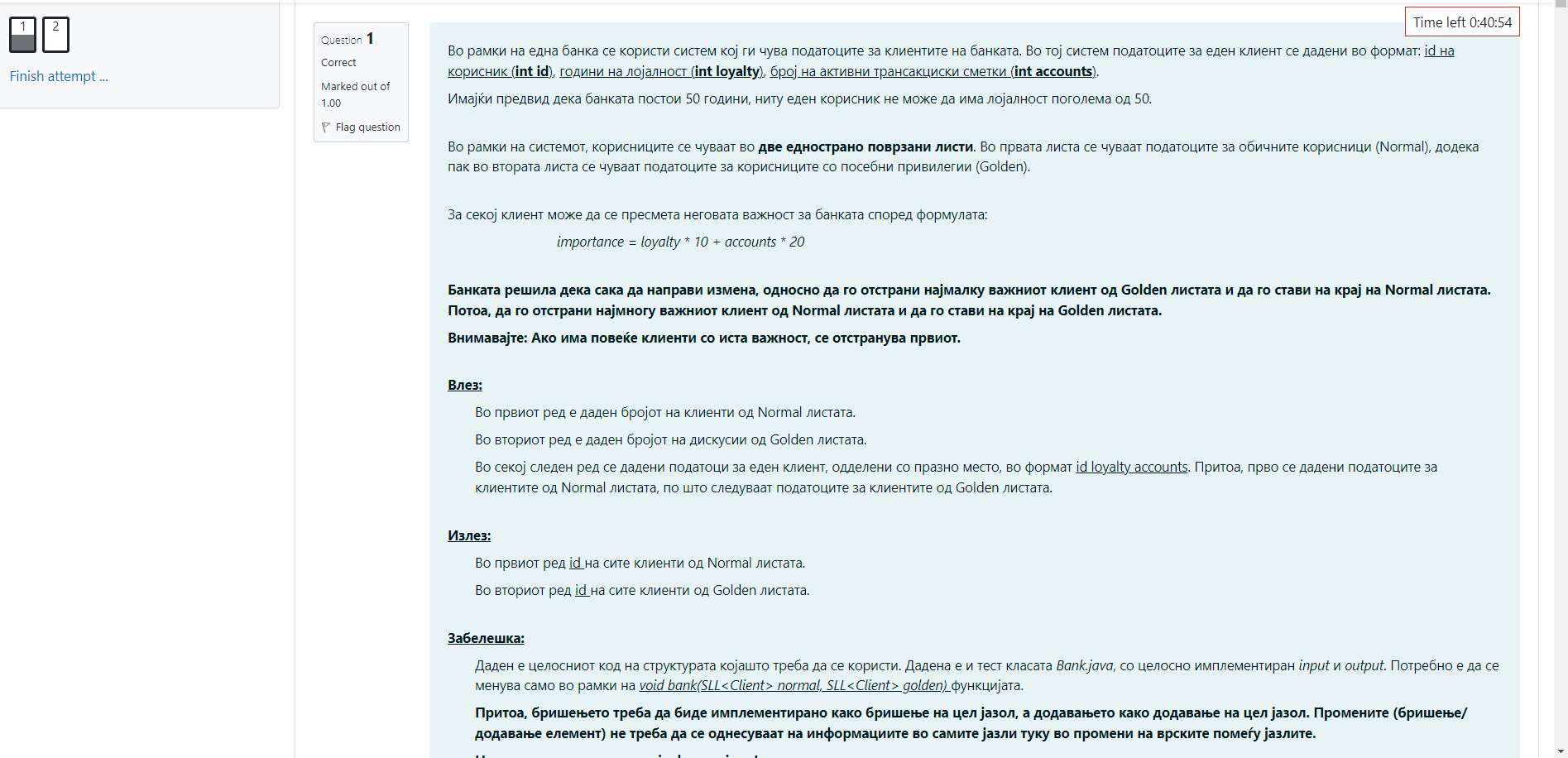
Испит 16/06/2022 Термин 2



import java.util.Scanner;

class Client {

private int id;

private int loyalty;

private int accounts;

public Client(int id, int loyalty, int accounts) {

this.id = id;

this.loyalty = loyalty;

this.accounts = accounts;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public int getLoyalty() {

return loyalty;

}

public void setLoyalty(int loyalty) {

this.loyalty = loyalty;

}

public int getAccounts() {

return accounts;

}

public void setAccounts(int accounts) {

this.accounts = accounts;

}

public int calculateImportance() {

return loyalty \* 10 + accounts \* 20;

}

@Override

public String toString() {

return String.valueOf(id);

}

}

class SLLNode<E> {

protected E element;

protected SLLNode<E> succ;

public SLLNode(E elem, SLLNode<E> succ) {

this.element = elem;

this.succ = succ;

}

@Override

public String toString() {

return element.toString();

}

}

class SLL<E> {

private SLLNode<E> first;

public SLL() {

this.first = null;

}

public void deleteList() {

first = null;

}

public int length() {

int ret;

if (first != null) {

SLLNode<E> tmp = first;

ret = 1;

while (tmp.succ != null) {

tmp = tmp.succ;

ret++;

}

return ret;

} else

return 0;

}

@Override

public String toString() {

String ret = new String();

if (first != null) {

SLLNode<E> tmp = first;

ret += tmp;

while (tmp.succ != null) {

tmp = tmp.succ;

ret += " " + tmp;

}

} else

ret = "Prazna lista!!!";

return ret;

}

public void insertFirst(E o) {

SLLNode<E> ins = new SLLNode<E>(o, first);

first = ins;

}

public void insertAfter(E o, SLLNode<E> node) {

if (node != null) {

SLLNode<E> ins = new SLLNode<E>(o, node.succ);

node.succ = ins;

} else {

System.out.println("Dadenot jazol e null");

}

}

public void insertBefore(E o, SLLNode<E> before) {

if (first != null) {

SLLNode<E> tmp = first;

if (first == before) {

this.insertFirst(o);

return;

}

while (tmp.succ != before)

tmp = tmp.succ;

if (tmp.succ == before) {

SLLNode<E> ins = new SLLNode<E>(o, before);

tmp.succ = ins;

} else {

System.out.println("Elementot ne postoi vo listata");

}

} else {

System.out.println("Listata e prazna");

}

}

public void insertLast(E o) {

if (first != null) {

SLLNode<E> tmp = first;

while (tmp.succ != null)

tmp = tmp.succ;

SLLNode<E> ins = new SLLNode<E>(o, null);

tmp.succ = ins;

} else {

insertFirst(o);

}

}

public E deleteFirst() {

if (first != null) {

SLLNode<E> tmp = first;

first = first.succ;

return tmp.element;

} else {

System.out.println("Listata e prazna");

return null;

}

}

public E delete(SLLNode<E> node) {

if (first != null) {

SLLNode<E> tmp = first;

if (first == node) {

return this.deleteFirst();

}

while (tmp.succ != node && tmp.succ.succ != null)

tmp = tmp.succ;

if (tmp.succ == node) {

tmp.succ = tmp.succ.succ;

return node.element;

} else {

System.out.println("Elementot ne postoi vo listata");

return null;

}

} else {

System.out.println("Listata e prazna");

return null;

}

}

public SLLNode<E> getFirst() {

return first;

}

public SLLNode<E> find(E o) {

if (first != null) {

SLLNode<E> tmp = first;

while (tmp.element != o && tmp.succ != null)

tmp = tmp.succ;

if (tmp.element == o) {

return tmp;

} else {

System.out.println("Elementot ne postoi vo listata");

}

} else {

System.out.println("Listata e prazna");

}

return first;

}

}

public class Bank {

// TODO: REMOVE

public static void bank(SLL<Client> normal, SLL<Client> golden) {

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int numNormal = Integer.parseInt(scanner.nextLine());

int numGolden = Integer.parseInt(scanner.nextLine());

SLL<Client> normal = new SLL<Client>();

SLL<Client> golden = new SLL<Client>();

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

String line = scanner.nextLine();

String[] parts = line.split("\\s+");

normal.insertLast(new Client(Integer.parseInt(parts[0]), Integer.parseInt(parts[1]), Integer.parseInt(parts[2])));

}

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

String line = scanner.nextLine();

String[] parts = line.split("\\s+");

golden.insertLast(new Client(Integer.parseInt(parts[0]), Integer.parseInt(parts[1]), Integer.parseInt(parts[2])));

}

bank(normal, golden);

System.out.println(normal.toString());

System.out.println(golden.toString());

}

}

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 3  2  5636 38 8  1705 49 4  3606 48 1  6698 40 2  1178 2 40 | 5636 3606 6698  1178 1705 | 5636 3606 6698  1178 1705 |  |
|  | 2  2  4899 9 3  7878 9 3  4078 31 1  9098 31 1 | 4899 7878  9098 4078 | 4899 7878  9098 4078 |  |
|  | 2  2  9680 12 10  2513 21 10  6549 9 7  4028 9 7 | 9680 6549  4028 2513 | 9680 6549  4028 2513 |  |
|  | 5  5  8254 41 4  4292 50 4  3404 41 4  4718 20 5  371 16 3  7113 35 1  7352 47 9  7610 2 2  10080 2 2  2322 2 2 | 8254 3404 4718 371 7610  7113 7352 10080 2322 4292 | 8254 3404 4718 371 7610  7113 7352 10080 2322 4292 |  |
|  | 15  15  4858 13 4  8255 29 9  1428 18 4  4030 10 2  1600 20 10  7389 48 4  2924 41 1  275 24 4  3513 45 9  4927 47 6  8518 31 9  5244 42 7  697 3 6  8489 38 1  3829 17 7  6739 10 4  9112 20 7  6531 37 3  2829 5 9  1095 30 9  3265 43 6  10092 35 8  6837 48 2  7045 49 10  751 23 3  6820 2 4  2208 34 7  9914 9 6  3018 28 1  4220 45 1 | 4858 8255 1428 4030 1600 7389 2924 275 4927 8518 5244 697 8489 3829 6820  6739 9112 6531 2829 1095 3265 10092 6837 7045 751 2208 9914 3018 4220 3513 | 4858 8255 1428 4030 1600 7389 2924 275 4927 8518 5244 697 8489 3829 6820  6739 9112 6531 2829 1095 3265 10092 6837 7045 751 2208 9914 3018 4220 3513 |  |
|  | 6  6  5531 24 3  2695 2 7  5555 38 9  9121 7 2  9943 22 5  6252 47 9  1024 7 5  9723 41 10  9566 25 3  8201 6 7  7106 7 6  1536 17 4 | 5531 2695 5555 9121 9943 1024  9723 9566 8201 7106 1536 6252 | 5531 2695 5555 9121 9943 1024  9723 9566 8201 7106 1536 6252 |  |
|  | 2  2  2004 48 7  6578 29 2  3524 26 5  8092 47 4 | 6578 3524  8092 2004 | 6578 3524  8092 2004 |  |
|  | 17  15  282 35 8  6829 18 8  2593 5 10  8417 17 7  9360 8 2  10029 20 6  1217 9 5  3707 17 6  5689 19 3  9849 3 9  9529 33 1  8943 49 3  2751 50 9  8465 32 7  6002 6 3  3070 6 5  5559 40 6  10030 35 9  9485 49 6  184 12 7  6613 23 1  4049 26 2  2682 5 3  6083 11 5  8003 49 7  3820 17 7  10001 20 8  8273 20 2  4761 3 10  1574 24 9  3089 24 10  1076 47 8 | 282 6829 2593 8417 9360 10029 1217 3707 5689 9849 9529 8943 8465 6002 3070 5559 2682  10030 9485 184 6613 4049 6083 8003 3820 10001 8273 4761 1574 3089 1076 2751 | 282 6829 2593 8417 9360 10029 1217 3707 5689 9849 9529 8943 8465 6002 3070 5559 2682  10030 9485 184 6613 4049 6083 8003 3820 10001 8273 4761 1574 3089 1076 2751 |  |
|  | 7  5  2855 45 8  6497 31 8  3858 13 3  8550 15 6  7854 38 2  837 8 1  5323 32 10  2916 28 6  2225 32 4  3409 43 8  2624 20 10  2576 50 10 | 6497 3858 8550 7854 837 5323 2916  2225 3409 2624 2576 2855 | 6497 3858 8550 7854 837 5323 2916  2225 3409 2624 2576 2855 |  |
|  | 3  4  7622 17 10  3607 9 9  3525 9 5  437 37 10  10050 24 5  8264 21 4  6561 36 7 | 3607 3525 8264  437 10050 6561 7622 | 3607 3525 8264  437 10050 6561 7622 |  |

Passed all tests!