYear, int beginMonth, int beginDay, int beginHour, int beginMins, int beginSecs,int endYear, int endMonth, int endDay, int endHour,int endMins, int endSecs, String activityName)- constructorul.
-getters si setters

Clasa **FileReader** este creat pentru a citi datele din fisierul Activities.txt. Clasa are o singura metoda statica pentru citire:
-public static List<String>readFile() throws IOException,ClassNotFoundException- returneaza o lista de Stringuri despre datele citite din fisier.

Clasa **MonitoredData** este clasa care a aparut si in cerinta, pentru a separa datele unei linii in trei bucati. Are urmatoarele variabile de instanta: String startTime, String endTime, String activityLabel. Clasa are urmatoarele metode:
-public MonitoredData(String startTime, String endTime, String activityLabel)- constructorul
-getters si setters

Clasa **Simulator** este folosita pentru a testa toata metodele enumerate anterior. In el este apelat citirea datelor din fisier si prelucrarea datelor tot. Clasa are urmatoarele metode importante:
-public Simulator()- constructorul, aici inseamna ca apelez metodele pentru a afisa rezultatele in consol.
-public int DayNrOfMonth(int monthNR)-returneaza 31,30 sau 28, este o metoda care calculeaza ca o luna cate zile are, in functie de numarul lunii.
-public ArrayList<MonitoredData> createAListOfMonitoredData(List<String> content)- prelucreaza o Lista de Stirnguri, citi de la fisier la un ArrayList de MonitoredData cu start time, end time si activity name.
-publicArrayList<DetailedMonitoredData>createDatasMoreDetailed(ArrayList<MonitoredData> datas)- creaza un ArrrayList de DetailedMonitoredData dintr-un ArrayList de MonitoredData. Aceasta inseamna ca aici sunt salvate fiecare detalii si date ale unei linii de activitate.
-Operations countDays=(ArrayList<DetailedMonitoredData> datas)->{return int}- este lambda expression care numara cate zile exista in total, returneaza un int.
-CountAtivities countEachActivity=(ArrayList<MonitoredData> datas)->{}- metoda care calculeaza de cate ori a aparut fiecare activitate in intreaga timp, returneaza un Map de String si int.
ActivityCountForEachDay countActivitiesforDays=(ArrayList<DetailedMonitoredData>datas,int nrDays)-> lambda expression care calculeaza aparenta fiecarui activitate in fiecare zi.
-RecordDurationForEachLine recordDurationForEachLine=(ArrayList<DetailedMonitoredData>datas)->{}- lambda expression care lalculeaza durata fiecarui activitate pe linie in secunde.
-RecordDurationOfEachActivity actRec=(int [] durations,ArrayList<MonitoredData>datas)->{}- lambda expression care calculeaza durata fiecarui activitate in total pe parcursul masurarii.
-LastTask_90 task=(Map<String, Integer> activityApparance, int[] durations,String[]names, ArrayList<MonitoredData> lines)->{}- lambda expression care filtreaza activitatile dintre care 90% au terminat mai putin decat 5 minute.

5. Rezultate

Rezultatele programului pot fi vazute in conzol, testarile pot fi vazute dupa afisarea rezultatelor in conzol. In clasa Simultator sunt implementate functiile care calculeaza diferite sarcine ale cerintei.

Rezultatele sunt corecte, niste rezultate au fost calculate si pe foaie, de aceea pot sa afirm ca functiile si expresile lambda sunt implementat in mod corect.

6. Concluzii:

Aplicatiile de genu ceea ce este cerut in tema aceasta pot fi utilizate in orice aplicatie care terebuie sa lucreaza cu datele unui fisier. Java este un limbaj de programare ideala pentdu dezovltarea aplicatiilor ca si aceasta, exista multe functii pentru a lucra cu Stringuri si pentru a procesa Stringurile, de aceea este folositoara si la lucrarea cu fisiere.

7. Bibliografie

file:///C:/Users/Balazs-PC/Downloads/HW5_Indications%20(1).pdf

8. Diagrama UML: