package kursach;

import java.io.BufferedReader;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import static java.lang.System.in;

import java.util.ArrayList;

import java.util.Scanner;

public class Kursach {

static mainBank koyash1;

public static void main(String[] args) {

NewJFrame frame = new NewJFrame();

frame.setVisible(true);

}

static class mainBank { //объект банк

private klient[] bank; //кольцевая очередь для объединения клиентов на основе массива

private int first, last; //указатели на клиентов первого и последнего в очереди

private int maxClients; //ограничение на количество клиентов

private int count = 0;//счётчик клиентов

public void mainBank(int maxCount){

//конструктор

first = 0;

last = 0;

maxClients = maxCount;

bank = new klient[maxClients];

}

public int getMaxClients(){

return maxClients;

}

public int getCount(){

return count;

}

klient[] getBank(){

return bank;

}

public int getFirst(){

return first;

}

public int getLast(){

return last;

}

public boolean addClient(String newSur){

//фун-ция для добавления клиентов

if (count < maxClients) {

if (!checkForSur(newSur)){

return false;

}

count++;

bank[last] = new klient();

bank[last].klient(newSur);

last++;

if (last >= maxClients)

last = 0;

return true;

} else {

return false;

}

}

public boolean checkForSur(String checkSur){

if (count > 0){

int i = first;

int j = 0;

while (j < count){

if (bank[i].getSurname().equals(checkSur)){

return false;

}

i++;

j++;

if (i >= maxClients)

i = 0;

}

return true;

} return true;

}

public klient getKlient(String newSur){

//фун-кция для получение клиента

int i = first;

int j = 0;

while (j < count){

if (bank[i].getSurname().equals(newSur)){

return bank[i];

}

i++;

j++;

if (i >= maxClients)

i = 0;

}

return null;

}

public boolean deleteFirstClient() {

//удаление первого

if (count > 0) {

bank[first] = null;

first++;

if (first == maxClients)

first = 0;

count--;

return true;

}else{

return false;

}

}

public String[][] getInfo(klient tempInfo) {

//получение информации о клиенте

String[][] result;

result = tempInfo.getInfo(tempInfo.getCountOp());

return result;

}

public boolean loadFromFile(String path) throws FileNotFoundException, ParseException{

//загрузка из файла всей структуры

BufferedReader br = new BufferedReader(new InputStreamReader(new FileInputStream(path)));

String n;

try {

if (null != (n = br.readLine())){

koyash1 = new mainBank();

koyash1.mainBank(Integer.valueOf(n));

for (int i = 0; i < Integer.valueOf(n); i++) {

String name = br.readLine();

koyash1.addClient(name);

String countOfOp = br.readLine();

for (int j = 0; j < Integer.valueOf(countOfOp); j++) {

String dateFromfile = br.readLine();

String opFromFile = br.readLine();

klient loadClientTemp = koyash1.getKlient(name);

loadClientTemp.setHistory (dateFromfile, Integer.valueOf(opFromFile));

}

}

br.close();

return true;

} else {

br.close();

return false;

}

} catch (IOException e){

return false;

}

}

public void saveToFile(FileWriter sFout) {

//сохранение в файл всей стркутуры

try {

sFout.write(String.valueOf(count) + "\n");

for (int i = 0; i < count; i++) {

sFout.write(bank[i].getSurname() + "\n" + bank[i].getCountOp() + "\n");

String[][] info = bank[i].getInfo(bank[i].getCountOp());

if (!info.equals("") && info != (null))

for (int j = 0; j < info.length;j++){

sFout.write(info[j][0] + "\n" + info[j][1] + "\n");

}

}

sFout.close();

} catch (IOException e){

System.out.println("Some error!");

}

}

}

static public class klient {

private String surname;

private int budget, countOp = 0;

private operation history; //двунаправленный список без заголовка для операций (а это первый элемент)

public void klient(String sur){

//конструктор для создания нового клиента

surname = sur;

budget = 0;

}

public void setNewSurname(String newSur){

//функция которая устанавливает новую фамилию

surname = newSur;

}

public boolean setHistory(String newData, int newOp) throws ParseException {

//функция для добавления действий в список history(дата и +число или -число) а так же изменения значения бюджета

if (newOp < 0 && Math.abs(newOp) > budget) { //если на счету не достаточно денег при снятии

return false;

} else {

budget += newOp; //изменение бюджета

if (history == null) { //если история операций ещё не была создана

history = new operation(); //то выделяем место для новой переменной

history.operation(history, newData, newOp); //и строим(через конструктор) её

} else {//если же история уже была создана

int i = 0; //вспомогательный счётчик для поиска места

operation curr = history; //вспомогательная переменная для поиска куда ставить новое значение

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy, HH:mm");

Date dateCurr = null, dateTemp = null;

dateCurr = sdf.parse(curr.getData());

dateTemp = sdf.parse(newData);

while (i < countOp - 1 && dateCurr.compareTo(dateTemp) == -1){ //пока мы не пробежались по всем элементам И не нашли бОльший элемент

i++;//идём дальше

curr = curr.getNext();

dateCurr = sdf.parse(curr.getData());

}

operation temp = new operation();//когда мы нашли ПРИМЕРНОЕ подходяещее место создаем вспомгательную переменную

temp.operation(temp, newData, newOp); //определяем её

switch (dateCurr.compareTo(dateTemp)) {

case 0: //если даты равны

while (i < countOp - 1 && curr.getOper() < newOp && dateCurr.compareTo(dateTemp) == 0){ //пока мы не пробежались по всем элементам И не нашли бОльший элемент

i++;//идём дальше

curr = curr.getNext();

dateCurr = sdf.parse(curr.getData());

}

if (dateCurr.compareTo(dateTemp) == 0){ //если мы не вышли за диапазон

if (curr.getOper() >= newOp){

temp.setNext(curr);

temp.setPrev(curr.getPrev());

temp.getPrev().setNext(temp);

curr.setPrev(temp);

} else {

temp.setNext(curr.getNext());

temp.setPrev(curr);

temp.getNext().setPrev(temp);

curr.setNext(temp);

}

if (temp.getOper() < history.getOper() && i == 0)

history = temp;

} else { //если всё таки вышли

curr = curr.getPrev();

temp.setNext(curr.getNext());

temp.setPrev(curr);

temp.getNext().setPrev(temp);

curr.setNext(temp);

}

break;

case 1: //если новая дата меньше

temp.setNext(curr);

temp.setPrev(curr.getPrev());

temp.getPrev().setNext(temp);

curr.setPrev(temp);

if (i == 0)

history = temp;

break;

default: //а иначе

temp.setNext(curr.getNext());

temp.setPrev(curr);

temp.getNext().setPrev(temp);

curr.setNext(temp);

break;

}

}

countOp++; //увеличиваем счётчик

return true;

}

}

public operation findOp(String findData, int findOp){

operation curr = history;

int i = 0;

while (i < countOp) {

if (curr.getData().equals(findData) && curr.getOper() == findOp) {

return curr;

}

curr = curr.getNext();

i++;

}

return null;

}

public boolean deleteHistory(String deletingData, int deletingOp){

//функция для удаление операции

if (history != null) {

operation findDel = findOp(deletingData, deletingOp);

if (findDel != null){

if (countOp == 1) {

history = null;

budget -= deletingOp;

history = null;

countOp--;

return true;

}

if (history == findDel)

history = findDel.getNext();

findDel.getPrev().setNext(findDel.getNext());

findDel.getNext().setPrev(findDel.getPrev());

budget -= deletingOp;

countOp--;

return true;

}

} return false;

}

public int getBudget(){

return budget;

}

public String getSurname(){

return surname;

}

public int getCountOp(){return countOp;}

public String[][] getInfo(int size) {

//возвращает всю инфу про клиента (все его операции и даты)

String[][] result = new String[size][2];

if (history != null) {

operation tempInfo = history;

result[0][0] = tempInfo.getData();

result[0][1] = String.valueOf(tempInfo.getOper());

tempInfo = tempInfo.getNext();

int j = 1;

while (j < countOp) {

result[j][0] = tempInfo.getData();

result[j][1] = String.valueOf(tempInfo.getOper());

j++;

tempInfo = tempInfo.getNext();

}

}

return result;

}

}

static public class operation {

private String data; //дата

private int oper; //операция

private operation next, prev; //указатель на следующую и предыдущую операцию

public void operation(operation first, String newData, int newOp) {

//конструктор для создания новой операции

data = newData;

oper = newOp;

next = first;

prev = first;

}

public operation getNext() {

return next;

}

public void setNext(operation newNext) {

next = newNext;

}

public void setPrev(operation newPrev) {

prev = newPrev;

}

public void setOper(int newOp) {

oper = newOp;

}

public void setData(String newData) {

data = newData;

}

public String getData() {

return data;

}

public int getOper() {

return oper;

}

public operation getPrev() {

return prev;

}

}

}