****

**Data Structures project**

**Nagham Thamer Alotaibi**

**202110560**

**Dr. Abu Arqoub**

**Second semester 2023-2024**

**Table of contents:**

|  |
| --- |
| File data.txt |
| Customer class |
| VInterface |
| Vlist (implements) |
| StackInterface |
| Vstack (implements) |
| QInterface |
| LQueue (implements) |
| Project (main) |

**Problem definition:**

**A text file containing records about customers who applied to Arab Bank for a Mastercard-Prepaid subscription.**

**Brief information:**

**I used**

**stack for date of birth “ vector”**

**Queue for city “linked list”**

**List for all customer information “vector”**

**Customer class**

**package project;**

**public class customer {**

**String Fname;**

**String Lastname;**

**int Id;**

**int Salary;**

**int Tel;**

**String City;**

**int DateOfBirth;**

**private boolean servedFlag;**

**customer(){**

**}**

**customer(String Fname,String Lastname,int Id,int Salary,int Tel,String City,int DateOfBirth){**

**this.Fname=Fname;**

**this.Lastname=Lastname;**

**this.Id=Id;**

**this.Salary=Salary;**

**this.Tel=Tel;**

**this.City=City;**

**this.DateOfBirth=DateOfBirth;**

**}**

**public boolean servedFlag() {**

**return false; }**

**public void setservedFlag(boolean servedFlag) {**

**this.servedFlag = servedFlag;**

**}**

**public String toSring(){**

**return "customer:"+"Fname:"+Fname+"Lastname:"+Lastname+"Id:"+Id+"Salary:"+Salary+"Tel:"+Tel+"City:"+City+"DateOfBirth:"+DateOfBirth+"servedFlag:"+servedFlag; }**

**}**

**File: data.txt**

**Nagham,Alotaibi,560,3000,0504225529,Amman,2003**

**Haneen,Thamer,533,200,0789773552,Amman,2005**

**Balsam,Emad,432,143,0786611275,irbid,1999**

**Ahmad,Ali,123,834,0791954530,irbid,1887**

**jameel,jamal,566,300,0788080939,zarqa,1997**

**Hala,Lutfi,834,600,0503492997,zarqa,1987**

**Mohammad,adnan,187,90,9678754320,Amman,1992**

**StackInterface**

**package project;**

**public interface StackInterface {**

**/\*\***

**\* push an object into the stack**

**\* @param x as the object**

**\*/**

**public void push(Object x);**

**/\*\***

**\* remove the last element from the stack and returns it**

**\* @return Object as the last element from the stack**

**\*/**

**public Object pop();**

**/\*\***

**\* returns the last element without removing it**

**\* @return Object**

**\*/**

**public Object peek();**

**/\*\***

**\* remove every element from the stack**

**\*/**

**public void clear();**

**/\*\***

**\* checks wether the stack is empty or not**

**\* @return true if the stack is empty false otherwise**

**\*/**

**public boolean isEmpty();**

**}**

**Vstack**

**package project;**

**import java.util.Vector;**

**public class Vstack implements StackInterface{**

**private Vector vstk;**

**Vstack(){**

**vstk=new Vector();**

**}**

**Vstack(int v\_size){**

**vstk=new Vector(v\_size);**

**}**

**public boolean isEmpty(){**

**return vstk.isEmpty();**

**}**

**public void clear(){**

**vstk.removeAllElements();**

**}**

**public void push(Object x){**

**vstk.addElement(x);**

**}**

**public Object pop(){**

**Object R=null;**

**if(!isEmpty()){**

**R=vstk.elementAt(vstk.size()-1);**

**vstk.removeElement(vstk.size()-1);**

**}**

**return R; }**

**public Object peek(){**

**Object R=null;**

**if(!isEmpty()){**

**R=vstk.elementAt(vstk.size()-1);**

**}**

**return R; }**

**}**

**VInterface**

**package project;**

**public interface VInterface {**

**public void sort(Vlist myList);**

**/\*\***

**\* add an object at the end of the list**

**\* @param newEntry as an object**

**\* @return true if the process is complete**

**\*/**

**public boolean add(Object newEntry);**

**/\*\***

**\* add an object at a specified location on the list**

**\* @param newEntry as an Object**

**\* @param givenPosition as the location on the list**

**\* @return true if the process was successful**

**\*/**

**public boolean add(int givenPosition,Object newEntry);**

**/\*\***

**\* remove an object from the list**

**\* @param givenPosition the position of the object to be removed**

**\* @return the removed object**

**\*/**

**public Object remove(int givenPosition);**

**/\*\***

**\* display all the elements in the list**

**\*/**

**public void display();**

**/\*\***

**\* search for an Object in the list**

**\* @param anEntry the object to be searched**

**\* @return true if the object was found**

**\*/**

**public boolean contains(Object anEntry);**

**/\*\***

**\* checks wether the list is full**

**\* @return true if the list is full**

**\*/**

**public boolean isFull();**

**/\*\***

**\* checks wether the list is empty**

**\* @return true if the list is empty**

**\*/**

**public boolean isEmpty();**

**/\*\***

**\* return the size of the list**

**\* @return the size of the list**

**\*/**

**public int getLength();**

**/\*\***

**\* remove every elements in the list**

**\*/**

**public void clear();**

**/\*\***

**\* replace an existing element in the list with another**

**\* @param newEntry the new Object**

**\* @param givenPosition the position of the Object to be replaced**

**\* @return true if the process was successful**

**\*/**

**public boolean replace(int givenPosition,Object newEntry);**

**/\*\***

**\* returns the element at the specified position**

**\* @param givenPosition as the position between 0 and length**

**\* @return Object as the element at the specified position**

**\*/**

**public Object getElementAt(int givenPosition);**

**/\*\***

**\* return the last element**

**\* @return Object as the last element**

**\*/**

**public Object getLastElement();**

**/\*\***

**\* return the first element**

**\* @return the first element**

**\*/**

**public Object getFirstElement();**

**}**

**Vlist**

**package project;**

**import java.util.Vector;**

**public class Vlist implements VInterface{**

**private Vector entry;**

**Vlist(){**

**entry=new Vector();**

**}**

**Vlist(int vSize){**

**entry=new Vector(vSize);**

**}**

**@Override**

**public boolean isFull(){**

**return false;**

**}**

**@Override**

**public boolean isEmpty(){**

**return entry.isEmpty();**

**}**

**@Override**

**public void clear(){**

**entry.removeAllElements();**

**}**

**@Override**

**public void display(){**

**display\_recursion(0);**

**}**

**private void display\_recursion(int start){**

**System.out.println(entry.elementAt(start));**

**if(start<entry.size()){**

**display\_recursion(start+1);**

**}**

**}**

**@Override**

**public int getLength(){**

**return entry.size();**

**}**

**@Override**

**public boolean add(Object newEntry){**

**entry.addElement(newEntry);**

**return true;**

**}**

**@Override**

**public boolean add(int givenPosition,Object newEntry){**

**boolean R=true;**

**givenPosition=givenPosition-1;**

**if(givenPosition>=0 && givenPosition<=entry.size())**

**entry.insertElementAt(newEntry, givenPosition);**

**return R;**

**}**

**@Override**

**public Object remove (int givenPosition){**

**Object R=null;**

**givenPosition=givenPosition-1;**

**if(!isEmpty() && givenPosition>=0 && givenPosition<entry.size()){**

**R=entry.elementAt(givenPosition);**

**entry.removeElementAt(givenPosition);**

**}**

**return R;**

**}**

**@Override**

**public boolean replace(int givenPosition,Object newEntry){**

**boolean R=true;**

**givenPosition=givenPosition-1;**

**if(!isEmpty() && givenPosition>=0 && givenPosition<entry.size()){**

**entry.setElementAt(newEntry, givenPosition);**

**}**

**return R;**

**}**

**@Override**

**public boolean contains(Object anEntry){**

**return entry.contains(anEntry);**

**}**

**@Override**

**public Object getElementAt(int givenPosition){**

**Object R=null;**

**givenPosition=givenPosition-1;**

**if(!isEmpty() && givenPosition>=0 && givenPosition<entry.size()){**

**R=entry.elementAt(givenPosition);**

**}**

**return R;**

**}**

**@Override**

**public Object getLastElement(){**

**try{**

**return entry.lastElement();**

**}**

**catch(java.util.NoSuchElementException e){**

**return null;**

**}**

**}**

**@Override**

**public Object getFirstElement(){**

**try{**

**return entry.firstElement();**

**}**

**catch(java.util.NoSuchElementException e){**

**return null;**

**}**

**}**

**@Override**

**public void sort(Vlist myList) {**

**Vlist amman=new Vlist();**

**Vlist zarqa=new Vlist();**

**Vlist madaba=new Vlist();**

**for (int i = 1; i <= myList.getLength(); i++) {**

**customer c1= (customer) myList.getElementAt(i);**

**if (c1.City.equalsIgnoreCase("Amman")) {**

**amman.add(c1);**

**} else if (c1.City.equalsIgnoreCase("irbid")) {**

**zarqa.add(c1);**

**}**

**else if (c1.City.equalsIgnoreCase("zarqa")) {**

**madaba.add(c1);**

**}**

**}**

**}**

**}**

**QInterface**

**package project;**

**public interface QInterface {**

**//add object**

**public void enqueue(Object newEntry);**

**//remove object and return the Front**

**public Object dequeue();**

**//return the Front**

**public Object getFront();**

**//is the queue empty?**

**public boolean isEmpty();**

**//removes all the queue**

**public void clear();**

**}**

**LQueue**

**package project;**

**public class LQueue implements QInterface{**

**private Node FrontNode;**

**private Node BackNode;**

**LQueue(){**

**clear();**

**}**

**@Override**

**public void clear(){**

**FrontNode=null;**

**BackNode=null;**

**}**

**@Override**

**public void enqueue(Object newEntry){**

**Node nn=new Node();**

**nn.data=newEntry;**

**if(isEmpty()){**

**FrontNode=BackNode=nn;**

**}**

**else{**

**BackNode.next=nn;**

**BackNode=nn;**

**}**

**}**

**@Override**

**public Object dequeue(){**

**Object R=null;**

**if(!isEmpty()){**

**R=FrontNode.data;**

**FrontNode=FrontNode.next;**

**}**

**if(FrontNode==null){**

**BackNode=null;**

**}**

**return R;**

**}**

**@Override**

**public Object getFront(){**

**Object R=null;**

**if(!isEmpty()){**

**R=FrontNode.data;**

**}**

**return R;**

**}**

**@Override**

**public boolean isEmpty(){**

**return FrontNode==null;**

**}**

**private class Node{**

**Object data;**

**Node next;**

**Node(){**

**data=null;**

**next=null;**

**}**

**Node(Object d){**

**data=d;**

**next=null;**

**}**

**Node(Object d,Node n){**

**data=d;**

**next=n;**

**}**

**}**

**}**

**Main**

**package project;**

**import java.util.Scanner;**

**import java.io.File;**

**import java.io.FileNotFoundException;**

**public class Project {**

**public static void main(String[] args) throws FileNotFoundException{**

**int money =0;**

**Vlist myList=new Vlist();**

**Vstack underAge=new Vstack();**

**File text=new File("C:/Users/Nagham/Downloads/pro\_DS/project/src/project/data.txt");//reading the file**

**Scanner lines=new Scanner(text);**

**while(lines.hasNextLine()) {**

**String line = lines.nextLine();**

**System.out.println(line);**

**LQueue nn = new LQueue();**

**String[] s1 = line.split(",");**

**for(int i=0; i < s1.length; i++) {**

**try {**

**nn.enqueue(Integer.parseInt(s1[i]));**

**}//end try**

**catch(NumberFormatException e){**

**nn.enqueue(s1[i]);**

**}//end catch**

**}**

**customer customer = new customer((String) nn.dequeue(), (String) nn.dequeue(), (int) nn.dequeue(), (int) nn.dequeue(), (int) nn.dequeue(), (String) nn.dequeue(), (int) nn.dequeue());**

**myList.add(customer);**

**}//end while**

**// to save the object in the list then arranging them using Queue for the lowest Date of Birth will be served first**

**Vlist Amman=new Vlist();**

**LQueue AmmanQ=new LQueue();**

**Vlist irbid=new Vlist();**

**LQueue irbidQ=new LQueue();**

**Vlist zarqa=new Vlist();**

**LQueue zarqaQ=new LQueue();**

**myList.sort(Amman);**

**myList.sort(irbid);**

**myList.sort(zarqa);**

**for(int i=1;i<=myList.getLength();i++){**

**customer c=(customer)myList.getElementAt(i);**

**if(c.City.equals("Amman")){**

**AmmanQ.enqueue(c);**

**}**

**else if(c.City.equals("irbid")){**

**irbidQ.enqueue(c);**

**}**

**else if(c.City.equals("zarqa")){**

**zarqaQ.enqueue(c);**

**}**

**}**

**//Vlist**

**/\* for(int i=0;i<myList.getLength();i++){**

**customer c=(customer)myList.getElementAt(i);**

**if(c.City.equals("Amman")){**

**Amman.add(c);**

**}**

**else if(c.City.equals("irbid")){**

**irbid.add(c);**

**}**

**else if(c.City.equals("zarqa")){**

**zarqa.add(c);**

**}**

**}\*/**

**myList.clear();**

**// this method will sorted by id**

**AmmanQ=sortQ(Amman);**

**irbidQ=sortQ(irbid);**

**zarqaQ=sortQ(zarqa);**

**// this method will service the customer by there age**

**MasterCardFirst(AmmanQ, Amman, myList, underAge);**

**MasterCardFirst(irbidQ, irbid, myList, underAge);**

**MasterCardFirst(zarqaQ, zarqa, myList, underAge);**

**myList.display();**

**// the total cost for all citys**

**money=22\*(Amman.getLength()+irbid.getLength()+zarqa.getLength());**

**System.out.println("the total money is "+ money);**

**}**

**public static void MasterCardFirst(LQueue L,Vlist v1,Vlist v2,Vstack stk){//They are arranged according to who will receive the service first, according to age**

**while(!L.isEmpty()){**

**customer c1=(customer)L.dequeue();**

**if(c1.DateOfBirth<=2003){**

**c1.setservedFlag(true);**

**v1.add(c1);**

**v2.add(c1);**

**}**

**else**

**stk.push(c1);//Those older than 2003 will wait for their turn with a stack**

**}**

**}**

**public static LQueue sortQ(Vlist source){//I arrange them according to the ID, and the less the ID, they will provide the service first**

**LQueue des = new LQueue();**

**int pos = 0;**

**while (!source.isEmpty()) {**

**for (int i = 0; i < source.getLength(); i++) {**

**customer temp = ((customer) source.getElementAt(i));**

**customer min = ((customer) source.getFirstElement());**

**if (temp.Id <= min.Id) {**

**min = temp;**

**pos = i;**

**}**

**}**

**des.enqueue((customer) source.remove(pos));// remove them after the service is done**

**}**

**return des;**

**}**

**}**

**Result**

