GIT Department of Computer Engineering

CSE 222/505 - Spring 2021

Homework 3 Report

CANBERK ARICI

171044062

SYSTEM REQUIREMENTS

There are 3 types of user in this automation project and they are administrators, branch employees, customers. Administrators can add or remove branches, add or remove branch employees, see all branches and employees, also query whether there are any products that need to be supplied. Branch employees can check product stock, add or remove product, view previous orders of a customer and make sale.Customers can view product list and search for a product without login, customers have to login to buy a product and view their previous orders or if customer is shopping for the first time, customer will be subscribed to the system and when a customer subscribes to the system, a customer number will be created for this customer. If customers want to buy online, they have to enter their address and phone number. If the company does not have enough number of product that requested by customer, branch employee informs manager(administrator) and the product will be queried and supplied by manager(administrator). When customer buys a product, branch employee updates customer's previous orders and decrements number of the product in stock.System has users that are

administrators, branch employees and customers so there should be an interface for the users. This interface is called Person and it has common information of users that are name and surname, getter – setter for name and surname. There should be a class that is for Company. Company class has information about furniture, customers, employees, branches and it is more general than other classes. There are branches of company so there should be a class for branches and it is Branch

class. Branch class has information about branches. Branches of company has branch employees so there should be a class for branch employees. Branch Employee class has information about branch employees and methods that are branch employees' actions like making sale, seeing product list, adding product, removing product, checking whether customer is subscribed, subscribing customer to the system, viewing previous orders of a subscribed customer. There should be a class for administrators because system needs a manager. Administrator can take actions about braches, branch employees and products. The administrator class has information about administrator and methods that are administrators' actions like adding branch, removing branch, adding branch employee, removing branch employee, querying and adding products. Company has customers so there should be a class for customers. Customer class has information of customer and methods that are customers' actions like seeing product list, searching for a product, viewing previous orders and buying product. Also there are products of company so there should be a class for products and it is Furniture class. Furniture class has information about furniture and methods for these information. These classes have connections between them. These classes will use ArrayList, Linked List and HybridList according to instructions given.

Functional Requirements:

Administrator:

- **Adds branch**
- **Removes branch**
- **Adds branch employee**
- **Removes branch employee**
- **Query product**

Branch Employee:

- **Inquire about products in stock**
- **Inform manager that needed product should be purchased**
- **Add product**
- **Remove product**
- **Access the information of the previous orders of a customer by using the customer number**
- **Update customer's previous orders when customer buys furniture**

Customer:

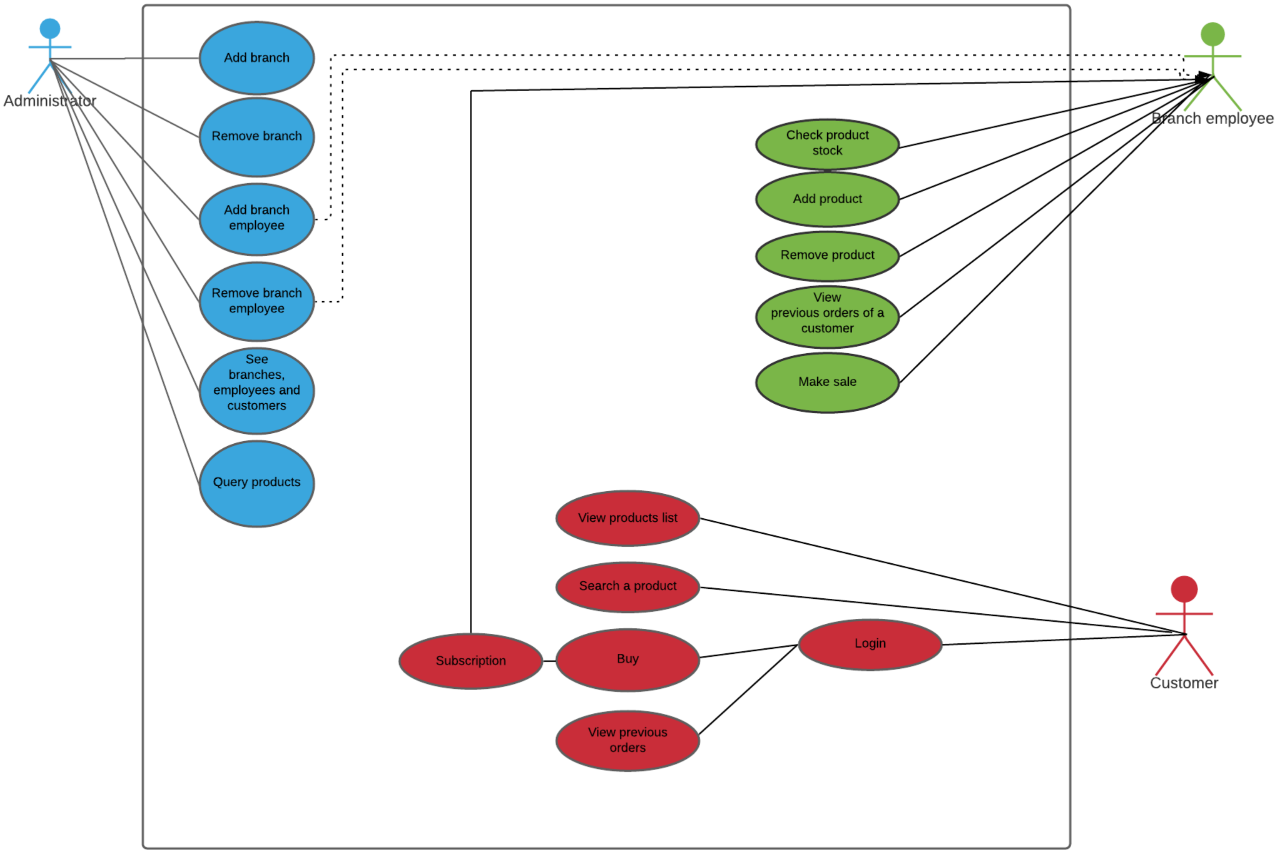
- **Search for product**
- **See the list of products**
- **See which store a product is in**
- **Shop online**
- **View their previous orders**

Non-Functional Requirements:

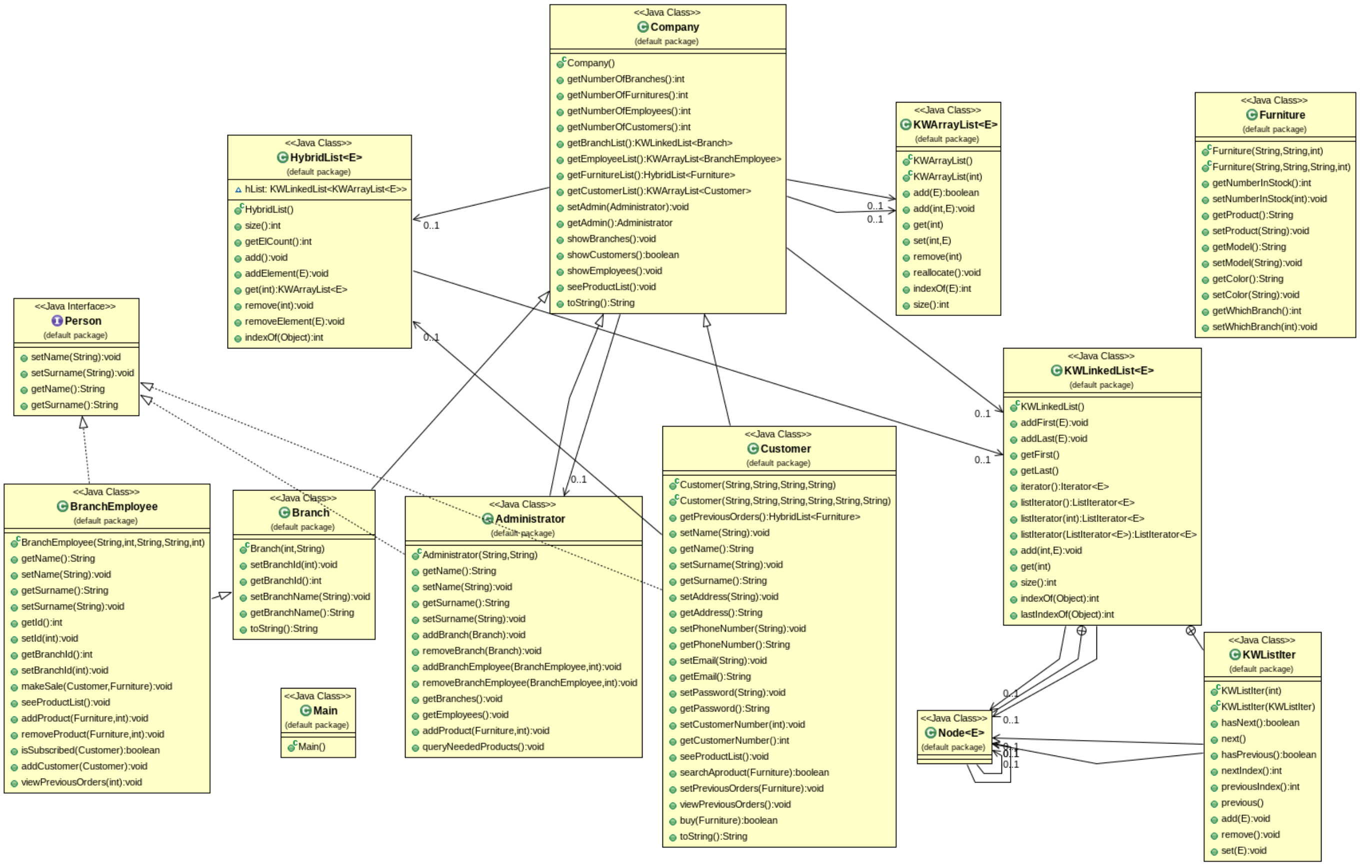
- **User friendly interface**
- **Less needed memory**
- **Detailed implementation**

*** javac version 11.0.10 is used**

USE CASE DIAGRAMS



CLASS DIAGRAMS

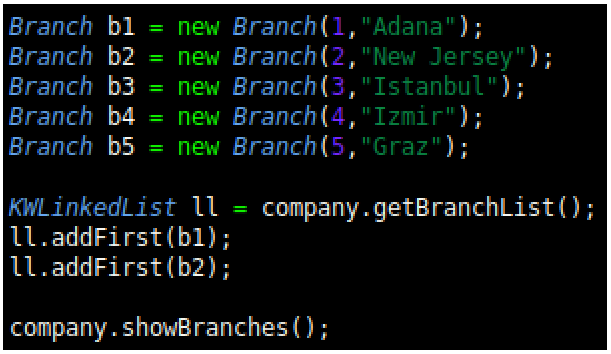


PROBLEM SOLUTIONS APPROACH

I firstly changed usage of array for users of my automation system to array list by changing every user array in code, it provided so much ease. Then I changed usage of array for branches to linked list by changing every, linked list also provided so ease. Also these changes improved efficieny of code as well. Making these changes were not very hard. Then I implemented HybridList class by using linked list as a component and I stored an arraylist in each node, it was really hard to think how could it be implemented, I tried many ways and these trials showed me the right way. Since we have linked list as a component, I have benefited its ease in some parts also I used ease of array list in some parts. I wrote appropriate methods for HybridList class and made it work as wanted. Finally I implemented HybridList into automation system.

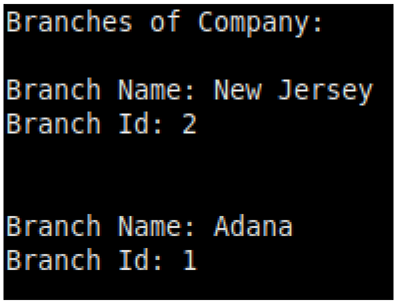
TEST CASES

Test Case #	Test Case Description
1	Add to head of linked list



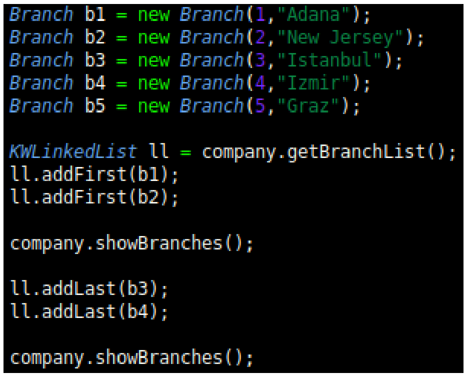
```
Branch b1 = new Branch(1,"Adana");
Branch b2 = new Branch(2,"New Jersey");
Branch b3 = new Branch(3,"Istanbul");
Branch b4 = new Branch(4,"Izmir");
Branch b5 = new Branch(5,"Graz");

KWLinkedList ll = company.getBranchList();
ll.addFirst(b1);
ll.addFirst(b2);

company.showBranches();
```

Result:

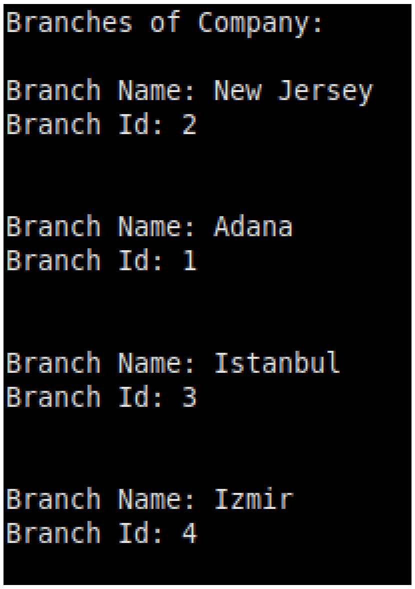


```
Branches of Company:

Branch Name: New Jersey
Branch Id: 2


Branch Name: Adana
Branch Id: 1
```

Test Case #	Test Case Description
2	Add to tail of linked list

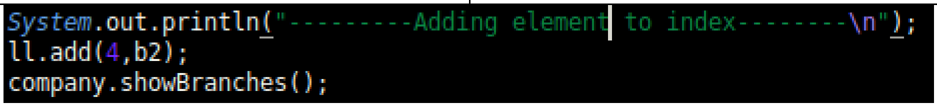


```
Branch b1 = new Branch(1,"Adana");
Branch b2 = new Branch(2,"New Jersey");
Branch b3 = new Branch(3,"Istanbul");
Branch b4 = new Branch(4,"Izmir");
Branch b5 = new Branch(5,"Graz");

KWLinkedList ll = company.getBranchList();
ll.addFirst(b1);
ll.addFirst(b2);

company.showBranches();

ll.addLast(b3);
ll.addLast(b4);

company.showBranches();
```

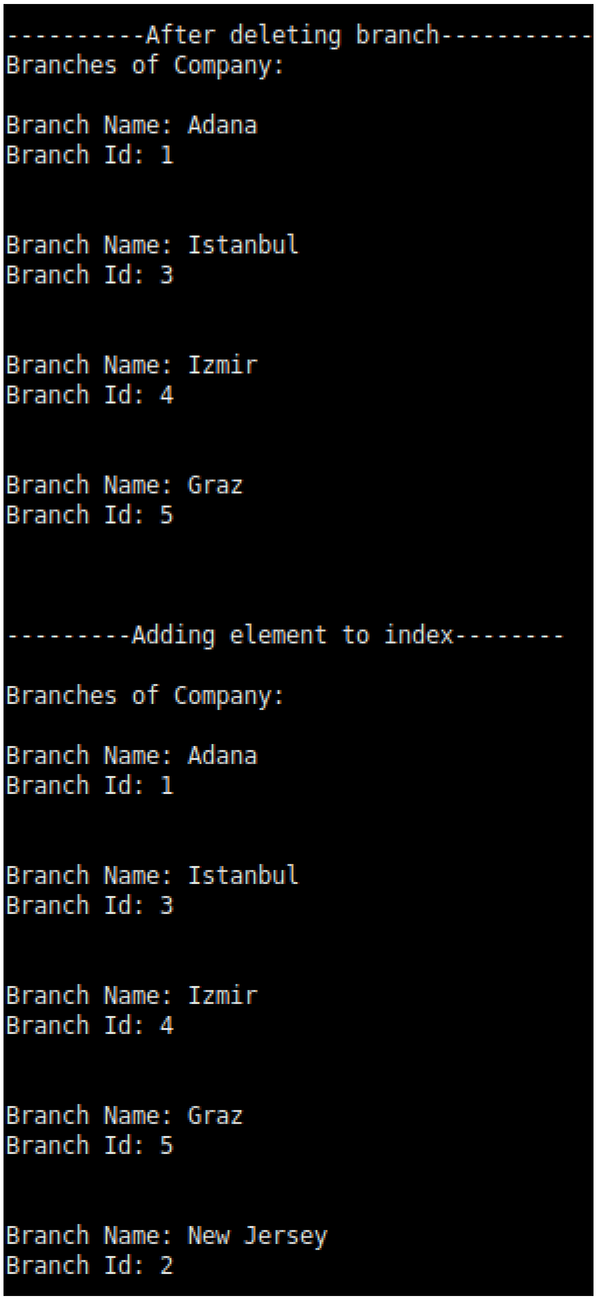


```
Branches of Company:

Branch Name: New Jersey
Branch Id: 2


Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4
```

Test Case #	Test Case Description
3	Add to ith index in linked list

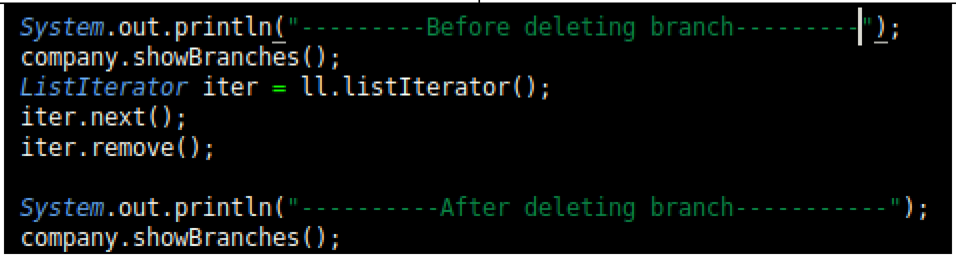


```
System.out.println("---------Adding element to index--------\n");
ll.add(4,b2);
company.showBranches();
```

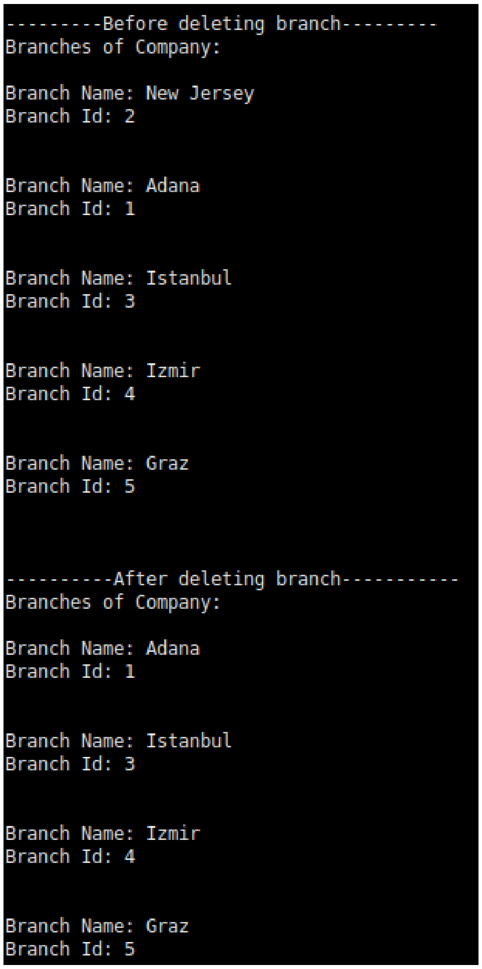


```
----------After deleting branch-----------
Branches of Company:

Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5



---------Adding element to index--------

Branches of Company:

Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5


Branch Name: New Jersey
Branch Id: 2
```

Test Case #	Test Case Description
4	Remove element from linked list

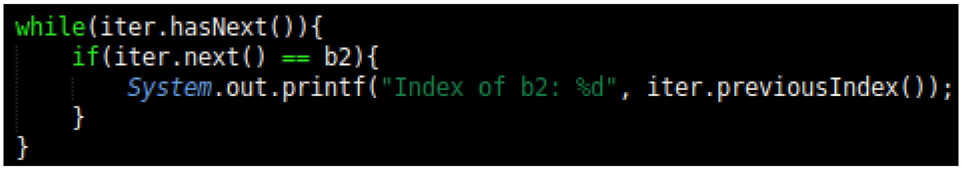


```
System.out.println("---------Before deleting branch---------");
company.showBranches();
ListIterator iter = ll.listIterator();
iter.next();
iter.remove();

System.out.println("----------After deleting branch-----------");
company.showBranches();
```

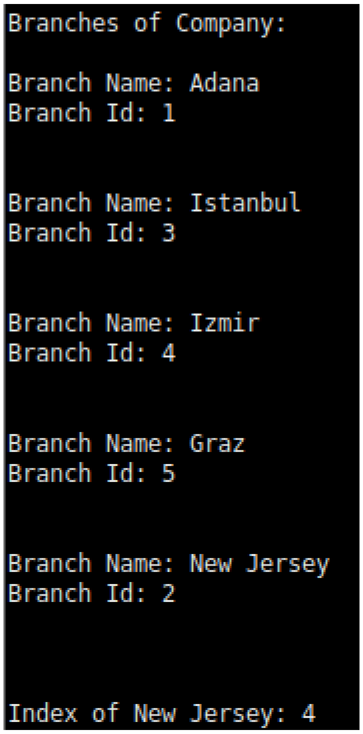


```
---------Before deleting branch---------
Branches of Company:

Branch Name: New Jersey
Branch Id: 2


Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5



----------After deleting branch-----------
Branches of Company:

Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5
```

Test Case #	Test Case Description
5	Search for a product in linked list and print its index

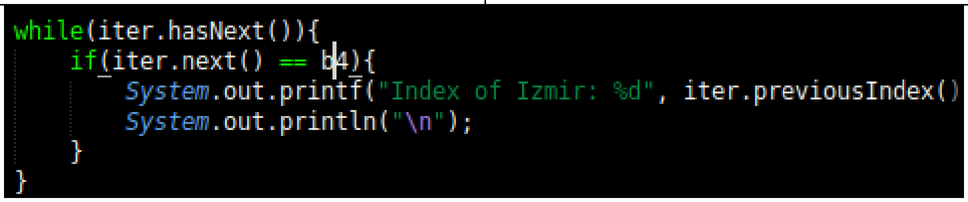


```
while(iter.hasNext()){
    if(iter.next() == b2){
        System.out.printf("Index of b2: %d", iter.previousIndex());
    }
}
```

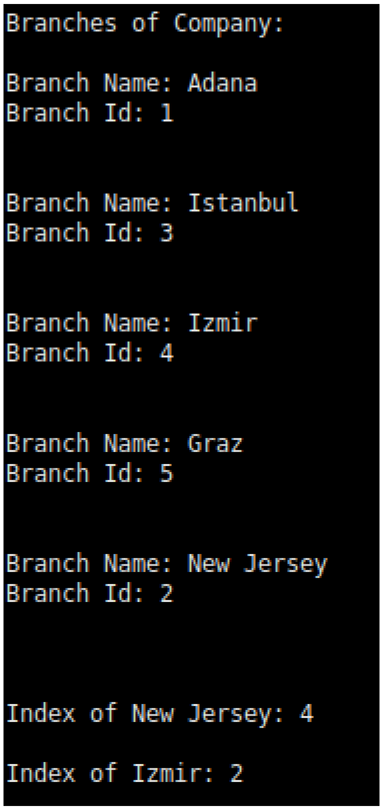


```
Branches of Company:

Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5


Branch Name: New Jersey
Branch Id: 2



Index of New Jersey: 4
```

Test Case #	Test Case Description
6	Search for a product in linked list and print its index

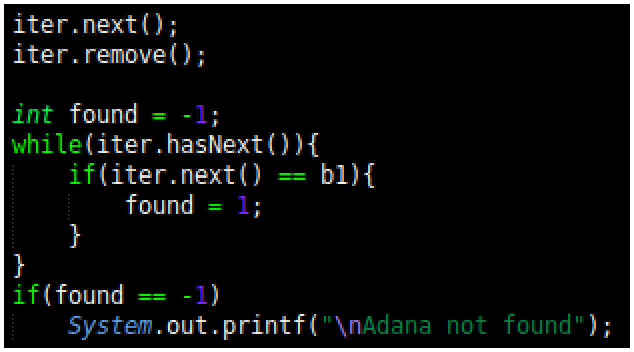


```java
while(iter.hasNext()){
    if(iter.next() == b4){
        System.out.printf("Index of Izmir: %d", iter.previousIndex()
        System.out.println("\n");
    }
}
```

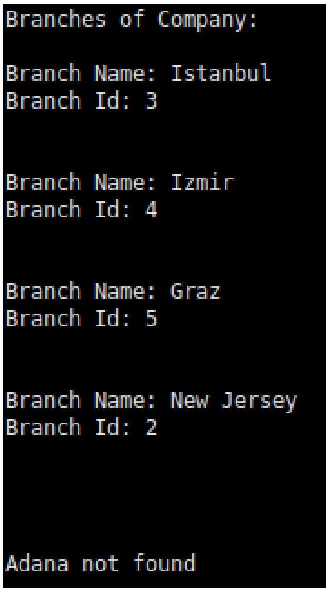


```
Branches of Company:

Branch Name: Adana
Branch Id: 1


Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5


Branch Name: New Jersey
Branch Id: 2



Index of New Jersey: 4

Index of Izmir: 2
```

Test Case #	Test Case Description
7	Delete an existing item from the list and make search

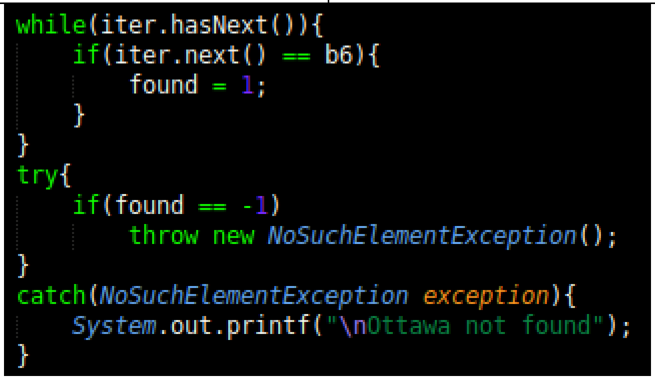


```
iter.next();
iter.remove();

int found = -1;
while(iter.hasNext()){
    if(iter.next() == b1){
        found = 1;
    }
}
if(found == -1)
    System.out.printf("\nAdana not found");
```

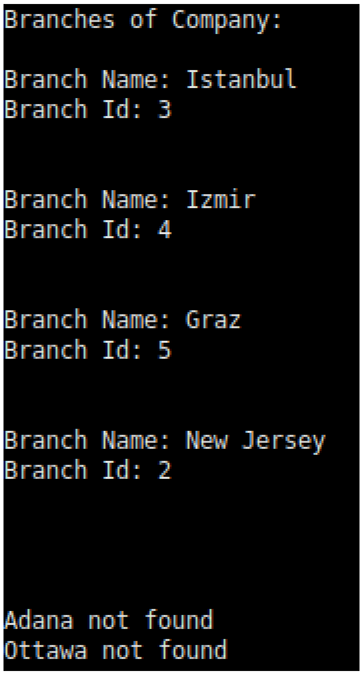


```
Branches of Company:

Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5


Branch Name: New Jersey
Branch Id: 2




Adana not found
```

Test Case #	Test Case Description
8	Search for a nonexistent product in linked list

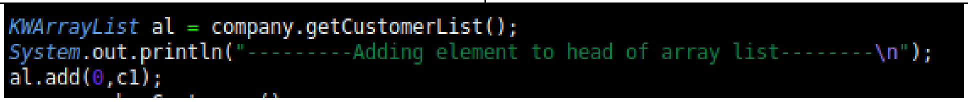


```
while(iter.hasNext()){
    if(iter.next() == b6){
        found = 1;
    }
}
try{
    if(found == -1)
        throw new NoSuchElementException();
}
catch(NoSuchElementException exception){
    System.out.printf("\nOttawa not found");
}
```

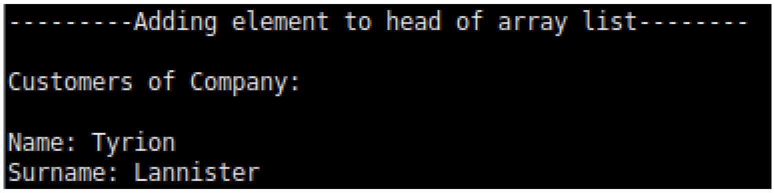


```
Branches of Company:

Branch Name: Istanbul
Branch Id: 3


Branch Name: Izmir
Branch Id: 4


Branch Name: Graz
Branch Id: 5


Branch Name: New Jersey
Branch Id: 2




Adana not found
Ottawa not found
```

Test Case #	Test Case Description
9	Adding element to head of array list

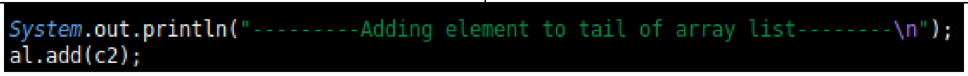


```
KWArrayList al = company.getCustomerList();
System.out.println("---------Adding element to head of array list--------\n");
al.add(0,c1);
```

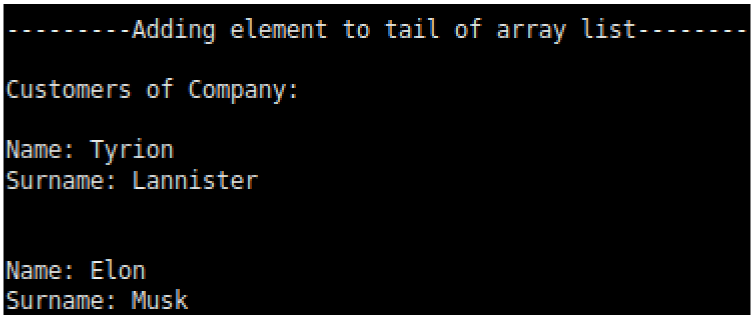


```
---------Adding element to head of array list--------

Customers of Company:

Name: Tyrion
Surname: Lannister
```

Test Case #	Test Case Description
10	Adding element to tail of array list

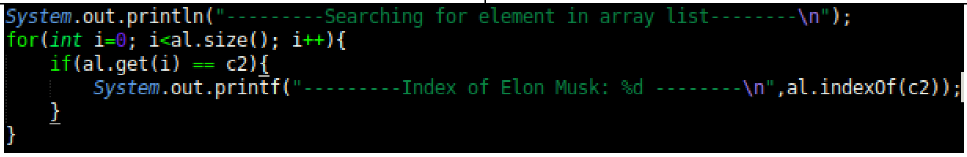


```
System.out.println("---------Adding element to tail of array list--------\n");
al.add(c2);
```

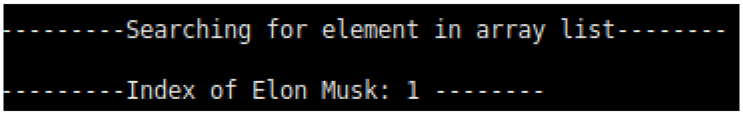


```
---------Adding element to tail of array list--------

Customers of Company:

Name: Tyrion
Surname: Lannister


Name: Elon
Surname: Musk
```

Test Case #	Test Case Description
11	Searching for element in array list

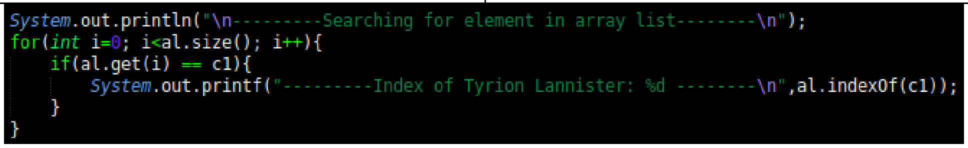


```
System.out.println("---------Searching for element in array list--------\n");
for(int i=0; i<al.size(); i++){
    if(al.get(i) == c2){
        System.out.printf("---------Index of Elon Musk: %d --------\n",al.indexOf(c2));
    }
}
```

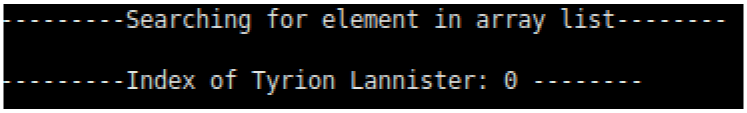


```
---------Searching for element in array list--------

---------Index of Elon Musk: 1 --------
```

Test Case #	Test Case Description
12	Searching for element in array list

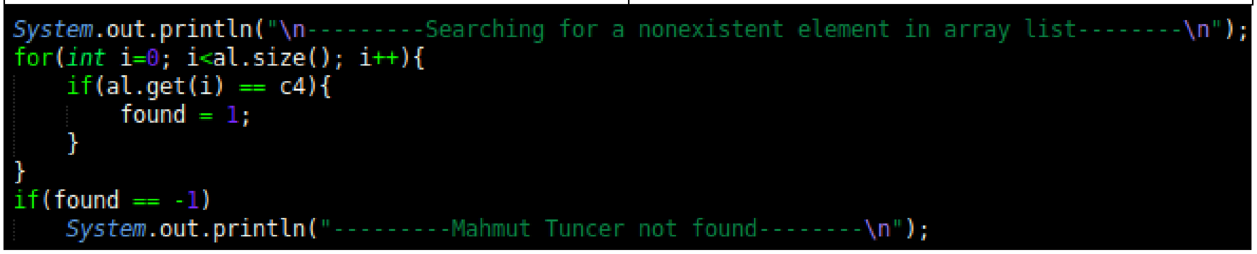


```
System.out.println("\n---------Searching for element in array list--------\n");
for(int i=0; i<al.size(); i++){
    if(al.get(i) == c1){
        System.out.printf("---------Index of Tyrion Lannister: %d --------\n",al.indexOf(c1));
    }
}
```

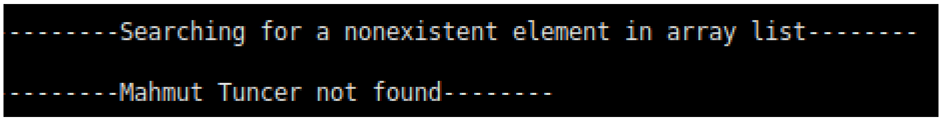


```
---------Searching for element in array list--------

---------Index of Tyrion Lannister: 0 --------
```

Test Case #	Test Case Description
13	Searching for a nonexistent element in array list

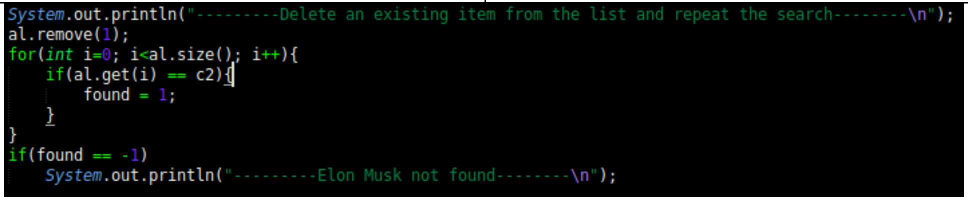


```
System.out.println("\n---------Searching for a nonexistent element in array list--------\n");
for(int i=0; i<al.size(); i++){
    if(al.get(i) == c4){
        found = 1;
    }
}
if(found == -1)
    System.out.println("---------Mahmut Tuncer not found--------\n");
```

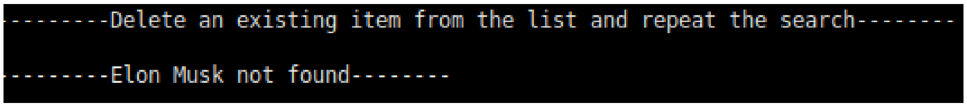


```
---------Searching for a nonexistent element in array list--------

---------Mahmut Tuncer not found--------
```

Test Case #	Test Case Description
14	Delete an existing item from the list and repeat the search

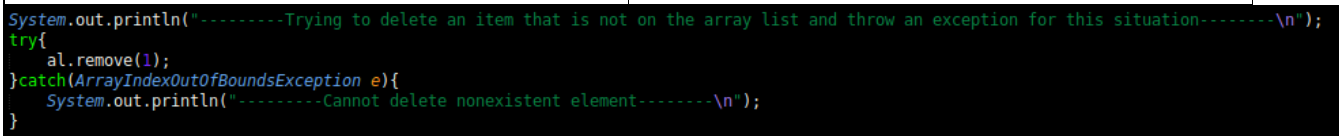


```
System.out.println("---------Delete an existing item from the list and repeat the search--------\n");
al.remove(1);
for(int i=0; i<al.size(); i++){
    if(al.get(i) == c2){
        found = 1;
    }
}
if(found == -1)
    System.out.println("---------Elon Musk not found--------\n");
```

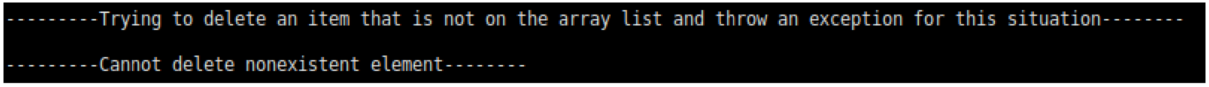


```
---------Delete an existing item from the list and repeat the search--------

---------Elon Musk not found--------
```

Test Case #	Test Case Description
15	Trying to delete an item that is not on the array list and throw an exception for this situation

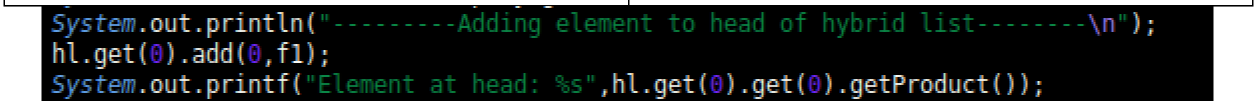


```
System.out.println("---------Trying to delete an item that is not on the array list and throw an exception for this situation--------\n");
try{
    al.remove(1);
}catch(ArrayIndexOutOfBoundsException e){
    System.out.println("---------Cannot delete nonexistent element--------\n");
}
```

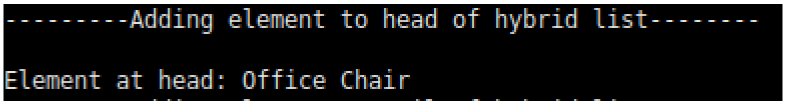


```
---------Trying to delete an item that is not on the array list and throw an exception for this situation--------

---------Cannot delete nonexistent element--------
```

Test Case #	Test Case Description
16	Adding element to head of hybrid list

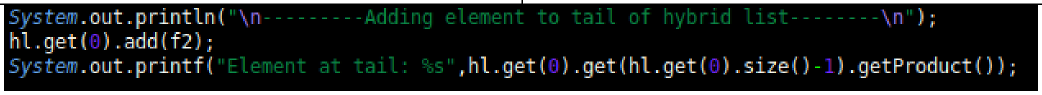


```
System.out.println("---------Adding element to head of hybrid list--------\n");
hl.get(0).add(0,f1);
System.out.printf("Element at head: %s",hl.get(0).get(0).getProduct());
```

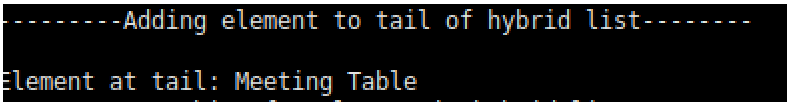


```
---------Adding element to head of hybrid list--------

Element at head: Office Chair
```

Test Case #	Test Case Description
17	Adding element to tail of hybrid list

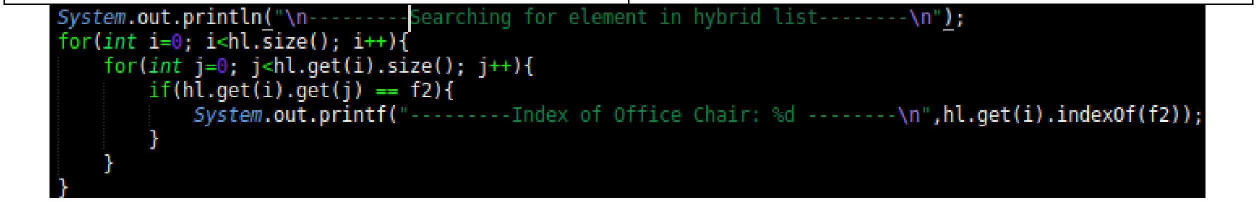


```
System.out.println("\n---------Adding element to tail of hybrid list--------\n");
hl.get(0).add(f2);
System.out.printf("Element at tail: %s",hl.get(0).get(hl.get(0).size()-1).getProduct());
```

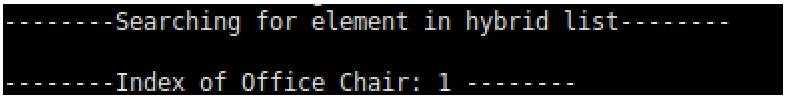


```
---------Adding element to tail of hybrid list--------

Element at tail: Meeting Table
```

Test Case #	Test Case Description
18	Searching for element in hybrid list

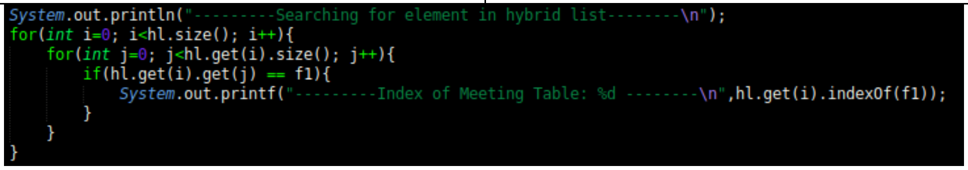


```
System.out.println("\n---------Searching for element in hybrid list--------\n");
for(int i=0; i<hl.size(); i++){
    for(int j=0; j<hl.get(i).size(); j++){
        if(hl.get(i).get(j) == f2){
            System.out.printf("---------Index of Office Chair: %d --------\n",hl.get(i).indexOf(f2));
        }
    }
}
```

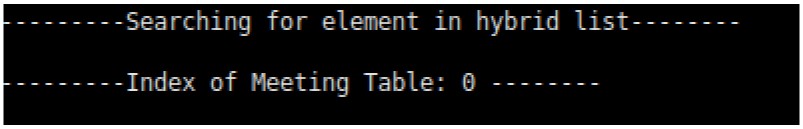


```
--------Searching for element in hybrid list--------

--------Index of Office Chair: 1 --------
```

Test Case #	Test Case Description
19	Searching for element in hybrid list

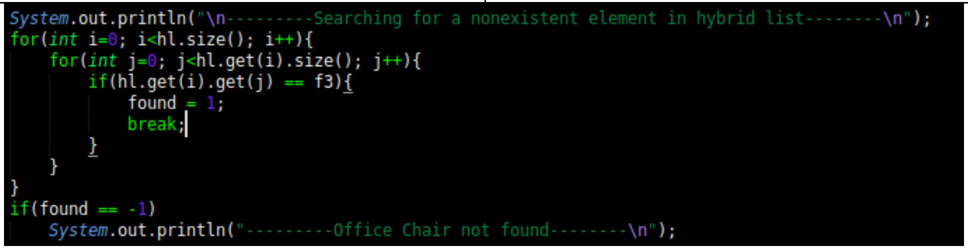


```
System.out.println("---------Searching for element in hybrid list--------\n");
for(int i=0; i<hl.size(); i++){
    for(int j=0; j<hl.get(i).size(); j++){
        if(hl.get(i).get(j) == f1){
            System.out.printf("---------Index of Meeting Table: %d --------\n",hl.get(i).indexOf(f1));
        }
    }
}
```

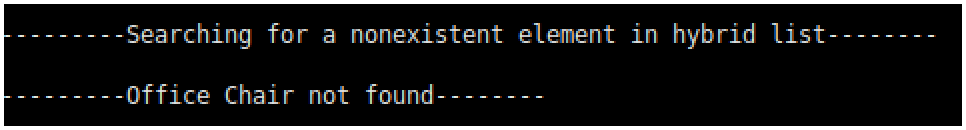


```
---------Searching for element in hybrid list--------

---------Index of Meeting Table: 0 --------
```

Test Case #	Test Case Description
20	Searching for a nonexistent element in hybrid list

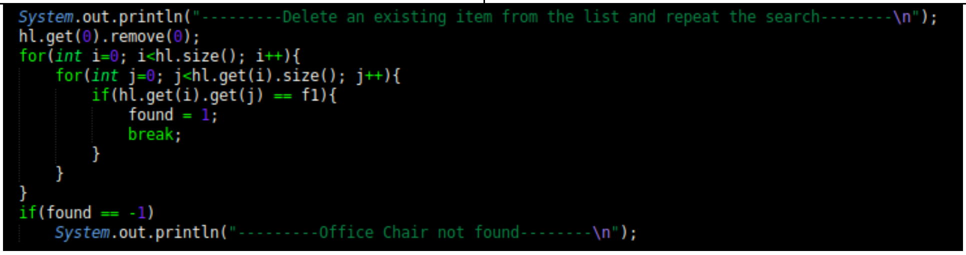


```
System.out.println("\n---------Searching for a nonexistent element in hybrid list--------\n");
for(int i=0; i<hl.size(); i++){
    for(int j=0; j<hl.get(i).size(); j++){
        if(hl.get(i).get(j) == f3){
            found = 1;
            break;
        }
    }
}
if(found == -1)
    System.out.println("---------Office Chair not found--------\n");
```

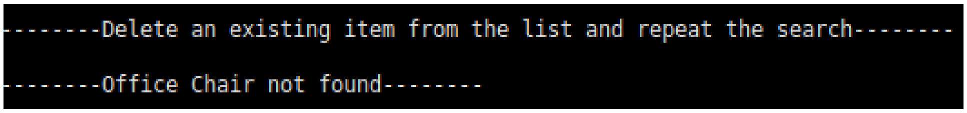


```
---------Searching for a nonexistent element in hybrid list--------

---------Office Chair not found--------
```

Test Case #	Test Case Description
21	Delete an existing item from the list and repeat the search

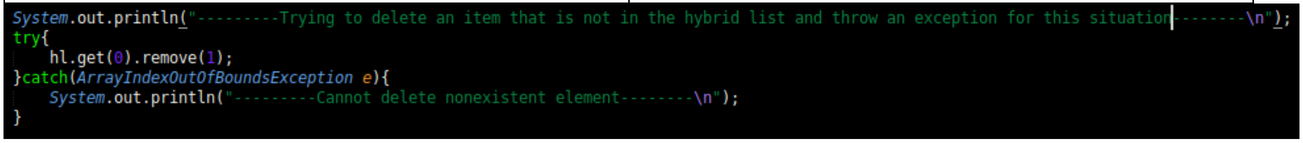


```
System.out.println("---------Delete an existing item from the list and repeat the search--------\n");
hl.get(0).remove(0);
for(int i=0; i<hl.size(); i++){
    for(int j=0; j<hl.get(i).size(); j++){
        if(hl.get(i).get(j) == f1){
            found = 1;
            break;
        }
    }
}
if(found == -1)
    System.out.println("---------Office Chair not found--------\n");
```

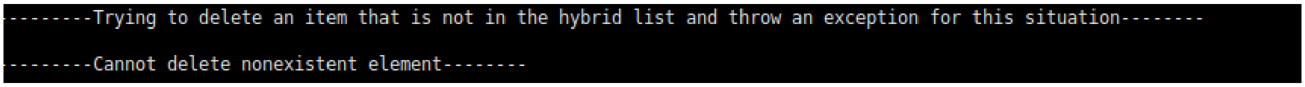


```
--------Delete an existing item from the list and repeat the search--------

--------Office Chair not found--------
```

Test Case #	Test Case Description
22	Trying to delete an item that is not in the hybrid list and throw an exception for this situation



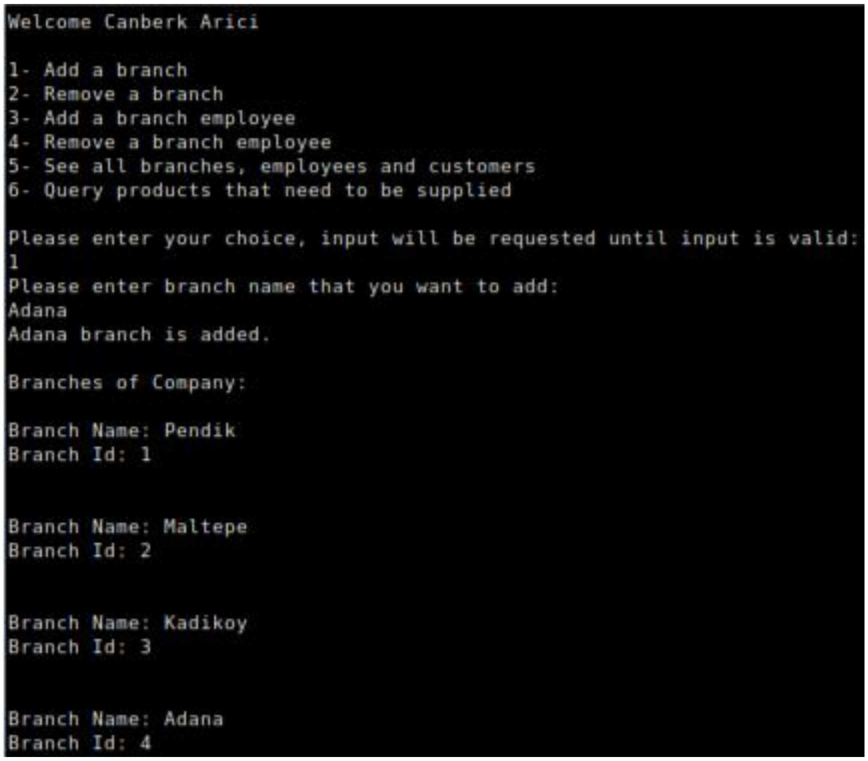
```
System.out.println("---------Trying to delete an item that is not in the hybrid list and throw an exception for this situation--------\n");
try{
    hl.get(0).remove(1);
}catch(ArrayIndexOutOfBoundsException e){
    System.out.println("---------Cannot delete nonexistent element--------\n");
}
```



```
--------Trying to delete an item that is not in the hybrid list and throw an exception for this situation--------

--------Cannot delete nonexistent element--------
```

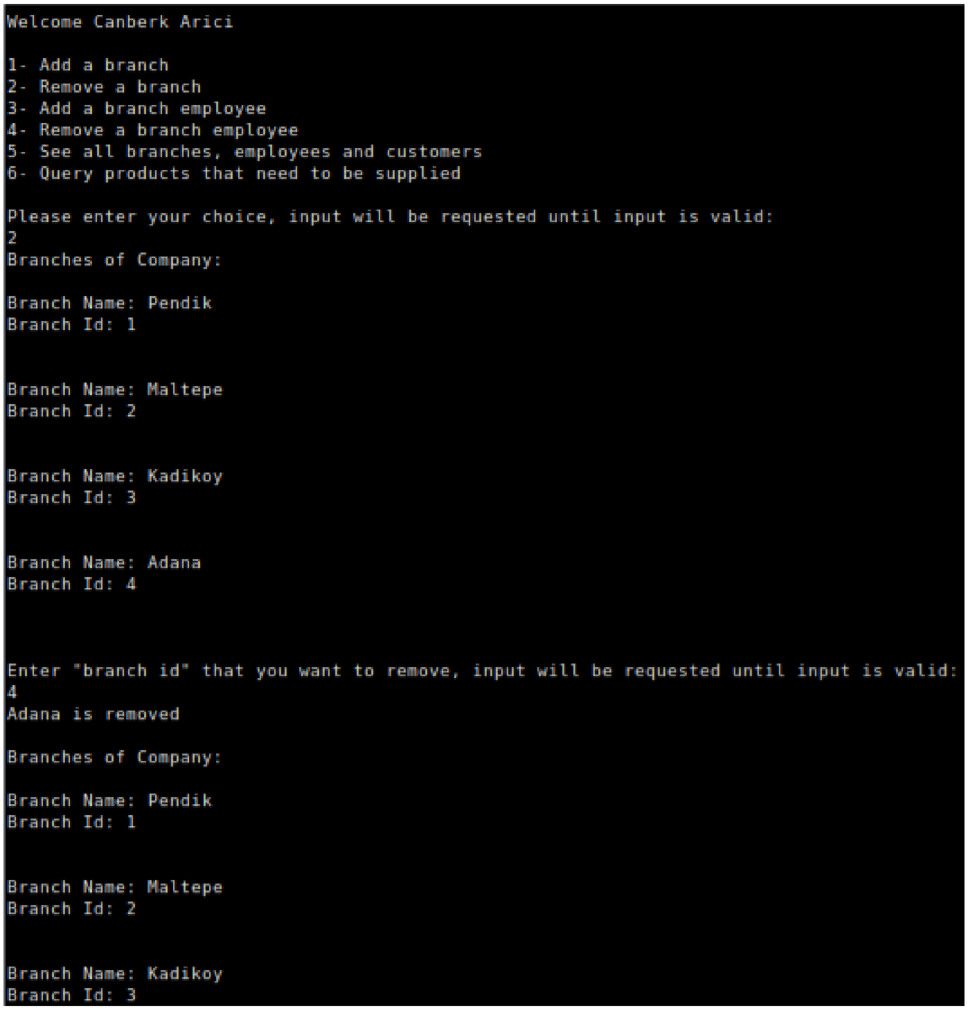
RUNNING COMMAND AND RESULTS

1) Adding branch (Admin)



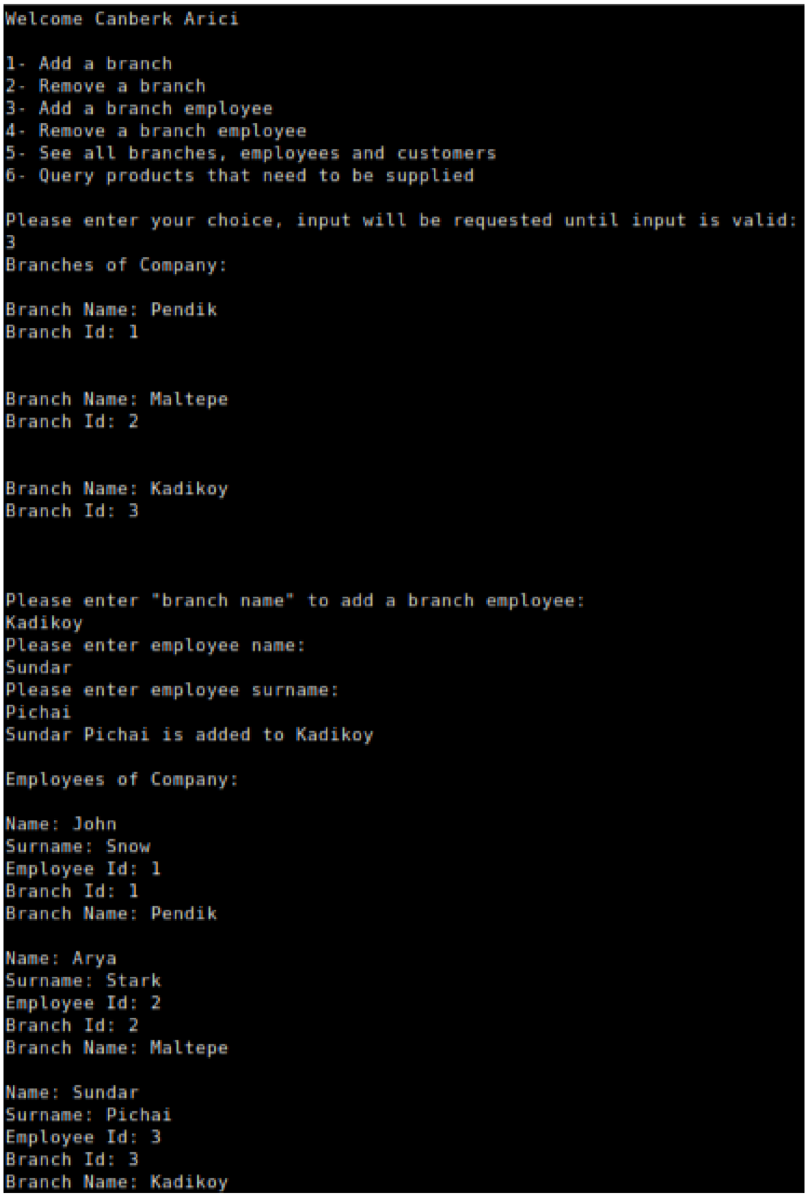
```
Welcome Canberk Arici

1- Add a branch
2- Remove a branch
3- Add a branch employee
4- Remove a branch employee
5- See all branches, employees and customers
6- Query products that need to be supplied

Please enter your choice, input will be requested until input is valid:
1
Please enter branch name that you want to add:
Adana
Adana branch is added.

Branches of Company:

Branch Name: Pendik
Branch Id: 1


Branch Name: Maltepe
Branch Id: 2


Branch Name: Kadikoy
Branch Id: 3


Branch Name: Adana
Branch Id: 4
```

2) Removing branch (Admin)



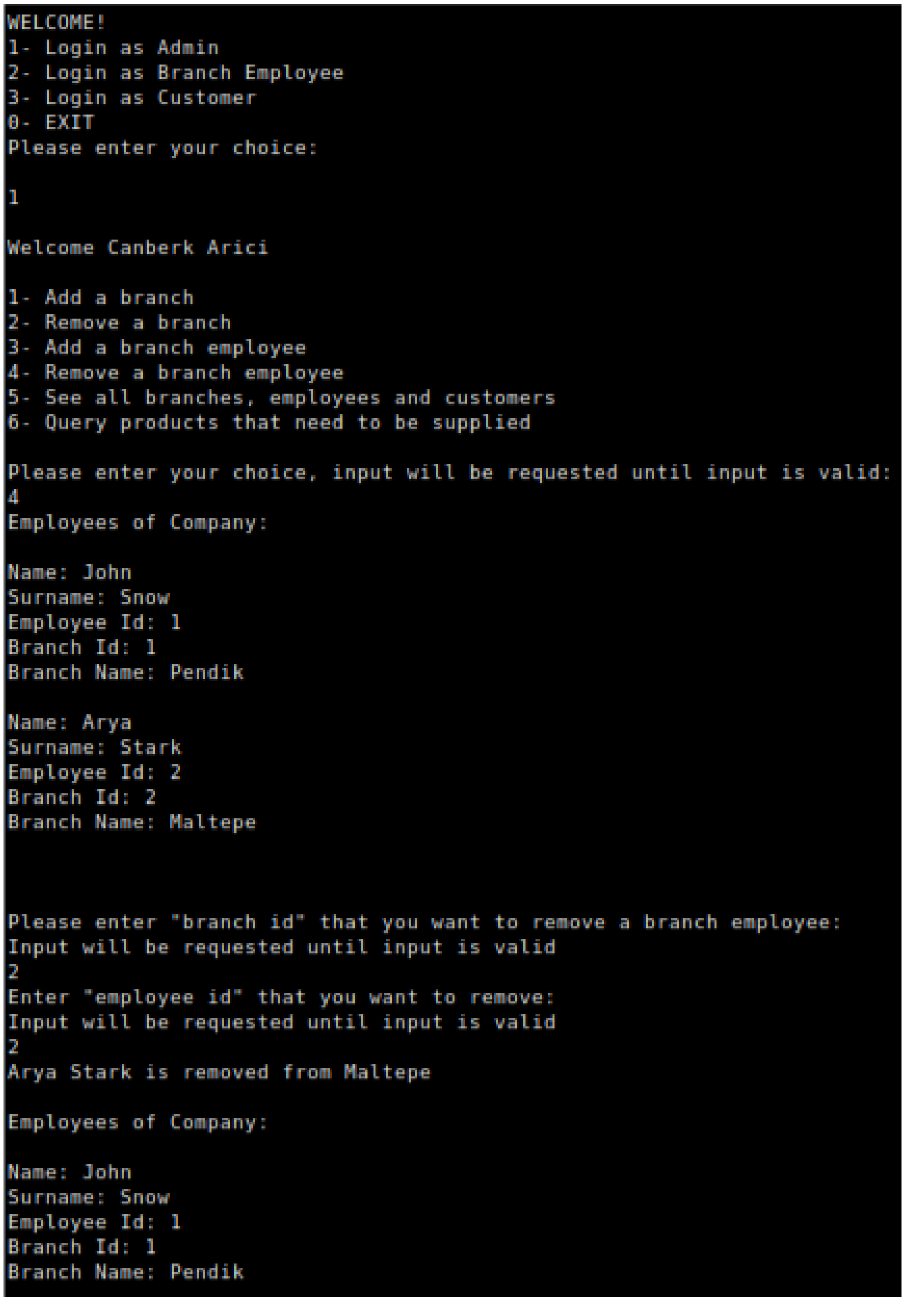
```
Welcome Canberk Arici

1- Add a branch
2- Remove a branch
3- Add a branch employee
4- Remove a branch employee
5- See all branches, employees and customers
6- Query products that need to be supplied

Please enter your choice, input will be requested until input is valid:
2
Branches of Company:

Branch Name: Pendik
Branch Id: 1


Branch Name: Maltepe
Branch Id: 2


Branch Name: Kadikoy
Branch Id: 3


Branch Name: Adana
Branch Id: 4



Enter "branch id" that you want to remove, input will be requested until input is valid:
4
Adana is removed

Branches of Company:

Branch Name: Pendik
Branch Id: 1


Branch Name: Maltepe
Branch Id: 2


Branch Name: Kadikoy
Branch Id: 3
```

3) Adding branch employee (Admin)



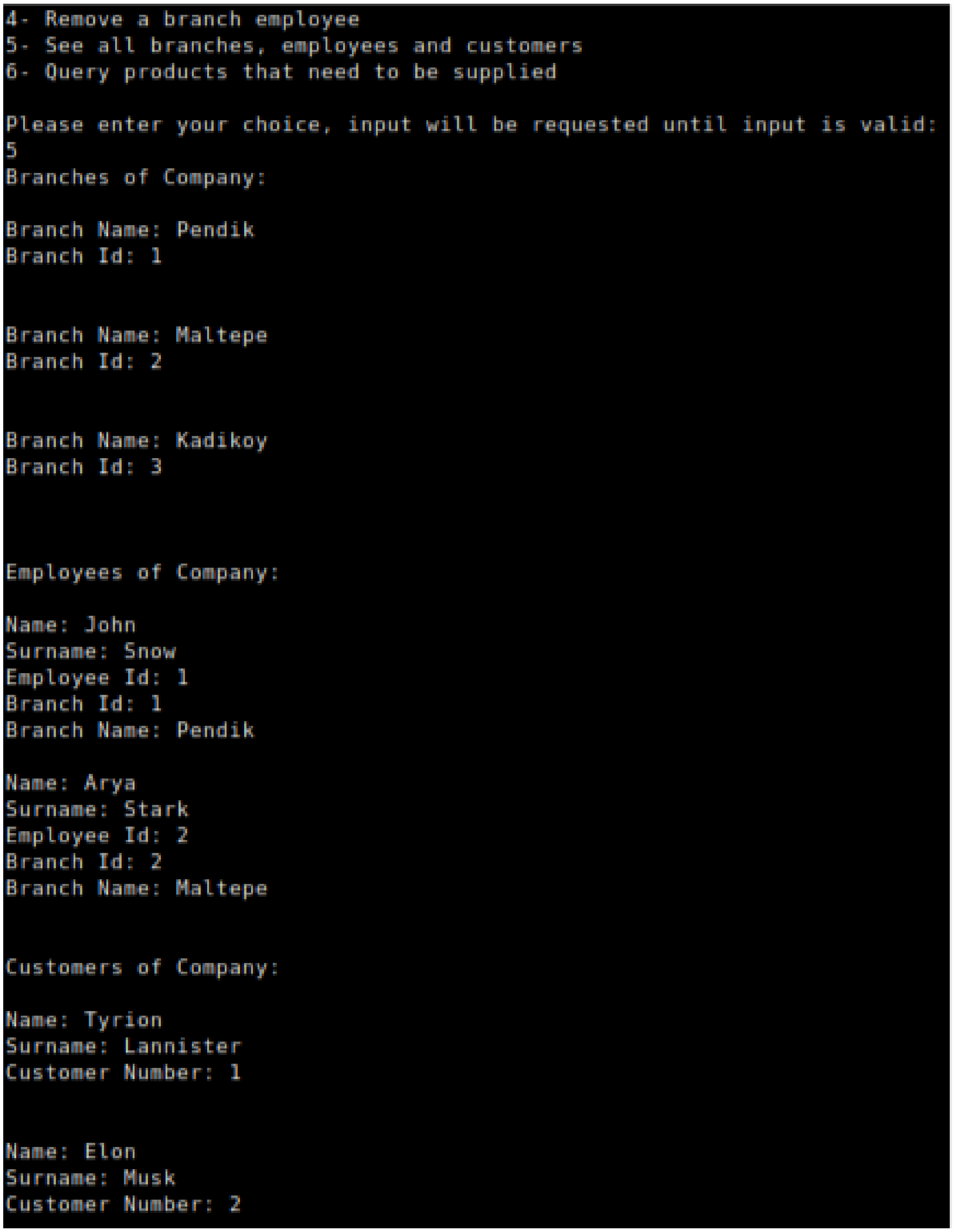
```
Welcome Canberk Arici

1- Add a branch
2- Remove a branch
3- Add a branch employee
4- Remove a branch employee
5- See all branches, employees and customers
6- Query products that need to be supplied

Please enter your choice, input will be requested until input is valid:
3
Branches of Company:

Branch Name: Pendik
Branch Id: 1


Branch Name: Maltepe
Branch Id: 2


Branch Name: Kadikoy
Branch Id: 3



Please enter "branch name" to add a branch employee:
Kadikoy
Please enter employee name:
Sundar
Please enter employee surname:
Pichai
Sundar Pichai is added to Kadikoy

Employees of Company:

Name: John
Surname: Snow
Employee Id: 1
Branch Id: 1
Branch Name: Pendik

Name: Arya
Surname: Stark
Employee Id: 2
Branch Id: 2
Branch Name: Maltepe

Name: Sundar
Surname: Pichai
Employee Id: 3
Branch Id: 3
Branch Name: Kadikoy
```

4) Removing branch employee (Admin)



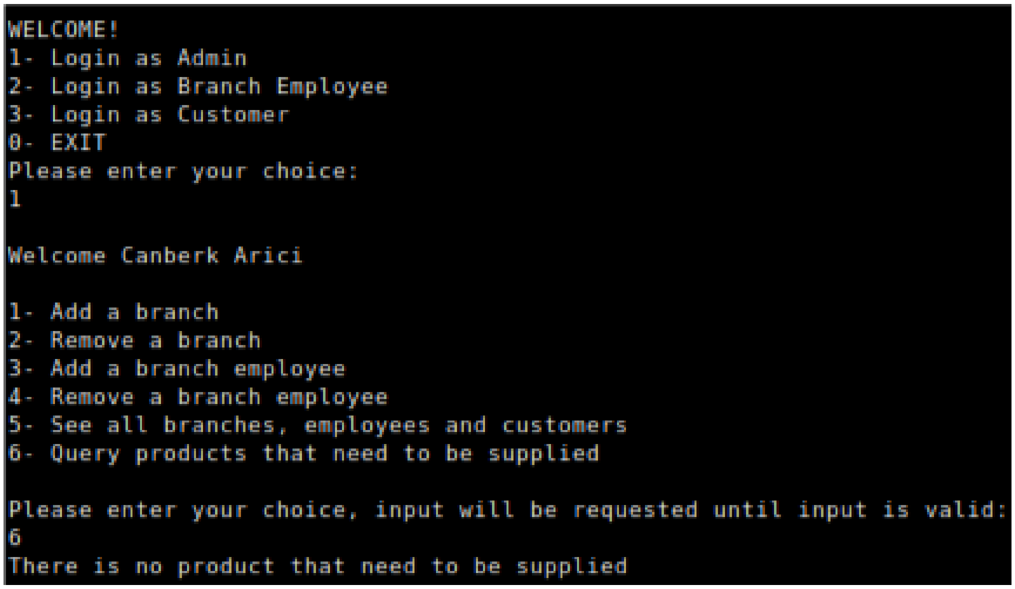
```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:

1

Welcome Canberk Arici

1- Add a branch
2- Remove a branch
3- Add a branch employee
4- Remove a branch employee
5- See all branches, employees and customers
6- Query products that need to be supplied

Please enter your choice, input will be requested until input is valid:
4
Employees of Company:

Name: John
Surname: Snow
Employee Id: 1
Branch Id: 1
Branch Name: Pendik

Name: Arya
Surname: Stark
Employee Id: 2
Branch Id: 2
Branch Name: Maltepe



Please enter "branch id" that you want to remove a branch employee:
Input will be requested until input is valid
2
Enter "employee id" that you want to remove:
Input will be requested until input is valid
2
Arya Stark is removed from Maltepe

Employees of Company:

Name: John
Surname: Snow
Employee Id: 1
Branch Id: 1
Branch Name: Pendik
```

5) See all branches, employees and customers (Admin)



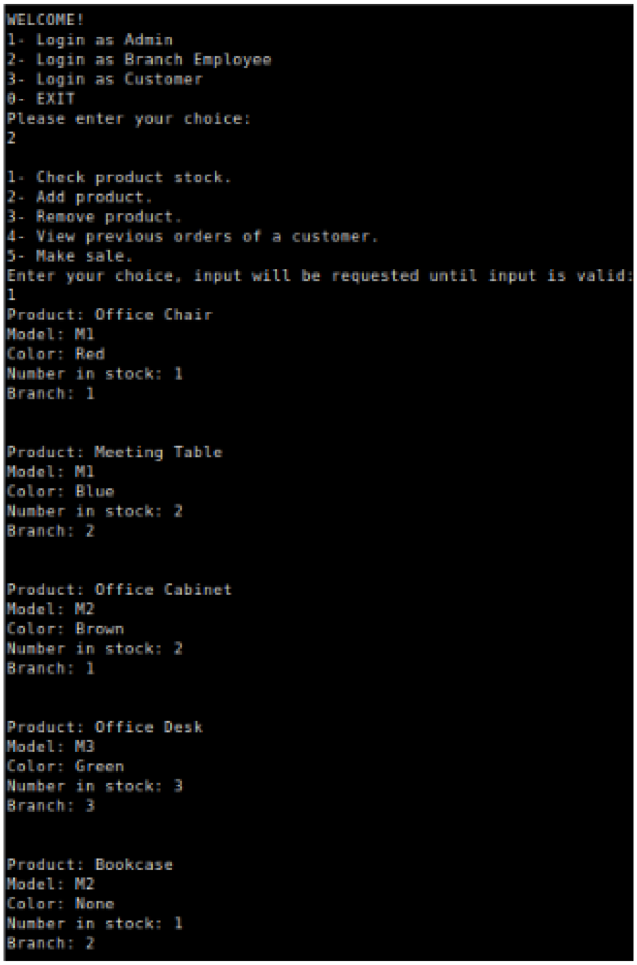
```
4- Remove a branch employee
5- See all branches, employees and customers
6- Query products that need to be supplied

Please enter your choice, input will be requested until input is valid:
5
Branches of Company:

Branch Name: Pendik
Branch Id: 1


Branch Name: Maltepe
Branch Id: 2


Branch Name: Kadikoy
Branch Id: 3



Employees of Company:

Name: John
Surname: Snow
Employee Id: 1
Branch Id: 1
Branch Name: Pendik

Name: Arya
Surname: Stark
Employee Id: 2
Branch Id: 2
Branch Name: Maltepe


Customers of Company:

Name: Tyrion
Surname: Lannister
Customer Number: 1


Name: Elon
Surname: Musk
Customer Number: 2
```

6) Query products that need to be supplied (Admin)



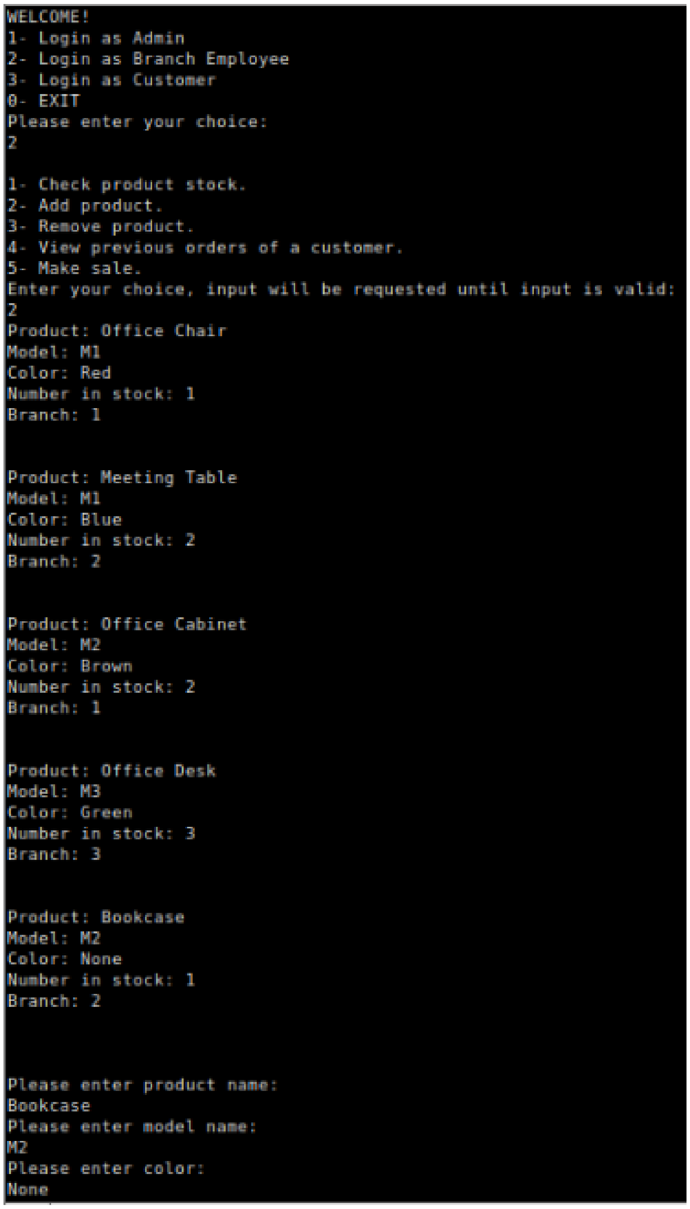
```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
1

Welcome Canberk Arici

1- Add a branch
2- Remove a branch
3- Add a branch employee
4- Remove a branch employee
5- See all branches, employees and customers
6- Query products that need to be supplied

Please enter your choice, input will be requested until input is valid:
6
There is no product that need to be supplied
```

7) Check product stock (Branch Employee)

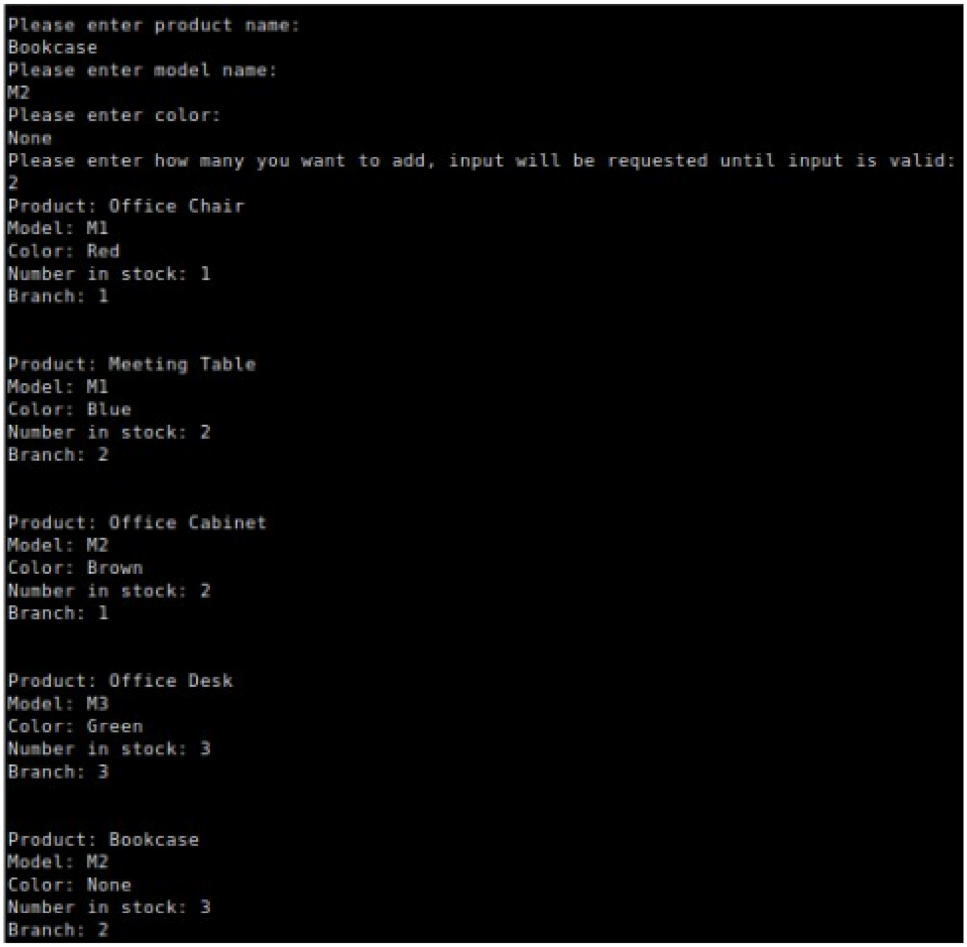


```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
2

1- Check product stock.
2- Add product.
3- Remove product.
4- View previous orders of a customer.
5- Make sale.
Enter your choice, input will be requested until input is valid:
1
Product: Office Chair
Model: M1
Color: Red
Number in stock: 1
Branch: 1


Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 1
Branch: 2
```

8) Add product (Branch Employee)

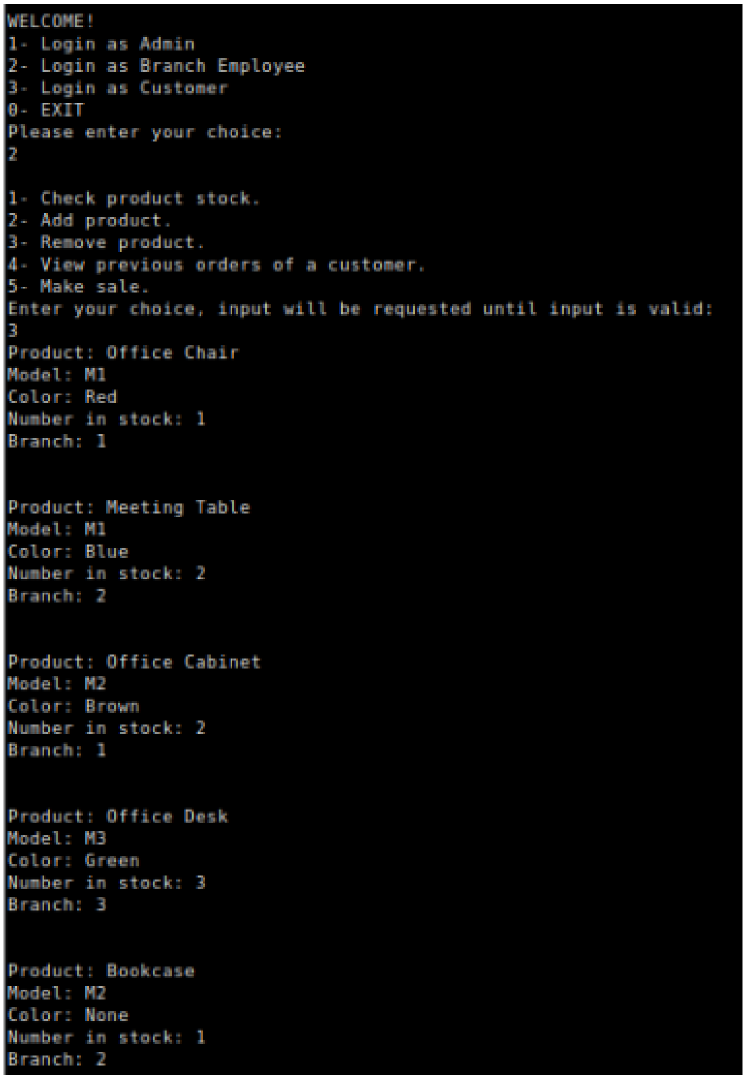


```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
2

1- Check product stock.
2- Add product.
3- Remove product.
4- View previous orders of a customer.
5- Make sale.
Enter your choice, input will be requested until input is valid:
2
Product: Office Chair
Model: M1
Color: Red
Number in stock: 1
Branch: 1


Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 1
Branch: 2



Please enter product name:
Bookcase
Please enter model name:
M2
Please enter color:
None
```

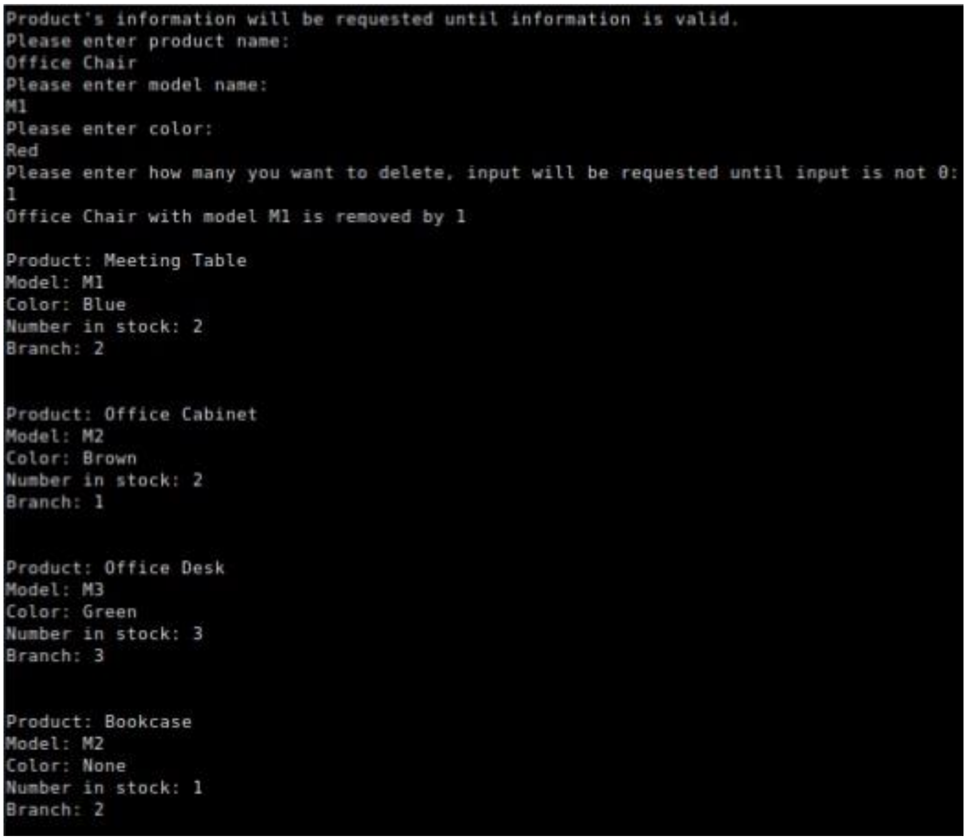


```
Please enter product name:
Bookcase
Please enter model name:
M2
Please enter color:
None
Please enter how many you want to add, input will be requested until input is valid:
2
Product: Office Chair
Model: M1
Color: Red
Number in stock: 1
Branch: 1


Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 3
Branch: 2
```

9) Remove product (Branch Employee)

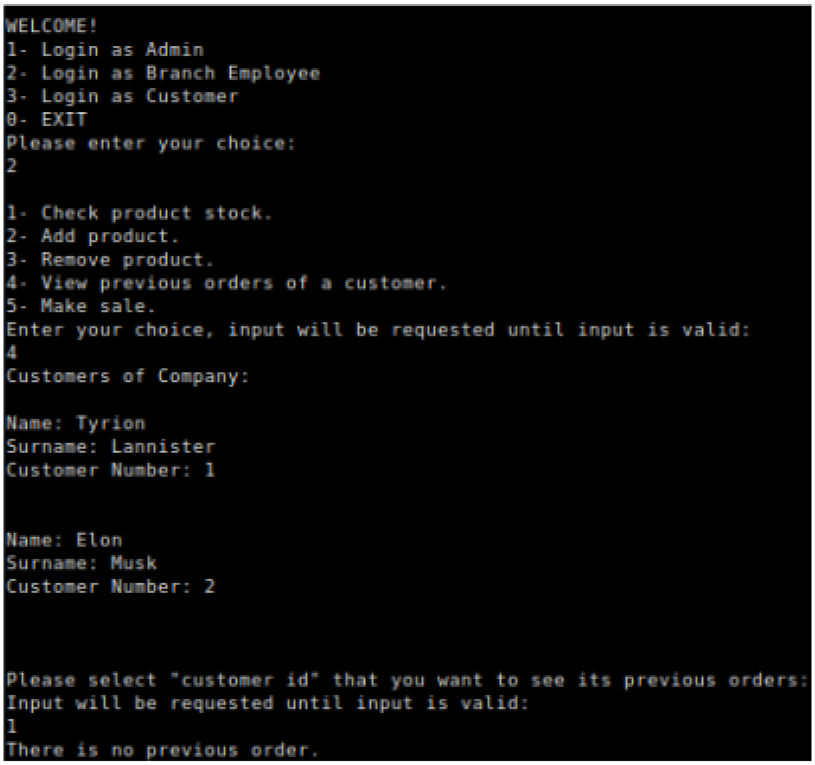


```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
2

1- Check product stock.
2- Add product.
3- Remove product.
4- View previous orders of a customer.
5- Make sale.
Enter your choice, input will be requested until input is valid:
3
Product: Office Chair
Model: M1
Color: Red
Number in stock: 1
Branch: 1


Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 1
Branch: 2
```



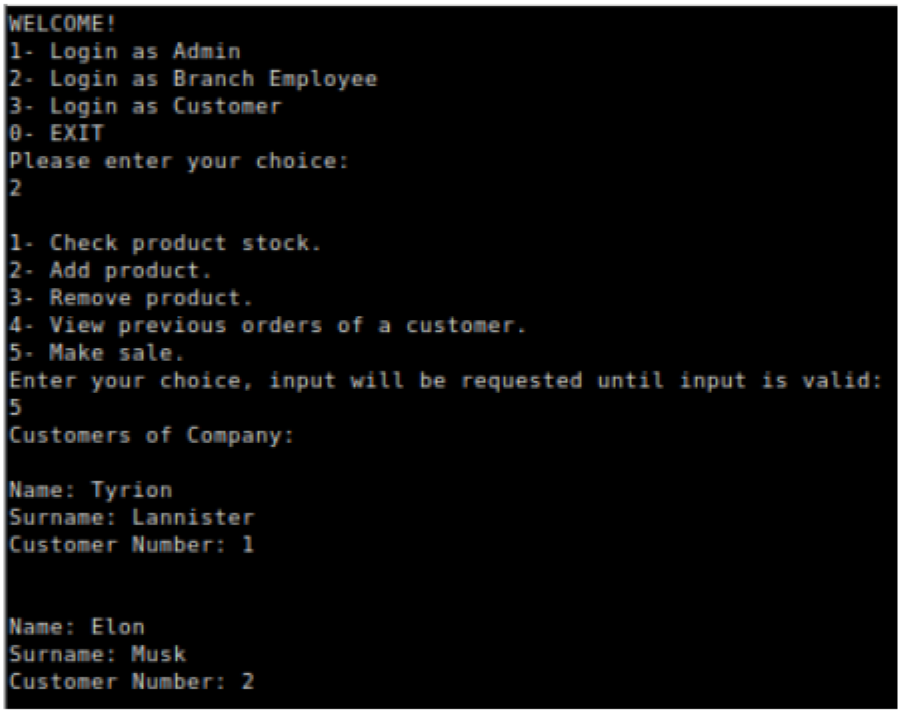
```
Product's information will be requested until information is valid.
Please enter product name:
Office Chair
Please enter model name:
M1
Please enter color:
Red
Please enter how many you want to delete, input will be requested until input is not 0:
1
Office Chair with model M1 is removed by 1

Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 1
Branch: 2
```

10) View previous orders of a customer (Branch Employee)

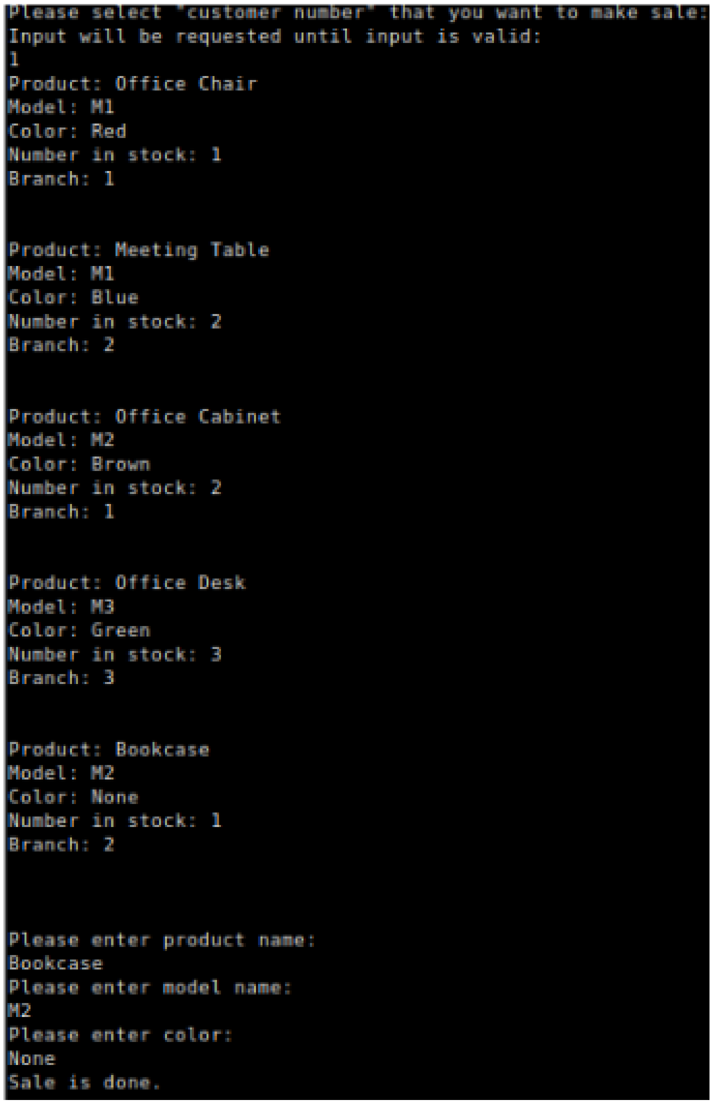


```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
2

1- Check product stock.
2- Add product.
3- Remove product.
4- View previous orders of a customer.
5- Make sale.
Enter your choice, input will be requested until input is valid:
4
Customers of Company:

Name: Tyrion
Surname: Lannister
Customer Number: 1


Name: Elon
Surname: Musk
Customer Number: 2



Please select "customer id" that you want to see its previous orders:
Input will be requested until input is valid:
1
There is no previous order.
```

11) Make sale (Branch Employee)



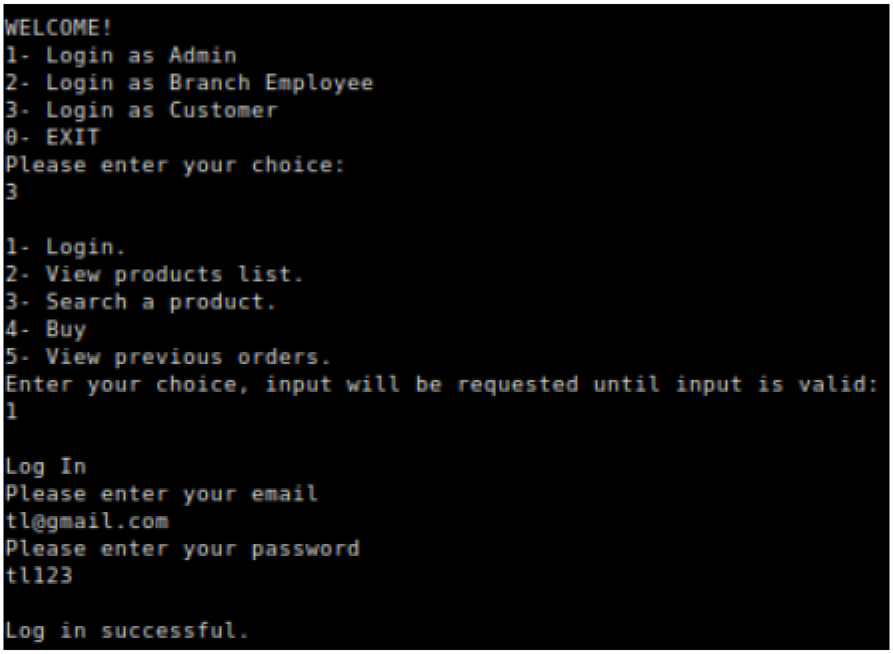
```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
2

1- Check product stock.
2- Add product.
3- Remove product.
4- View previous orders of a customer.
5- Make sale.
Enter your choice, input will be requested until input is valid:
5
Customers of Company:

Name: Tyrion
Surname: Lannister
Customer Number: 1


Name: Elon
Surname: Musk
Customer Number: 2
```



```
Please select "customer number" that you want to make sale:
Input will be requested until input is valid:
1
Product: Office Chair
Model: M1
Color: Red
Number in stock: 1
Branch: 1


Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 1
Branch: 2



Please enter product name:
Bookcase
Please enter model name:
M2
Please enter color:
None
Sale is done.
```

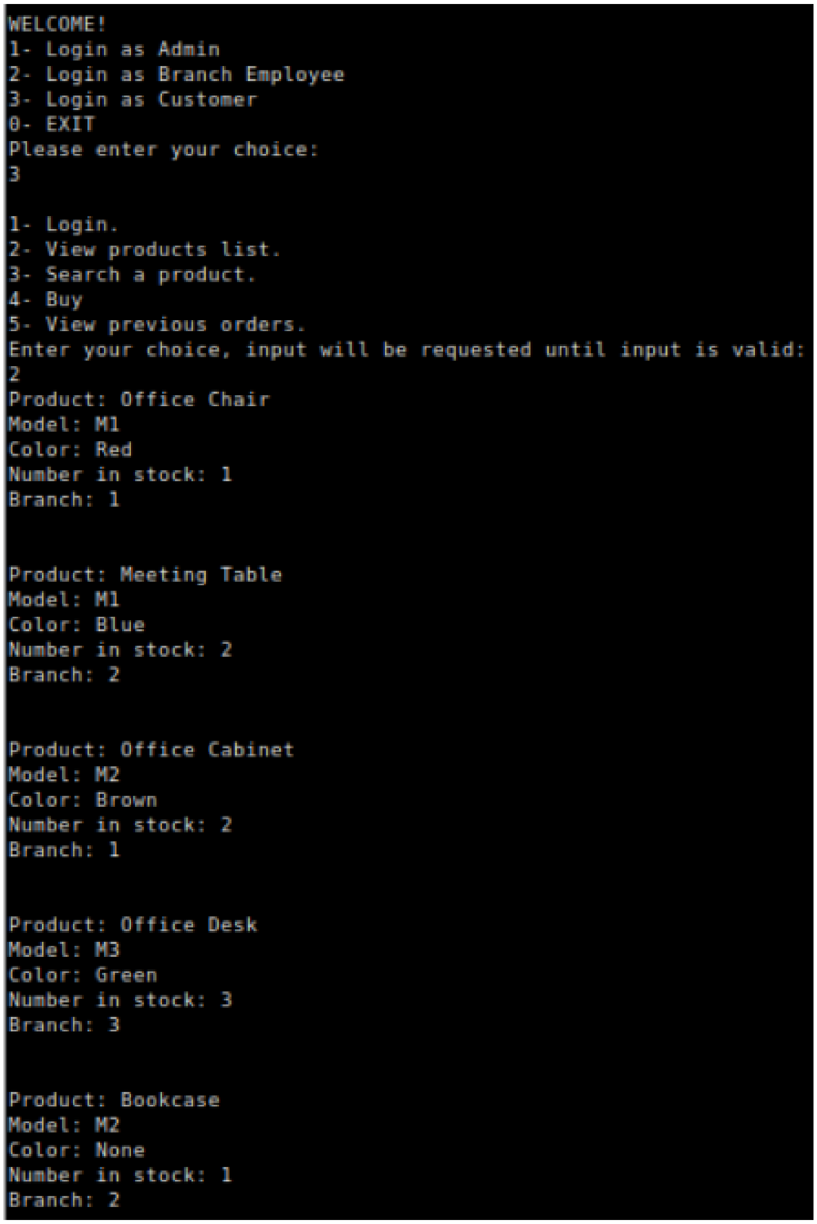
12) Login (Customer)

(If a customer logs in, he/she remains logged in until new login is done by different person)



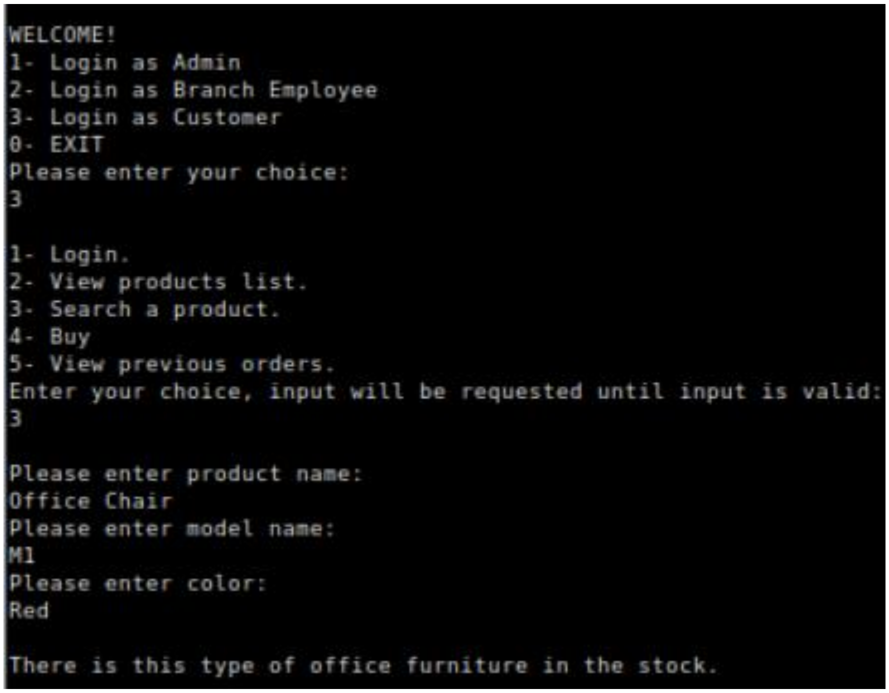
```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
1

Log In
Please enter your email
tl@gmail.com
Please enter your password
tl123

Log in successful.
```

13) View product list (Customer)



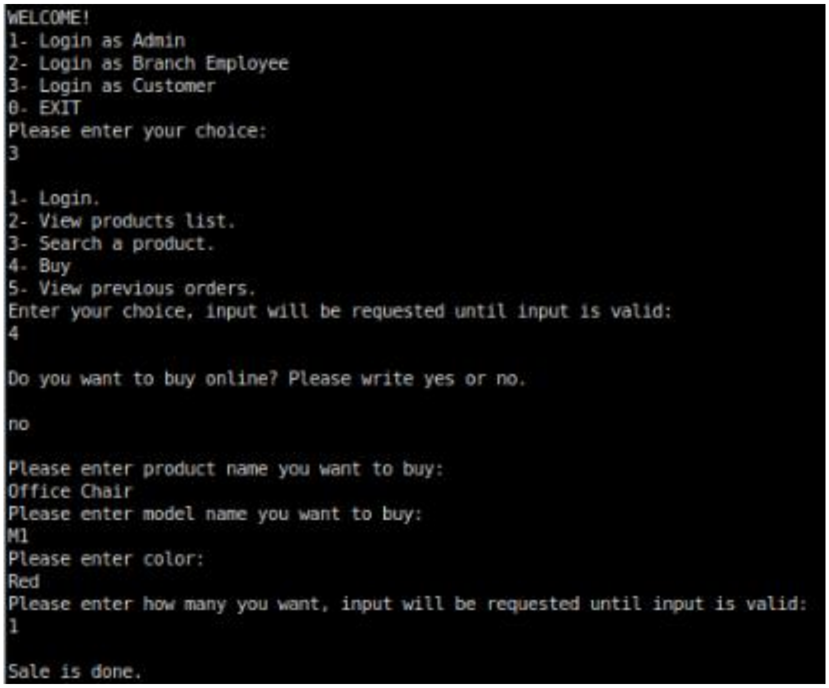
```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
2
Product: Office Chair
Model: M1
Color: Red
Number in stock: 1
Branch: 1


Product: Meeting Table
Model: M1
Color: Blue
Number in stock: 2
Branch: 2


Product: Office Cabinet
Model: M2
Color: Brown
Number in stock: 2
Branch: 1


Product: Office Desk
Model: M3
Color: Green
Number in stock: 3
Branch: 3


Product: Bookcase
Model: M2
Color: None
Number in stock: 1
Branch: 2
```

14) Search a product (Customer)



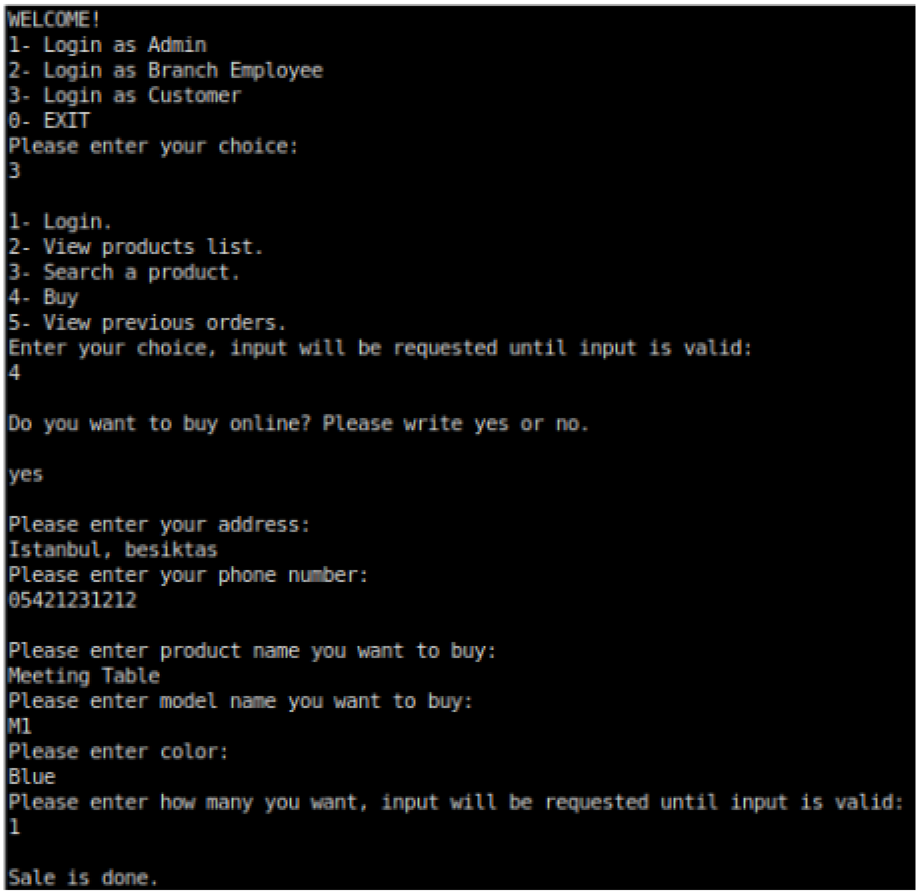
```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
3

Please enter product name:
Office Chair
Please enter model name:
M1
Please enter color:
Red

There is this type of office furniture in the stock.
```

15) Buy in store (Customer)



```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
4

Do you want to buy online? Please write yes or no.

no

Please enter product name you want to buy:
Office Chair
Please enter model name you want to buy:
M1
Please enter color:
Red
Please enter how many you want, input will be requested until input is valid:
1

Sale is done.
```

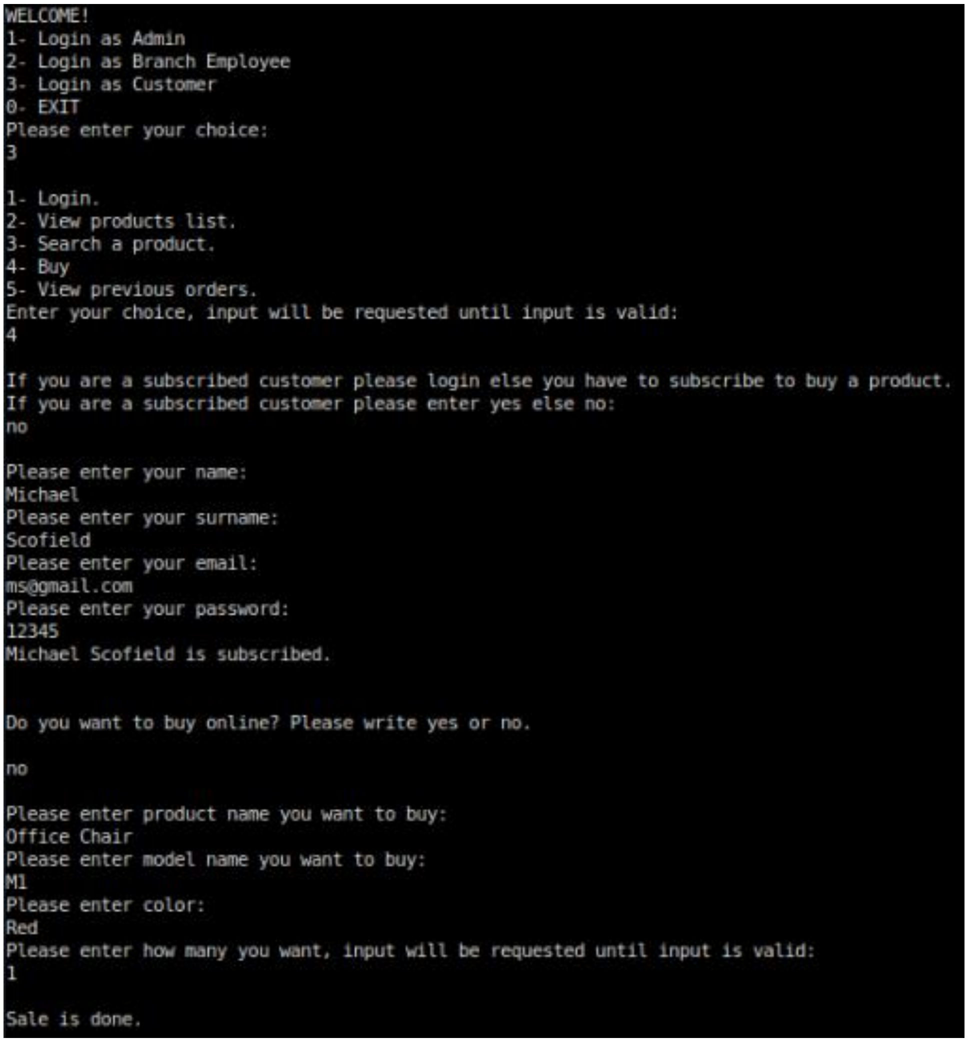
16) Buy online (Customer)



```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
4

Do you want to buy online? Please write yes or no.

yes

Please enter your address:
Istanbul, besiktas
Please enter your phone number:
05421231212

Please enter product name you want to buy:
Meeting Table
Please enter model name you want to buy:
M1
Please enter color:
Blue
Please enter how many you want, input will be requested until input is valid:
1

Sale is done.
```

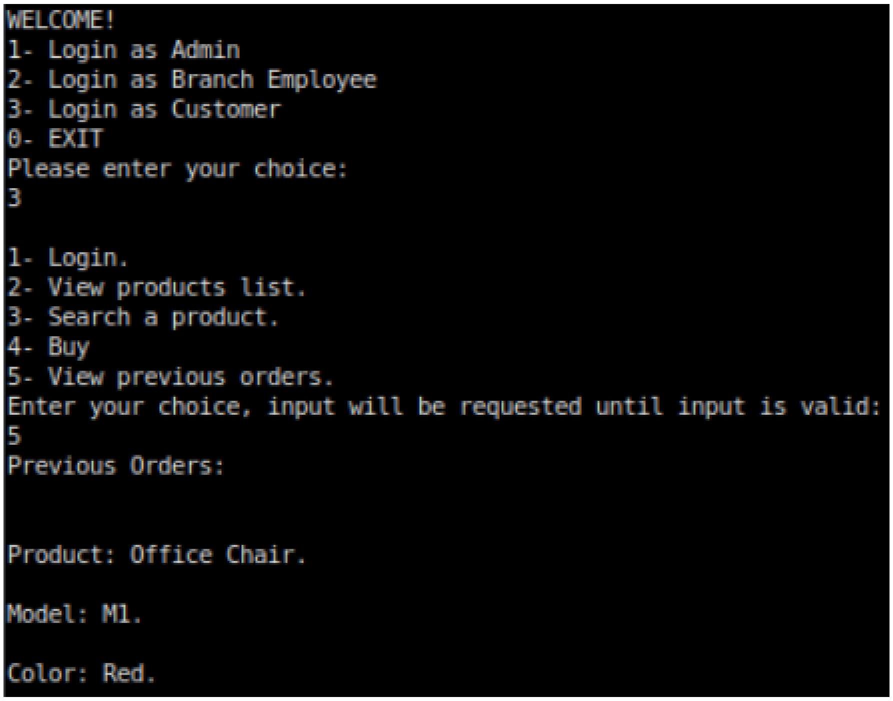
17) Creating subscription at first buying (Customer)

(If someone subscribes then he/she remains logged in until different person logs in)



```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
4

If you are a subscribed customer please login else you have to subscribe to buy a product.
If you are a subscribed customer please enter yes else no:
no

Please enter your name:
Michael
Please enter your surname:
Scofield
Please enter your email:
ms@gmail.com
Please enter your password:
12345
Michael Scofield is subscribed.


Do you want to buy online? Please write yes or no.

no

Please enter product name you want to buy:
Office Chair
Please enter model name you want to buy:
M1
Please enter color:
Red
Please enter how many you want, input will be requested until input is valid:
1

Sale is done.
```

18) View previous orders (Customer)

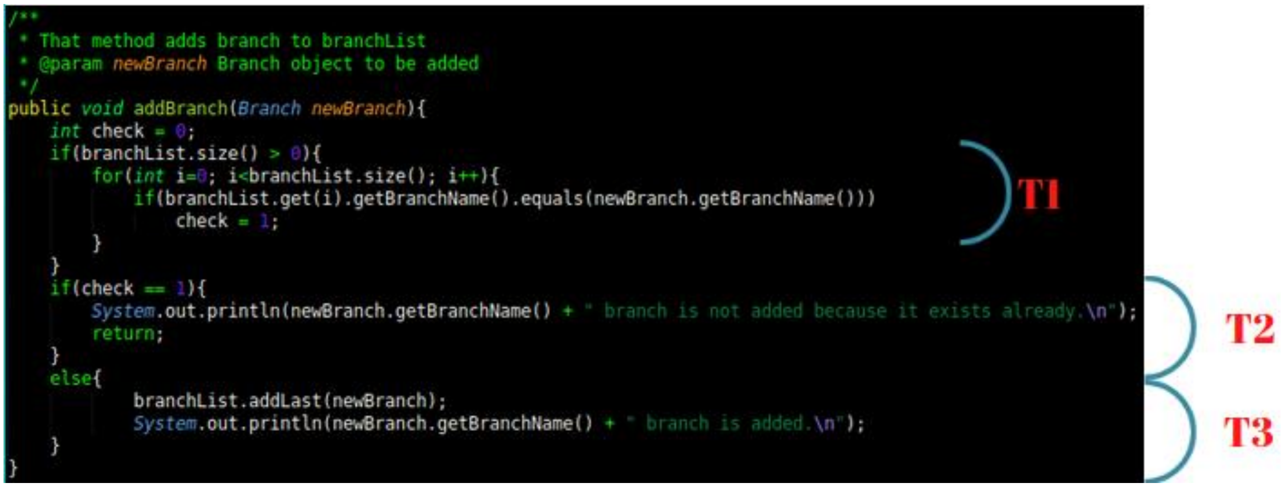


```
WELCOME!
1- Login as Admin
2- Login as Branch Employee
3- Login as Customer
0- EXIT
Please enter your choice:
3

1- Login.
2- View products list.
3- Search a product.
4- Buy
5- View previous orders.
Enter your choice, input will be requested until input is valid:
5
Previous Orders:


Product: Office Chair.

Model: M1.

Color: Red.
```

PART2 –ANALYSIS OF METHODS

ADMINISTRATOR CLASS:

addBranch Method:



```
/**
 * That method adds branch to branchList
 * @param newBranch Branch object to be added
 */
public void addBranch(Branch newBranch){
    int check = 0;
    if(branchList.size() > 0){
        for(int i=0; i<branchList.size(); i++){
            if(branchList.get(i).getBranchName().equals(newBranch.getBranchName()))
                check = 1;
        }
    }
    if(check == 1){
        System.out.println(newBranch.getBranchName() + " branch is not added because it exists already.\n");
        return;
    }
    else{
            branchList.addLast(newBranch);
            System.out.println(newBranch.getBranchName() + " branch is added.\n");
    }
}
```

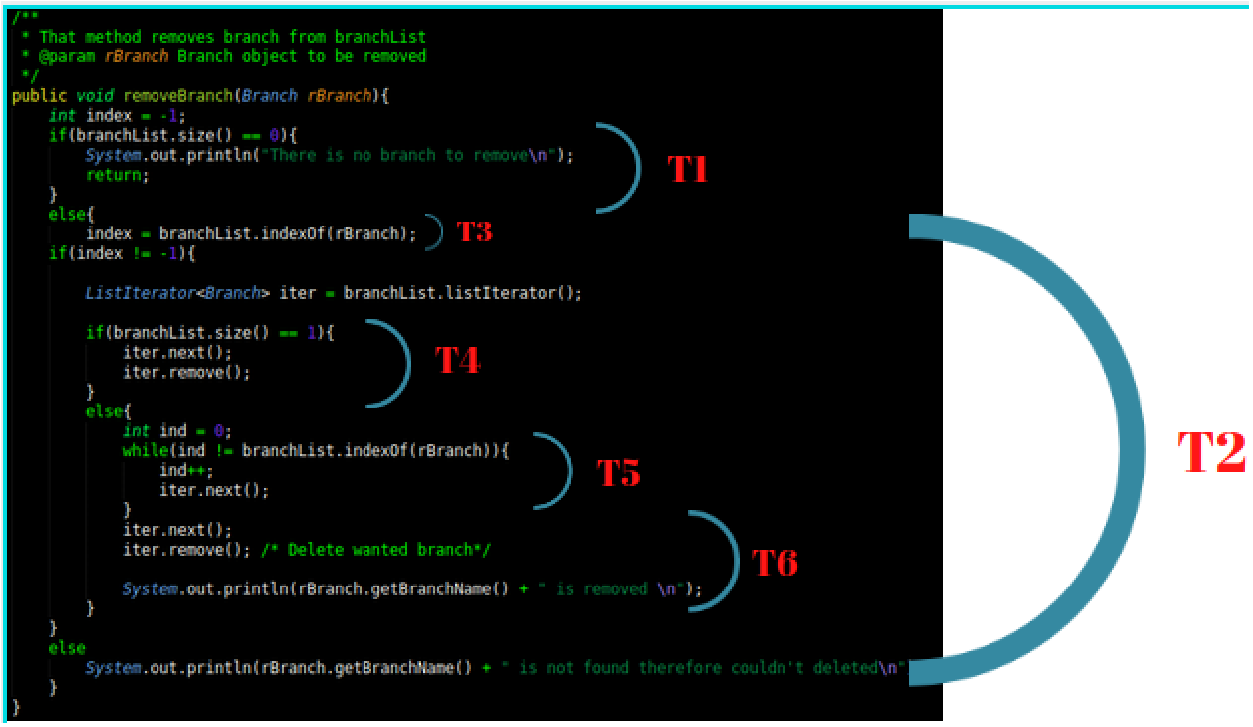
T1(N) = θ(n)(from for loop) * (θ(1)(getter) + θ(1) (getter) + O(n)(equals method) = O(n^2)

T2(N) = θ(1)(if + getter)

T3(N) = θ(1)(addLast + getter)

T(N) = T1(N) + θ(1) (T2 or T3) = O(n^2)

removeBranch Method:



```
/**
 * That method removes branch from branchList
 * @param rBranch Branch object to be removed
 */
public void removeBranch(Branch rBranch){
    int index = -1;
    if(branchList.size() == 0){
        System.out.println("There is no branch to remove\n");
        return;
    }
    else{
        index = branchList.indexOf(rBranch);
    if(index != -1){

        ListIterator<Branch> iter = branchList.listIterator();

        if(branchList.size() == 1){
            iter.next();
            iter.remove();
        }
        else{
            int ind = 0;
            while(ind != branchList.indexOf(rBranch)){
                ind++;
                iter.next();
            }
            iter.next();
            iter.remove(); /* Delete wanted branch*/

            System.out.println(rBranch.getBranchName() + " is removed \n");
        }
    }
    else
        System.out.println(rBranch.getBranchName() + " is not found therefore couldn't deleted\n"
    }
}
```

T1(N) = $\theta(1)$

T3(N) = O(n) because it goes through the list

T4(N) = $\theta(1) + \theta(1) + \theta(1)$ because complexity of if is $\theta(1)$ and complexities of iter.next and iter.remove are $\theta(1)$ thus $\theta(1)$.

T5(N) -> **Best case =** If object is found at index 0 then complexity is indexOf's complexity which is O(n)

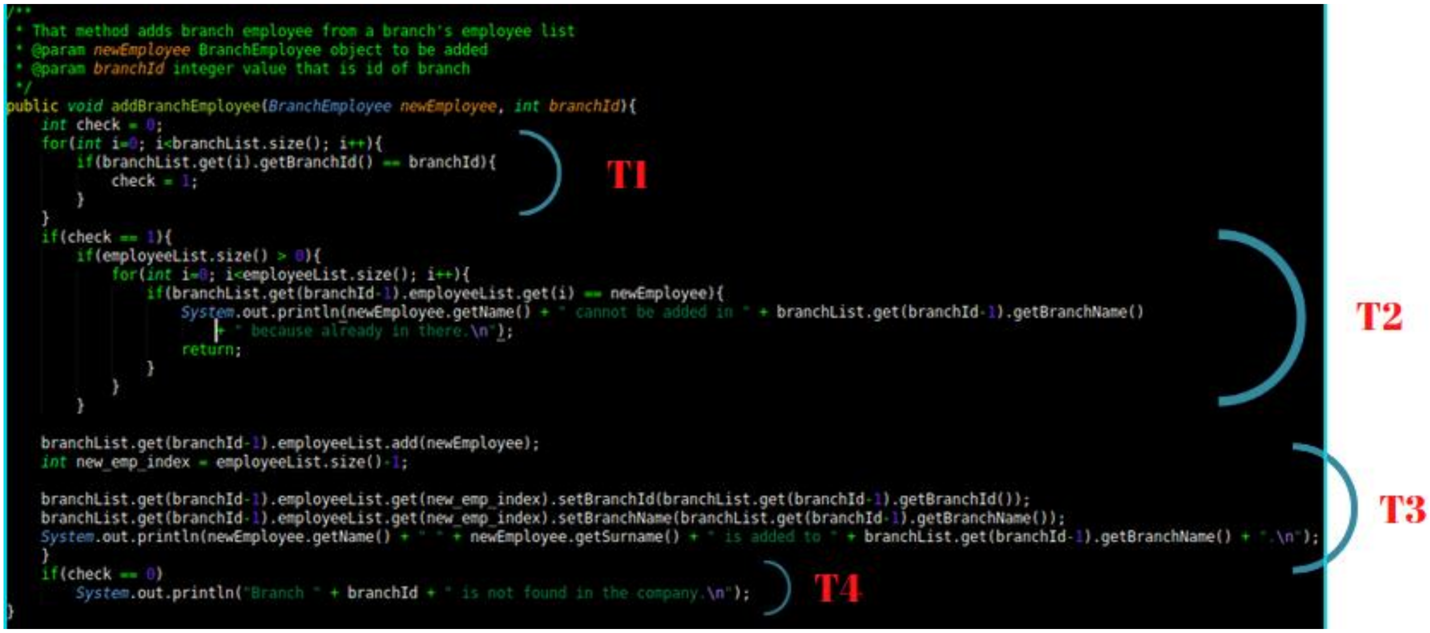
Worst case = O(n) * O(n) = $O(n^2)$ because of indexOf's complexity is O(n) and while's complexity is O(n)

T6(N) = $\theta(1)$

T2(N) = T3 + T4 + T5 + T6 -> **Best case** = O(n), **Worst case =** $O(n^2)$ because of T5

T(N) = $O(n^2)$ -> Best case = T1(N) = $\theta(1)$, Worst case = Worst case of T2(N) = $O(n^2)$

addBranchEmployee Method:



```java
/**
 * That method adds branch employee from a branch's employee list
 * @param newEmployee BranchEmployee object to be added
 * @param branchId integer value that is id of branch
 */
public void addBranchEmployee(BranchEmployee newEmployee, int branchId){
    int check = 0;
    for(int i=0; i<branchList.size(); i++){
        if(branchList.get(i).getBranchId() == branchId){
            check = 1;
        }
    }
    if(check == 1){
        if(employeeList.size() > 0){
            for(int i=0; i<employeeList.size(); i++){
                if(branchList.get(branchId-1).employeeList.get(i) == newEmployee){
                    System.out.println(newEmployee.getName() + " cannot be added in " + branchList.get(branchId-1).getBranchName()
                        + " because already in there.\n");
                    return;
                }
            }
        }

    branchList.get(branchId-1).employeeList.add(newEmployee);
    int new_emp_index = employeeList.size()-1;

    branchList.get(branchId-1).employeeList.get(new_emp_index).setBranchId(branchList.get(branchId-1).getBranchId());
    branchList.get(branchId-1).employeeList.get(new_emp_index).setBranchName(branchList.get(branchId-1).getBranchName());
    System.out.println(newEmployee.getName() + " " + newEmployee.getSurname() + " is added to " + branchList.get(branchId-1).getBranchName() + ".\n");
    }
    if(check == 0)
        System.out.println("Branch " + branchId + " is not found in the company.\n");
}
```

T1(N) = θ(n)(from for loop) * O(n)(from linked list get method) = O(n^2)

T2(N) -> **Best case =** θ(1) when employee list's size lower than 0,

Worst case =

θ(n)(from for loop) * (O(n)(from linked list get method) + θ(1) (from arraylist get method))) * O(n)(from linked list get method ,this is while printing) = O(n^3)

T3(N) = O(n)(from linked list get method) +

(O(n)(from linked list get method) + θ(1) (from arraylist get method))) +

(O(n)(from linked list get method) + θ(1) (from arraylist get method))) +

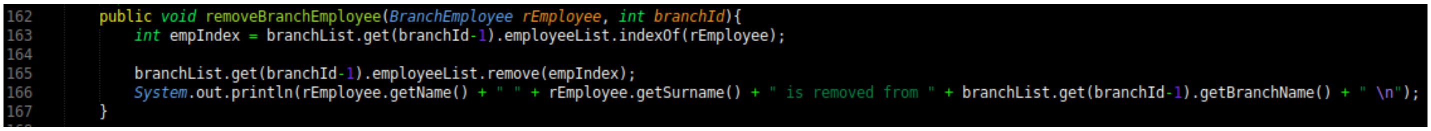
O(n)(from linked list get method) = O(n)

T4(N) = θ(1)

T(N) = O(n^3) -> Best Case = T1(N) + T2(N)(Best case) + T3(N) = O(n^2)

Worst Case = T1(N) + T2(N)(Worst case) + T3(N) + T4(N) = O(n^3)

removeBranchEmployee Method:



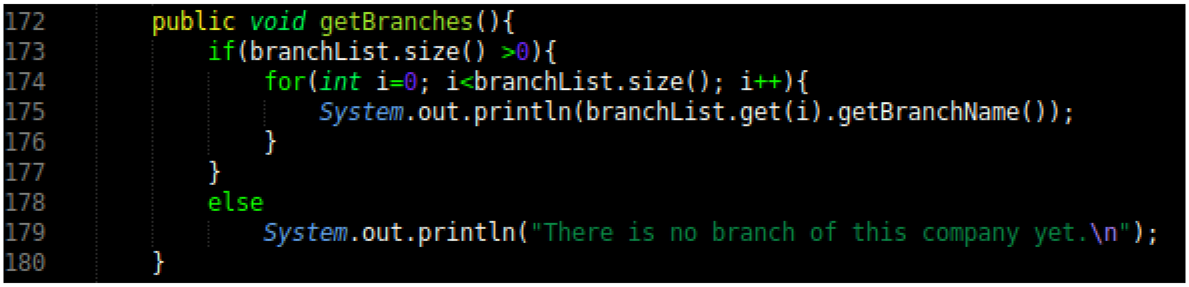
```
162        public void removeBranchEmployee(BranchEmployee rEmployee, int branchId){
163            int empIndex = branchList.get(branchId-1).employeeList.indexOf(rEmployee);
164
165            branchList.get(branchId-1).employeeList.remove(empIndex);
166            System.out.println(rEmployee.getName() + " " + rEmployee.getSurname() + " is removed from " + branchList.get(branchId-1).getBranchName() + " \n");
167        }
```

T(n) = O(n)(from linked list get method) + O(n)(from array list indexOf method) (line 163)

+ O(n)(from linked list get method) + O(n)(from array list remove method) (line 165)

+ O(n)(from linked list get method) = O(n)

getBranches Method:

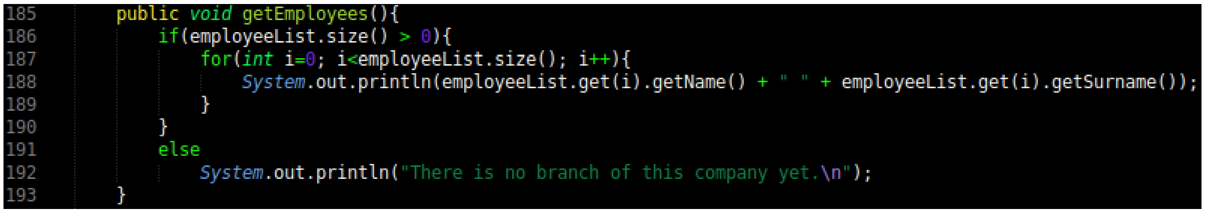


```
    public void getBranches(){
        if(branchList.size() >0){
            for(int i=0; i<branchList.size(); i++){
                System.out.println(branchList.get(i).getBranchName());
            }
        }
        else
            System.out.println("There is no branch of this company yet.\n");
    }
```

T(n) = $O(n^2)$ **-> Best Case =** θ(1) (else condition)

Worst Case = θ(n)(from for loop) * O(n)(from linked list get method) = $O(n^2)$

getEmployees Method:

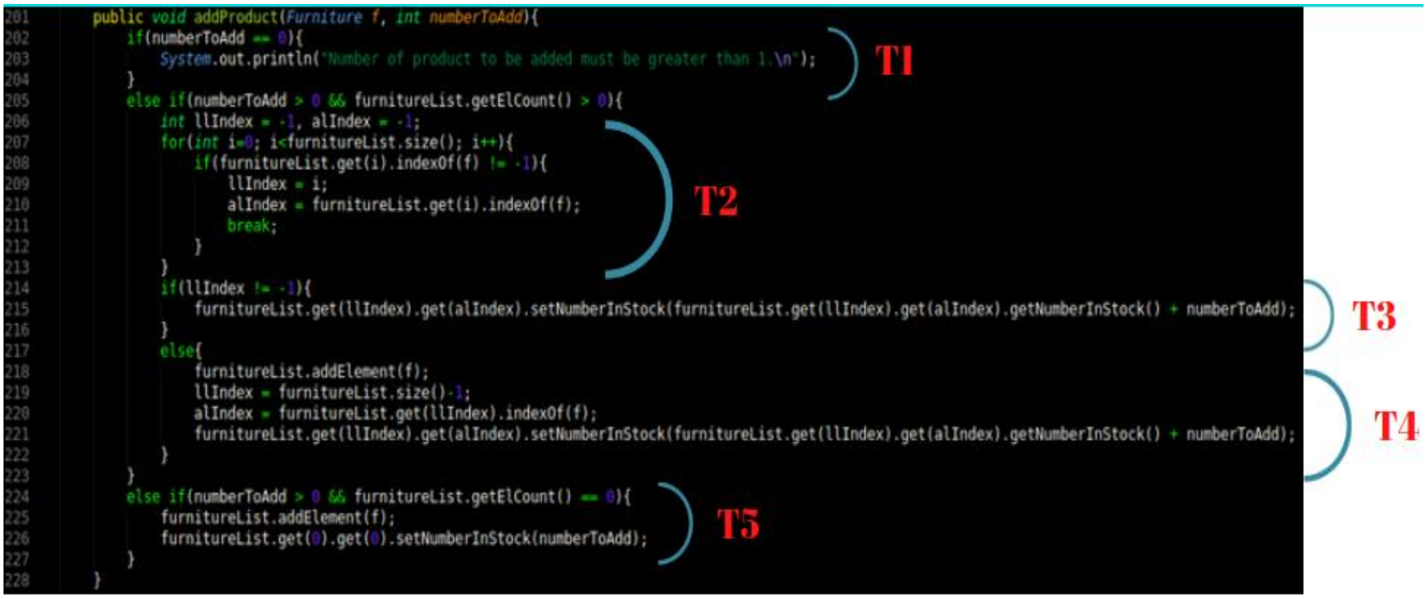


```
    public void getEmployees(){
        if(employeeList.size() > 0){
            for(int i=0; i<employeeList.size(); i++){
                System.out.println(employeeList.get(i).getName() + " " + employeeList.get(i).getSurname());
            }
        }
        else
            System.out.println("There is no branch of this company yet.\n");
    }
```

T(n) = θ(n) **-> Best Case =** θ(1) (else condition)

Worst Case = θ(n)(from for loop) * θ(1)(from array list get method) = θ(n)

addProduct Method:



```java
public void addProduct(Furniture f, int numberToAdd){
    if(numberToAdd == 0){
        System.out.println("Number of product to be added must be greater than 1.\n");
    }
    else if(numberToAdd > 0 && furnitureList.getElCount() > 0){
        int llIndex = -1, alIndex = -1;
        for(int i=0; i<furnitureList.size(); i++){
            if(furnitureList.get(i).indexOf(f) != -1){
                llIndex = i;
                alIndex = furnitureList.get(i).indexOf(f);
                break;
            }
        }
        if(llIndex != -1){
            furnitureList.get(llIndex).get(alIndex).setNumberInStock(furnitureList.get(llIndex).get(alIndex).getNumberInStock() + numberToAdd);
        }
        else{
            furnitureList.addElement(f);
            llIndex = furnitureList.size()-1;
            alIndex = furnitureList.get(llIndex).indexOf(f);
            furnitureList.get(llIndex).get(alIndex).setNumberInStock(furnitureList.get(llIndex).get(alIndex).getNumberInStock() + numberToAdd);
        }
    }
    else if(numberToAdd > 0 && furnitureList.getElCount() == 0){
        furnitureList.addElement(f);
        furnitureList.get(0).get(0).setNumberInStock(numberToAdd);
    }
}
```

T1(N) = θ(1)

T2(N) -> Best case = θ(1)(from for loop) *

(O(n)(from linked list get method) + O(n)(from array list indexOf method) +

O(n)(from linked list get method) + O(n)(from array list indexOf method))

= O(n)

Worst case = θ(n)(from for loop) *

(O(n)(from linked list get method) + O(n)(from array list indexOf method) +

O(n)(from linked list get method) + O(n)(from array list indexOf method))

= $O(n^2)$

T3(N) = O(n)(from linked list get method) + θ(1) (from array list get method) = O(n)

T4(N) = θ(1)(addElement method) +

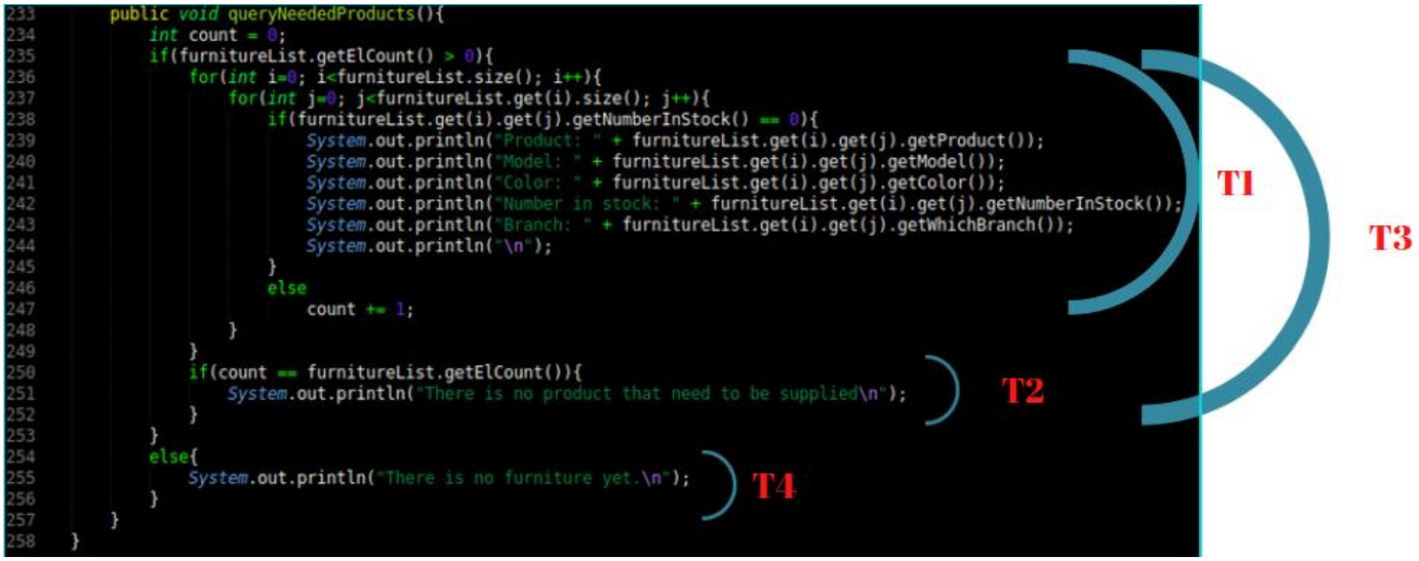
(O(n)(from linked list get method) + O(n)(from array list indexOf method) +

(O(n)(from linked list get method) + O(n)(from array list indexOf method) = O(n)

T5(N) = θ(1)(addElement method) + θ(1)(array list get(0) and linked list get(0)) = θ(1)

T(N) = $O(n^2)$ **-> Best case = T1(N) =** θ(1), **Worst Case = T2(N)(Worst case) + T3(N) + T4(N) + T5(N) =** $O(n^2)$

queryNeededProducts Method:



```java
    public void queryNeededProducts(){
        int count = 0;
        if(furnitureList.getElCount() > 0){
            for(int i=0; i<furnitureList.size(); i++){
                for(int j=0; j<furnitureList.get(i).size(); j++){
                    if(furnitureList.get(i).get(j).getNumberInStock() == 0){
                        System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                        System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                        System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                        System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                        System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                        System.out.println("\n");
                    }
                    else
                        count += 1;
                }
            }
            if(count == furnitureList.getElCount()){
                System.out.println("There is no product that need to be supplied\n");
            }
        }
        else{
            System.out.println("There is no furniture yet.\n");
        }
    }
}
```

T1(N) = θ(n)(from outer for loop) * θ(n)(from inner for loop) *

((O(n)(from linked list get method) + O(n)(from array list indexOf method) +

5 * (O(n)(from linked list get method) + O(n)(from array list indexOf method)) = $O(n^3)$

***Here there is best&worst case because condition in if condition makes complexity** $O(n^3)$ **independently from whether if or else condition is valid.**

T2(N) = θ(1)

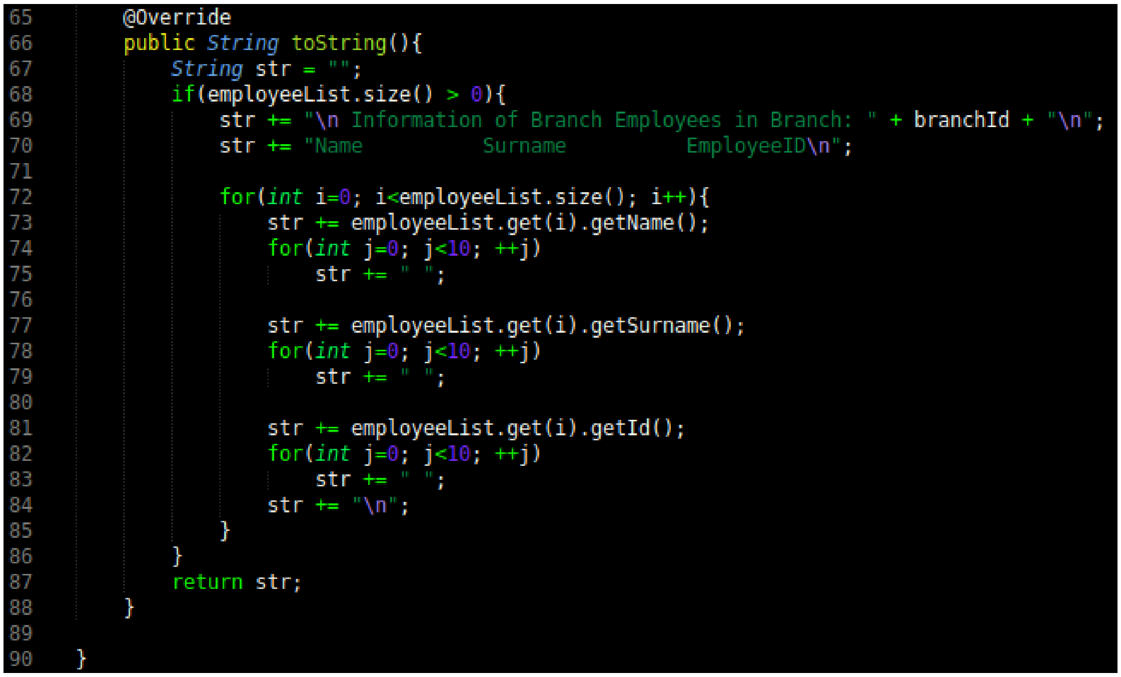
T3(N) = T1(N) + T2(N) = $O(n^3)$

T4(N) = θ(1)

T(N) = $O(n^3)$ **-> Best case = T4(N) =** θ(1), **Worst case = T3(N) =** $O(n^3)$

BRANCH CLASS

toString Method:



```
    @Override
    public String toString(){
        String str = "";
        if(employeeList.size() > 0){
            str += "\n Information of Branch Employees in Branch: " + branchId + "\n";
            str += "Name          Surname          EmployeeID\n";

            for(int i=0; i<employeeList.size(); i++){
                str += employeeList.get(i).getName();
                for(int j=0; j<10; ++j)
                    str += " ";

                str += employeeList.get(i).getSurname();
                for(int j=0; j<10; ++j)
                    str += " ";

                str += employeeList.get(i).getId();
                for(int j=0; j<10; ++j)
                    str += " ";
                str += "\n";
            }
        }
        return str;
    }

}
```

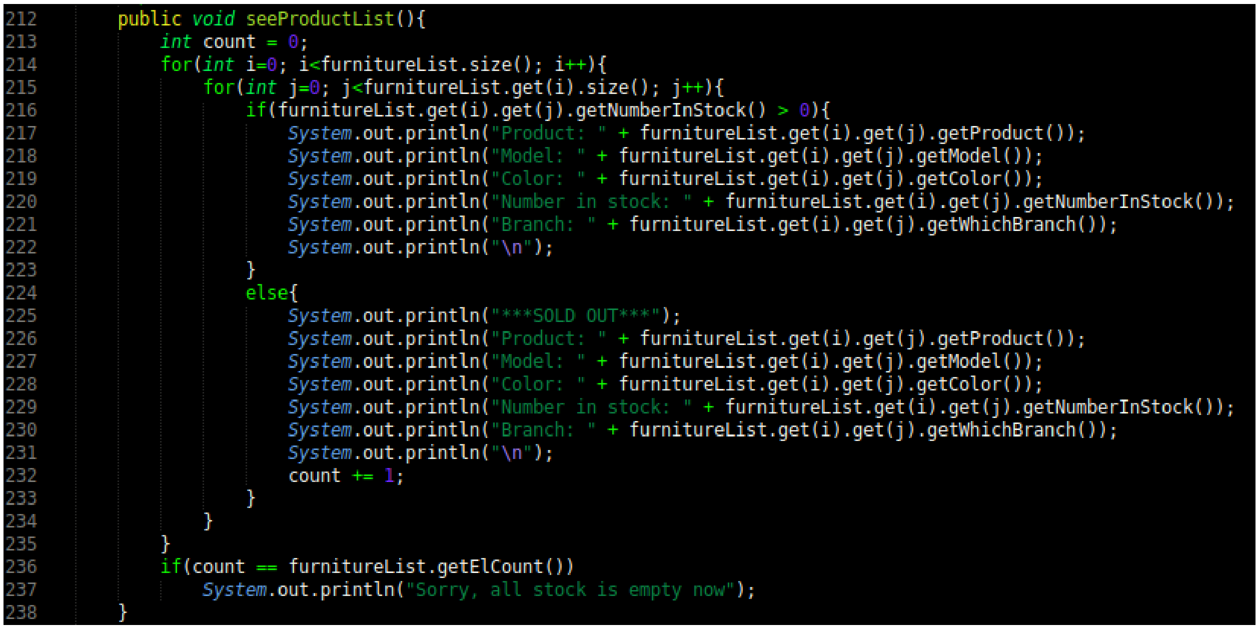
T(N) = θ(n)(for loop) * (O(n)(from linked list get method) + O(n)(from array list indexOf method) +

O(1)(for loop that turns for 10 times which is constant time) +

O(n)(Strings are immutable so it copies to new string)) = $O(n^2)$

CUSTOMER CLASS

seeProductList Method:



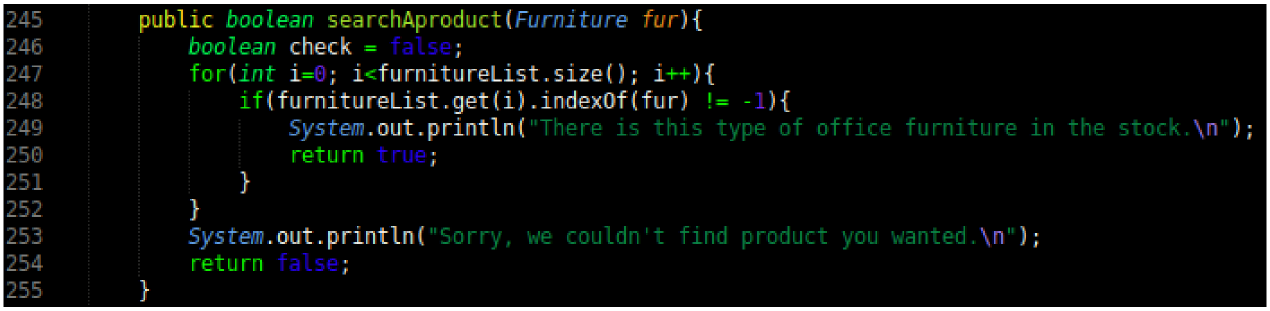
```java
public void seeProductList(){
    int count = 0;
    for(int i=0; i<furnitureList.size(); i++){
        for(int j=0; j<furnitureList.get(i).size(); j++){
            if(furnitureList.get(i).get(j).getNumberInStock() > 0){
                System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                System.out.println("\n");
            }
            else{
                System.out.println("***SOLD OUT***");
                System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                System.out.println("\n");
                count += 1;
            }
        }
    }
    if(count == furnitureList.getElCount())
        System.out.println("Sorry, all stock is empty now");
}
```

T(n) = θ(n)(from outer for loop) * θ(n)(from inner for loop) *

(O(n)(from linked list get method) + θ(1) (from array list get method)) + θ(1)(last if condition) = O(n^3)

***Here it doesn't matter whether first if or else condition is valid because condition in first if condition is O(n), bodies of first if and else are O(n) and O(n) + O(n) = O(n)**

searchAproduct Method:

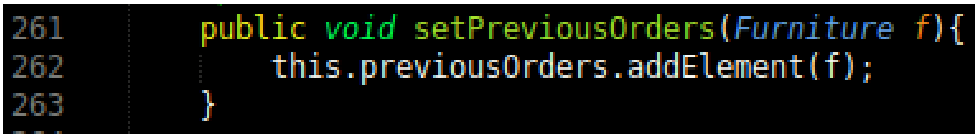


```java
public boolean searchAproduct(Furniture fur){
    boolean check = false;
    for(int i=0; i<furnitureList.size(); i++){
        if(furnitureList.get(i).indexOf(fur) != -1){
            System.out.println("There is this type of office furniture in the stock.\n");
            return true;
        }
    }
    System.out.println("Sorry, we couldn't find product you wanted.\n");
    return false;
}
```

T(N) = O(n^2) -> Best case = θ(1)(for loop) * (O(n)(from linked list get method) + O(n)(from array list indexOf method)) + θ(1)(print and return false) = O(n)

Worst case = θ(n)(for loop) * (O(n)(from linked list get method) + O(n)(from array list indexOf method)) + θ(1) (print and return false) = O(n^2)

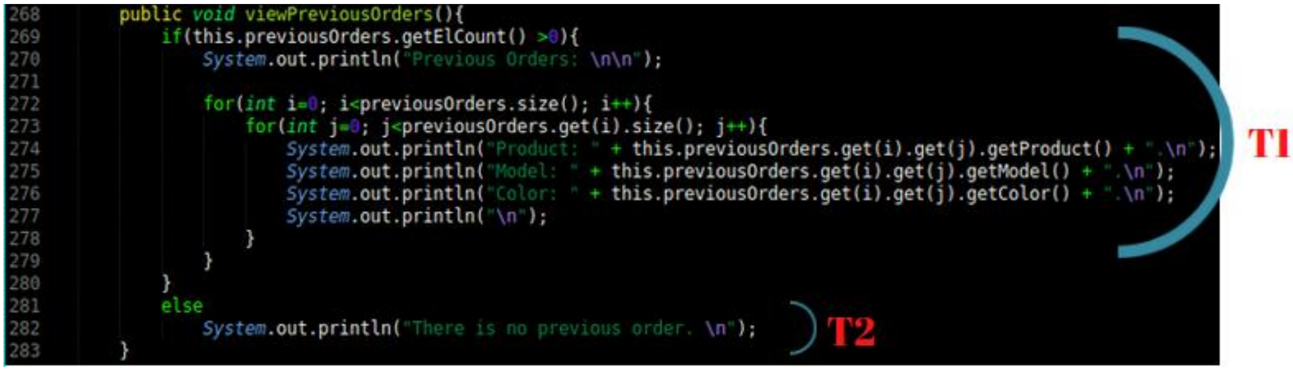
setPreviousOrders Method:



```
    public void setPreviousOrders(Furniture f){
        this.previousOrders.addElement(f);
    }
```

T(N) = θ(1)(addElement method's complexity)

viewPreviousOrders Method:



```
    public void viewPreviousOrders(){
        if(this.previousOrders.getElCount() >0){
            System.out.println("Previous Orders: \n\n");

            for(int i=0; i<previousOrders.size(); i++){
                for(int j=0; j<previousOrders.get(i).size(); j++){
                    System.out.println("Product: " + this.previousOrders.get(i).get(j).getProduct() + ".\n");
                    System.out.println("Model: " + this.previousOrders.get(i).get(j).getModel() + ".\n");
                    System.out.println("Color: " + this.previousOrders.get(i).get(j).getColor() + ".\n");
                    System.out.println("\n");
                }
            }
        }
        else
            System.out.println("There is no previous order. \n");
    }
```

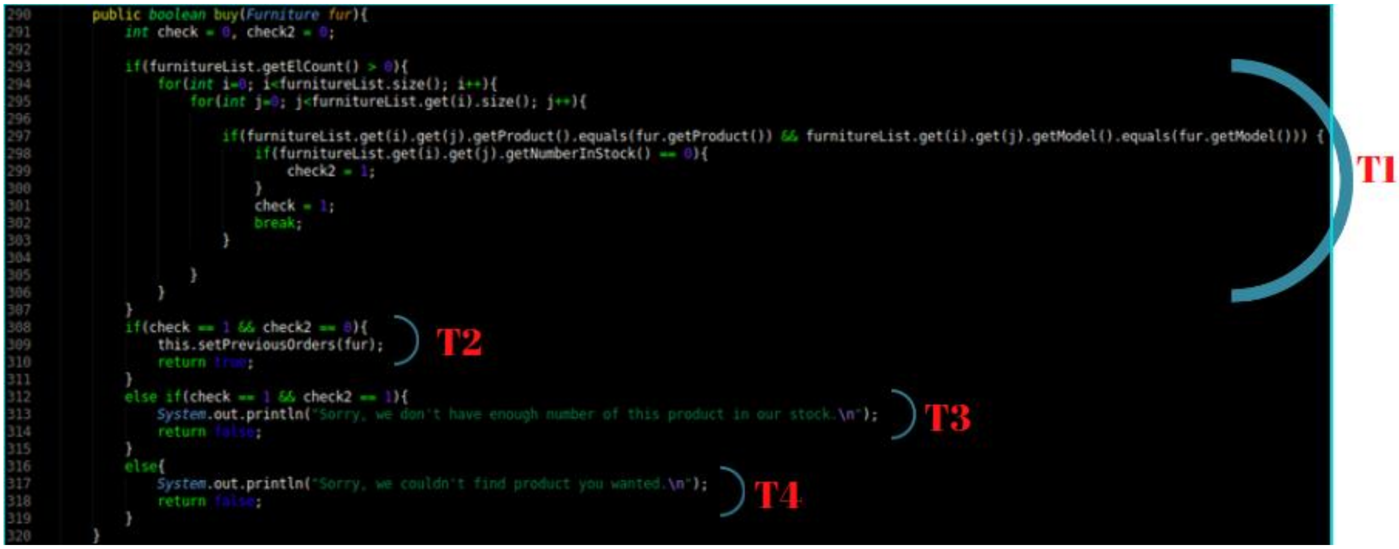
T1(N) = θ(n)(from outer for loop) * θ(n)(from inner for loop) *

(O(n)(from linked list get method) + θ(1) (from array list get method)) = O(n^3)

T2(N) = θ(1)

T(N) = O(n^3) -> Best case = T2(N) = θ(1), **Worst case = T1(N) =** O(n^3)

buy Method:



```java
    public boolean buy(Furniture fur){
        int check = 0, check2 = 0;

        if(furnitureList.getElCount() > 0){
            for(int i=0; i<furnitureList.size(); i++){
                for(int j=0; j<furnitureList.get(i).size(); j++){

                    if(furnitureList.get(i).get(j).getProduct().equals(fur.getProduct()) && furnitureList.get(i).get(j).getModel().equals(fur.getModel())) {
                        if(furnitureList.get(i).get(j).getNumberInStock() == 0){
                            check2 = 1;
                        }
                        check = 1;
                        break;
                    }

                }
            }
        }
        if(check == 1 && check2 == 0){
            this.setPreviousOrders(fur);
            return true;
        }
        else if(check == 1 && check2 == 1){
            System.out.println("Sorry, we don't have enough number of this product in our stock.\n");
            return false;
        }
        else{
            System.out.println("Sorry, we couldn't find product you wanted.\n");
            return false;
        }
    }
```

T1(N) = θ(n)(from outer for loop) * θ(n)(from inner for loop) *

(O(n)(from linked list get method) + θ(1) (from array list get method) + θ(1) + O(n)(equals method) +

O(n)(from linked list get method) + θ(1) (from array list get method)) = $O(n^3)$

T2(N) = θ(1)

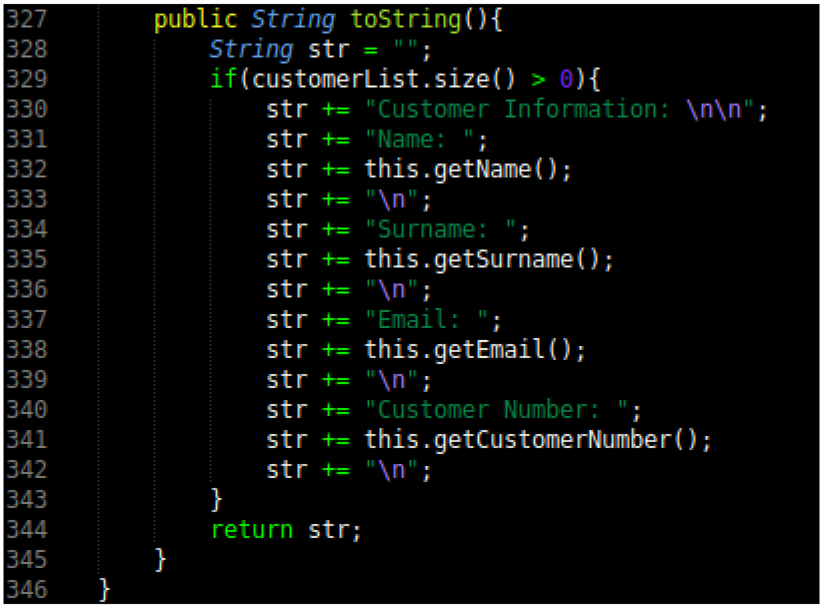
T3(N) = θ(1)

T4(N) = θ(1)

T(N) = $O(n^3)$ -> Best case = T2(N) or T3(N) or T4(N) = θ(1)

Worst case = T1(N) = $O(n^3)$

toString Method:

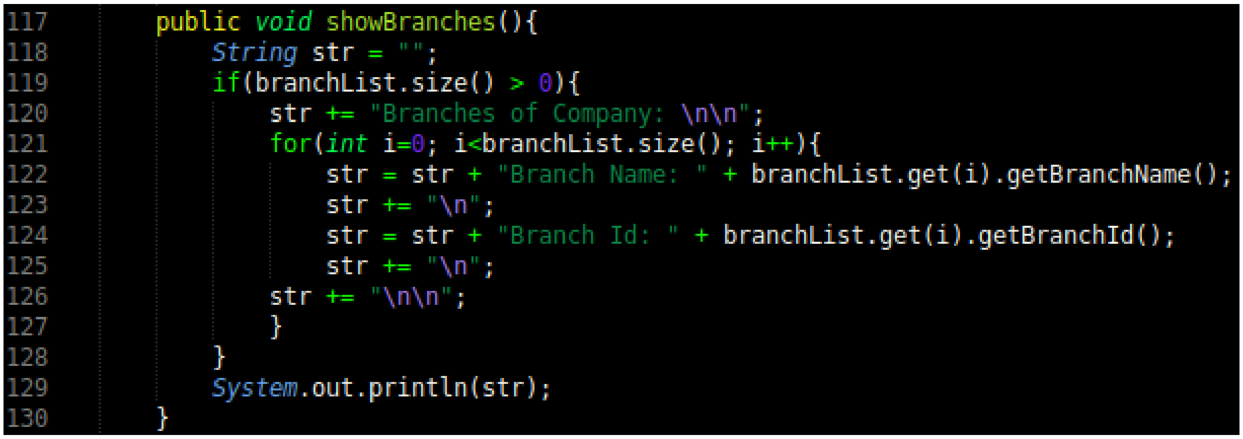


```
    public String toString(){
        String str = "";
        if(customerList.size() > 0){
            str += "Customer Information: \n\n";
            str += "Name: ";
            str += this.getName();
            str += "\n";
            str += "Surname: ";
            str += this.getSurname();
            str += "\n";
            str += "Email: ";
            str += this.getEmail();
            str += "\n";
            str += "Customer Number: ";
            str += this.getCustomerNumber();
            str += "\n";
        }
        return str;
    }
}
```

T(N) = θ(n)(for loop) * O(n)(String is immutable so string will be copied to new string) = O(n^2)

COMPANY CLASS

showBranches Method:

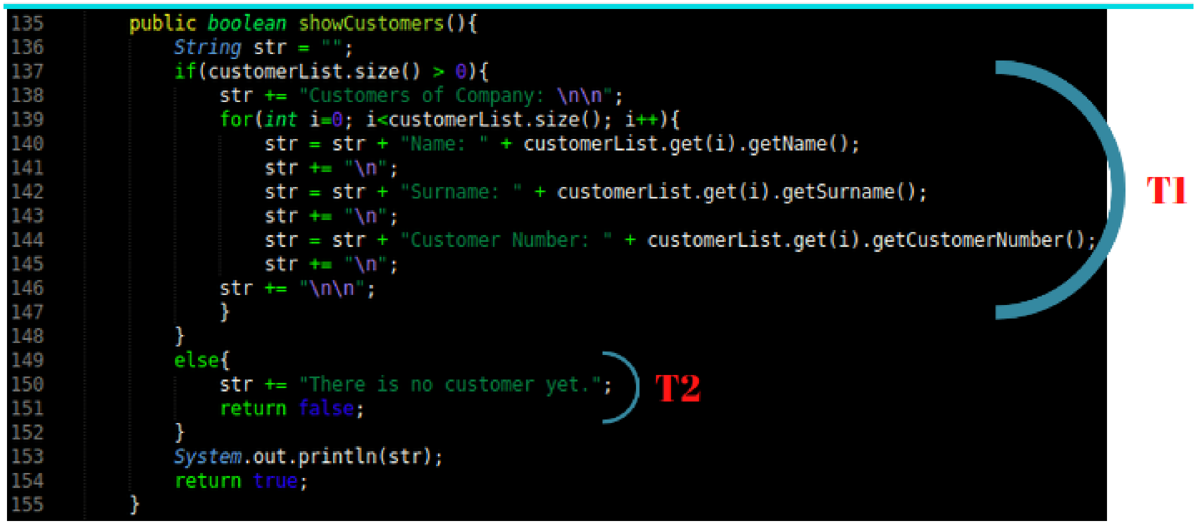


```
    public void showBranches(){
        String str = "";
        if(branchList.size() > 0){
            str += "Branches of Company: \n\n";
            for(int i=0; i<branchList.size(); i++){
                str = str + "Branch Name: " + branchList.get(i).getBranchName();
                str += "\n";
                str = str + "Branch Id: " + branchList.get(i).getBranchId();
                str += "\n";
            str += "\n\n";
            }
        }
        System.out.println(str);
    }
```

T(N) = O(n^2) -> Best case = θ(1) when brach list's size is lower than 0

Worst case = θ(n)(for loop) * O(n) (String is immutable so string will be copied to new string) = O(n^2)

ShowCustomers Method:



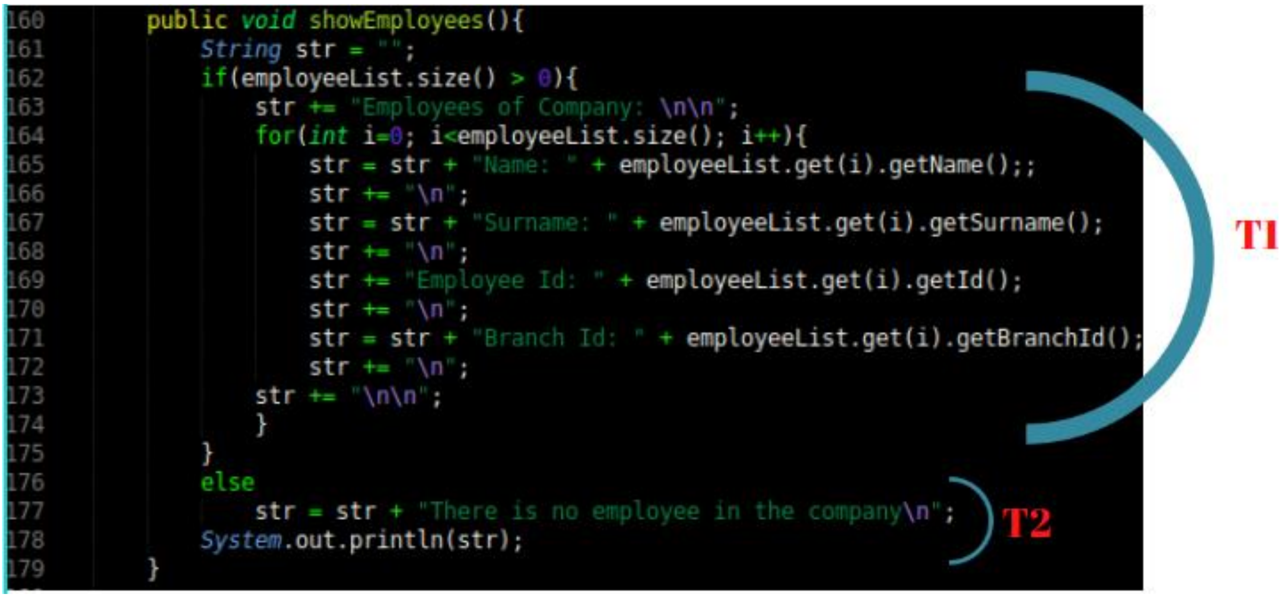
```
    public boolean showCustomers(){
        String str = "";
        if(customerList.size() > 0){
            str += "Customers of Company: \n\n";
            for(int i=0; i<customerList.size(); i++){
                str = str + "Name: " + customerList.get(i).getName();
                str += "\n";
                str = str + "Surname: " + customerList.get(i).getSurname();
                str += "\n";
                str = str + "Customer Number: " + customerList.get(i).getCustomerNumber();
                str += "\n";
            str += "\n\n";
            }
        }
        else{
            str += "There is no customer yet.";
            return false;
        }
        System.out.println(str);
        return true;
    }
```

T1(N) = θ(n)(for loop) * O(n) (String is immutable so string will be copied to new string) = O(n^2)

T2(N) = θ(1)

T(N) = O(n^2) **-> Best case = T2(N) =** θ(1), **Worst case = T1(N) =** O(n^2)

showEmployees Method:



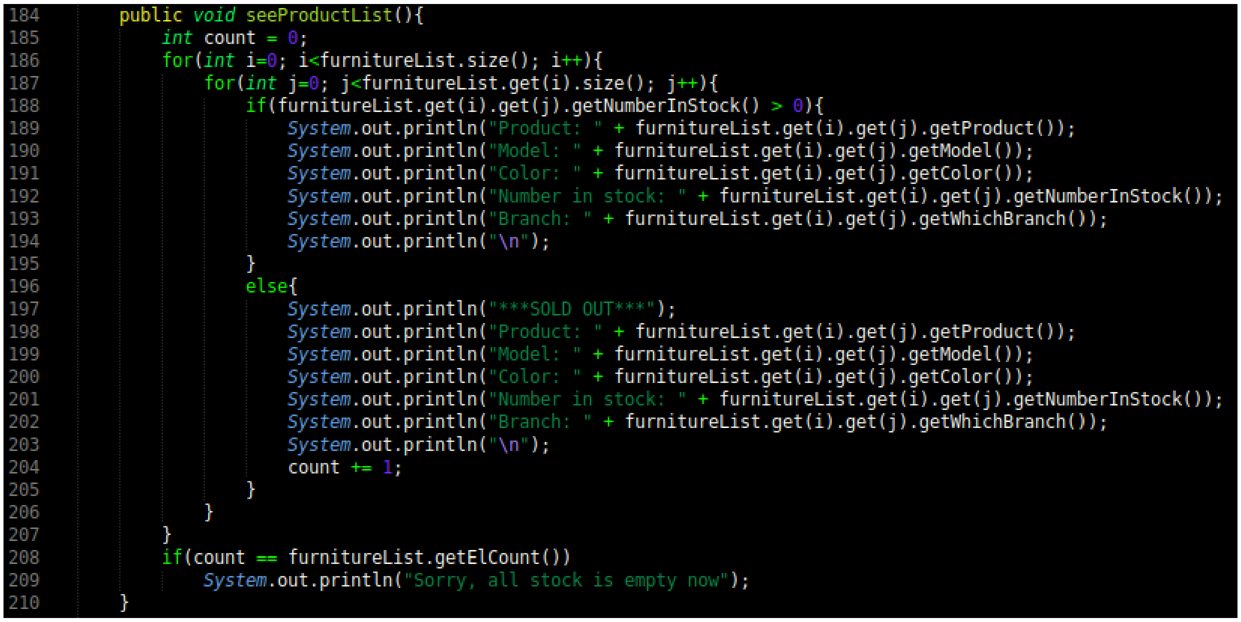
```java
    public void showEmployees(){
        String str = "";
        if(employeeList.size() > 0){
            str += "Employees of Company: \n\n";
            for(int i=0; i<employeeList.size(); i++){
                str = str + "Name: " + employeeList.get(i).getName();;
                str += "\n";
                str = str + "Surname: " + employeeList.get(i).getSurname();
                str += "\n";
                str += "Employee Id: " + employeeList.get(i).getId();
                str += "\n";
                str = str + "Branch Id: " + employeeList.get(i).getBranchId();
                str += "\n";
            str += "\n\n";
            }
        }
        else
            str = str + "There is no employee in the company\n";
        System.out.println(str);
    }
```

T1(N) = θ(n)(for loop) * O(n) (String is immutable so string will be copied to new string) = O(n^2)

T2(N) = θ(1)

T(N) = O(n^2) **-> Best case = T2(N) =** θ(1), **Worst case = T1(N) =** O(n^2)

seeProductList Method:



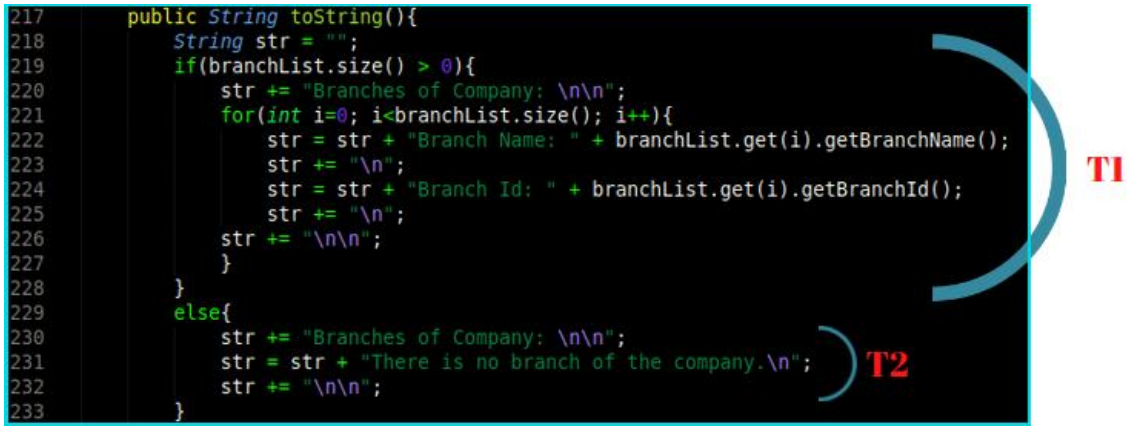
```
    public void seeProductList(){
        int count = 0;
        for(int i=0; i<furnitureList.size(); i++){
            for(int j=0; j<furnitureList.get(i).size(); j++){
                if(furnitureList.get(i).get(j).getNumberInStock() > 0){
                    System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                    System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                    System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                    System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                    System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                    System.out.println("\n");
                }
                else{
                    System.out.println("***SOLD OUT***");
                    System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                    System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                    System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                    System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                    System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                    System.out.println("\n");
                    count += 1;
                }
            }
        }
        if(count == furnitureList.getElCount())
            System.out.println("Sorry, all stock is empty now");
    }
```

T(n) = θ(n)(from outer for loop) * θ(n)(from inner for loop) *

(O(n)(from linked list get method) + θ(1) (from array list get method)) + θ(1)(last if condition) = O(n^3)

***Here it doesn't matter whether first if or else condition is valid because condition in first if condition is O(n), bodies of first if and else are O(n) and O(n) + O(n) = O(n)**

toString Method:



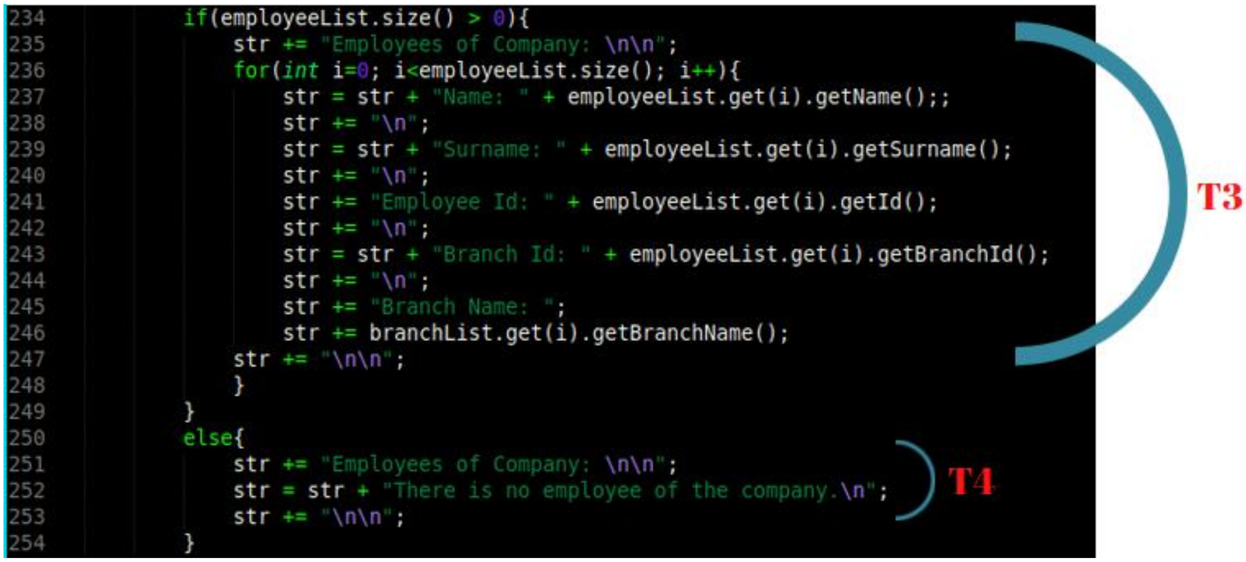
```
    public String toString(){
        String str = "";
        if(branchList.size() > 0){
            str += "Branches of Company: \n\n";
            for(int i=0; i<branchList.size(); i++){
                str = str + "Branch Name: " + branchList.get(i).getBranchName();
                str += "\n";
                str = str + "Branch Id: " + branchList.get(i).getBranchId();
                str += "\n";
            str += "\n\n";
            }
        }
        else{
            str += "Branches of Company: \n\n";
            str = str + "There is no branch of the company.\n";
            str += "\n\n";
        }
```

T1(N) = θ(n)(for loop) *

(O(n)(from linked list get method) + O(n) (String is immutable so string will be copied to new string))

= O(n^2)

T2(N) = O(n) (String is immutable so string will be copied to new string)

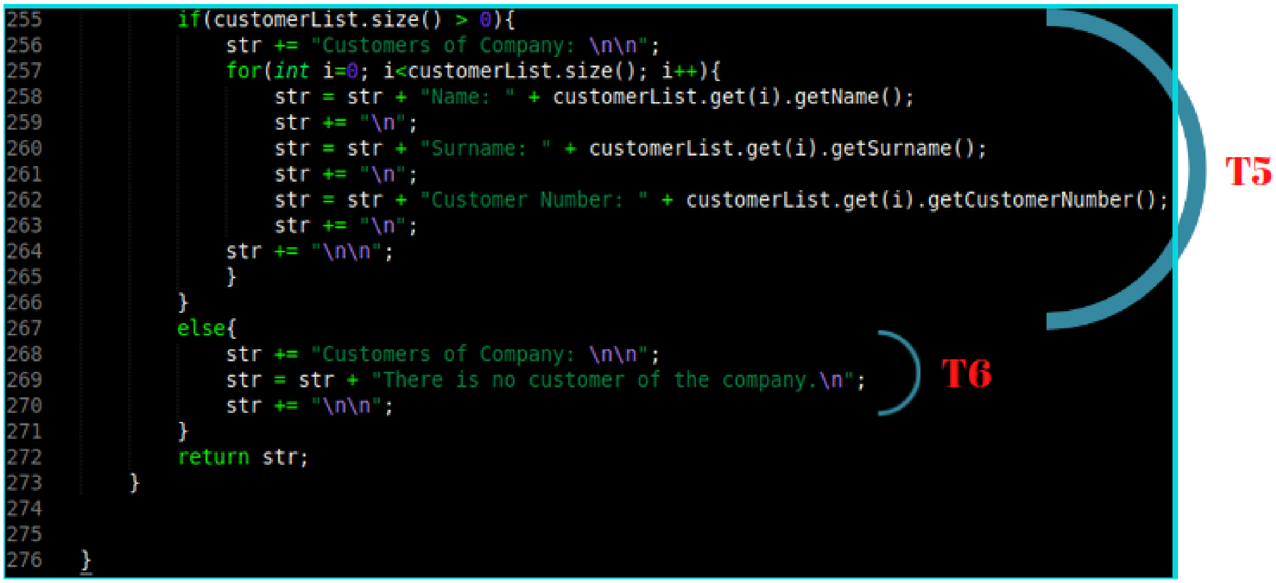


```
        if(employeeList.size() > 0){
            str += "Employees of Company: \n\n";
            for(int i=0; i<employeeList.size(); i++){
                str = str + "Name: " + employeeList.get(i).getName();;
                str += "\n";
                str = str + "Surname: " + employeeList.get(i).getSurname();
                str += "\n";
                str += "Employee Id: " + employeeList.get(i).getId();
                str += "\n";
                str = str + "Branch Id: " + employeeList.get(i).getBranchId();
                str += "\n";
                str += "Branch Name: ";
                str += branchList.get(i).getBranchName();
            str += "\n\n";
            }
        }
        else{
            str += "Employees of Company: \n\n";
            str = str + "There is no employee of the company.\n";
            str += "\n\n";
        }
```

T3(N) = θ(n)(for loop) *

(θ(1) (from array list get method) + O(n) (String is immutable so string will be copied to new string)) = O(n^2)

T4(N) = O(n) (String is immutable so string will be copied to new string)



```
            if(customerList.size() > 0){
                str += "Customers of Company: \n\n";
                for(int i=0; i<customerList.size(); i++){
                    str = str + "Name: " + customerList.get(i).getName();
                    str += "\n";
                    str = str + "Surname: " + customerList.get(i).getSurname();
                    str += "\n";
                    str = str + "Customer Number: " + customerList.get(i).getCustomerNumber();
                    str += "\n";
                str += "\n\n";
                }
            }
            else{
                str += "Customers of Company: \n\n";
                str = str + "There is no customer of the company.\n";
                str += "\n\n";
            }
            return str;
        }


    }
```

T5(N) = $\theta(n)$(for loop) *

($\theta(1)$ (from array list get method) + O(n) (String is immutable so string will be copied to new string)) = $O(n^2)$

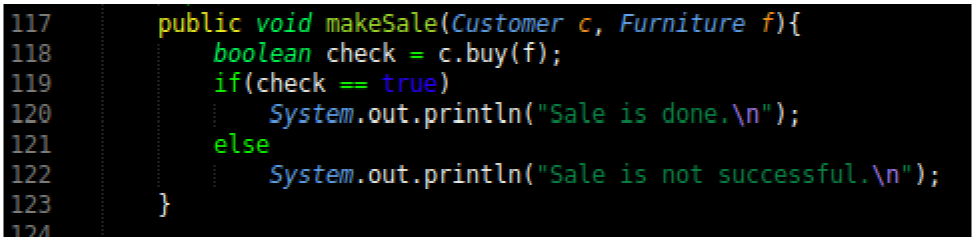
T6(N) = O(n) (String is immutable so string will be copied to new string)

T(N) = $O(n^2)$ -> Best case = T2(N) + T4(N) + T6(N) = O(n)

Worst case = T1(N) + T3(N) + T5(N) = $O(n^2)$

BRANCHEMPLOYEE CLASS

makeSale Method:

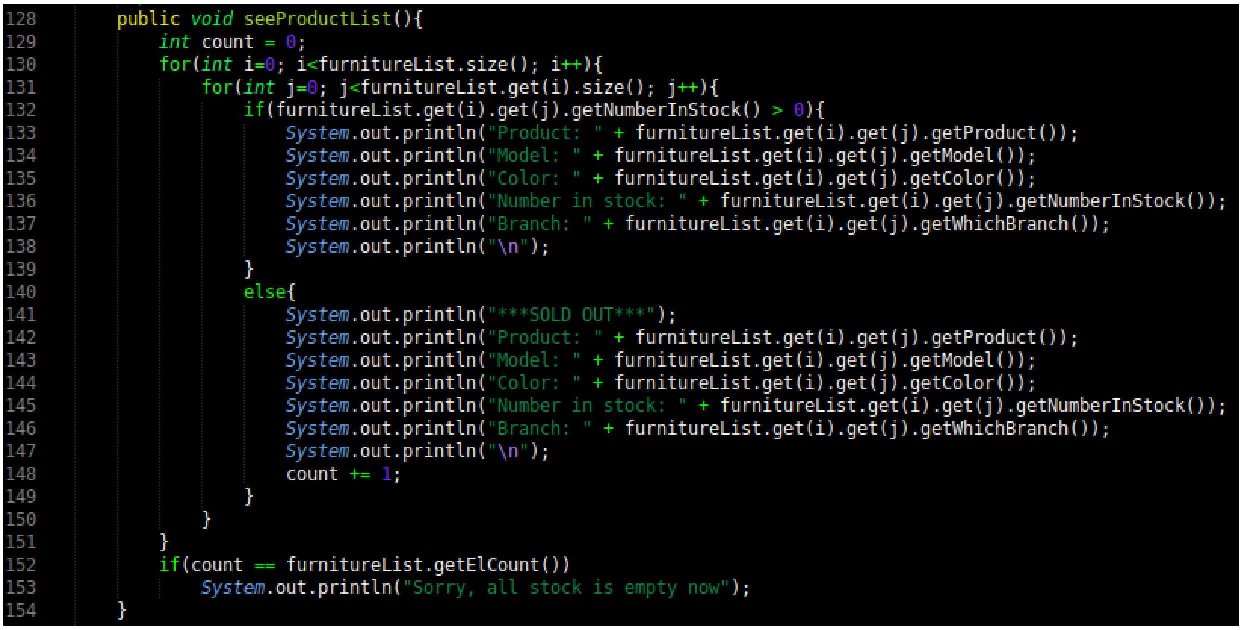


```java
    public void makeSale(Customer c, Furniture f){
        boolean check = c.buy(f);
        if(check == true)
            System.out.println("Sale is done.\n");
        else
            System.out.println("Sale is not successful.\n");
    }
```

T(N) = $O(n^3)$ -> Best case = buy method best case + constant time(if,else) = $\theta(1)$

Worst case = buy method worst case + constant time(if,else) = $O(n^3)$

seeProductList Method:



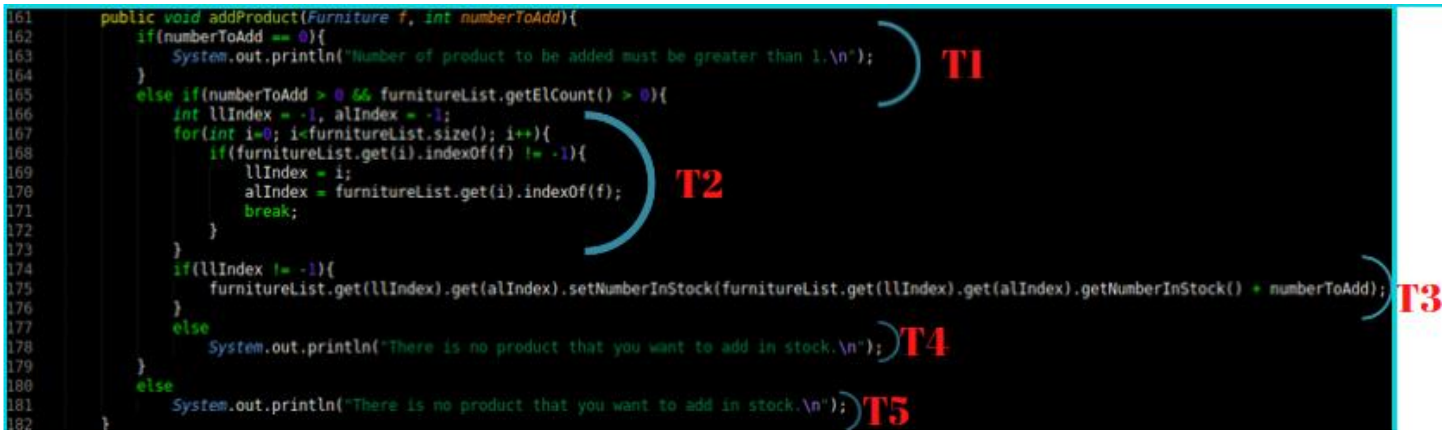
```java
    public void seeProductList(){
        int count = 0;
        for(int i=0; i<furnitureList.size(); i++){
            for(int j=0; j<furnitureList.get(i).size(); j++){
                if(furnitureList.get(i).get(j).getNumberInStock() > 0){
                    System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                    System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                    System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                    System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                    System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                    System.out.println("\n");
                }
                else{
                    System.out.println("***SOLD OUT***");
                    System.out.println("Product: " + furnitureList.get(i).get(j).getProduct());
                    System.out.println("Model: " + furnitureList.get(i).get(j).getModel());
                    System.out.println("Color: " + furnitureList.get(i).get(j).getColor());
                    System.out.println("Number in stock: " + furnitureList.get(i).get(j).getNumberInStock());
                    System.out.println("Branch: " + furnitureList.get(i).get(j).getWhichBranch());
                    System.out.println("\n");
                    count += 1;
                }
            }
        }
        if(count == furnitureList.getElCount())
            System.out.println("Sorry, all stock is empty now");
    }
```

T(n) = $\theta(n)$(from outer for loop) * $\theta(n)$(from inner for loop) *

($O(n)$(from linked list get method) + $\theta(1)$ (from array list get method)) + $\theta(1)$(last if condition) = $O(n^3)$

***Here it doesn't matter whether first if or else condition is valid because condition in first if condition is O(n), bodies of first if and else are O(n) and O(n) + O(n) = O(n)**

addProduct Method:



```java
public void addProduct(Furniture f, int numberToAdd){
    if(numberToAdd == 0){
        System.out.println("Number of product to be added must be greater than 1.\n");
    }
    else if(numberToAdd > 0 && furnitureList.getElCount() > 0){
        int llIndex = -1, alIndex = -1;
        for(int i=0; i<furnitureList.size(); i++){
            if(furnitureList.get(i).indexOf(f) != -1){
                llIndex = i;
                alIndex = furnitureList.get(i).indexOf(f);
                break;
            }
        }
        if(llIndex != -1){
            furnitureList.get(llIndex).get(alIndex).setNumberInStock(furnitureList.get(llIndex).get(alIndex).getNumberInStock() + numberToAdd);
        }
        else
            System.out.println("There is no product that you want to add in stock.\n");
    }
    else
        System.out.println("There is no product that you want to add in stock.\n");
}
```

T1(N) = $\theta(1)$

T2(N) = O(n) -> Best case = $\theta(1)$(for loop) *

($O(n)$ (from linked list get method) + $O(n)$ (from array list indexOf method)) = $O(n)$

Worst case = $\theta(n)$(for loop) *

($O(n)$ (from linked list get method) + $O(n)$ (from array list indexOf method)) = $O(n^2)$

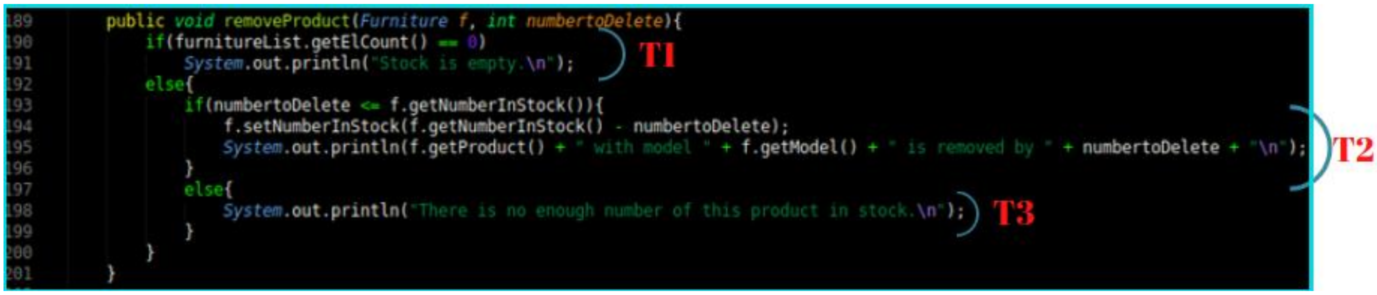
T3(N) = $O(n)$ (from linked list get method) + $\theta(1)$ (from array list get method) = $O(n)$

T4(N) = $\theta(1)$

T5(N) = $\theta(1)$

T(N) = $O(n^2)$ **-> Best case = T1(N) or T5(N) =** $\theta(1)$, **Worst case = T2(N)(worst case) + T3(N) =** $O(n^2)$

removeProduct Method



```
        public void removeProduct(Furniture f, int numbertoDelete){
            if(furnitureList.getElCount() == 0)
                System.out.println("Stock is empty.\n");
            else{
                if(numbertoDelete <= f.getNumberInStock()){
                    f.setNumberInStock(f.getNumberInStock() - numbertoDelete);
                    System.out.println(f.getProduct() + " with model " + f.getModel() + " is removed by " + numbertoDelete + "\n");
                }
                else{
                    System.out.println("There is no enough number of this product in stock.\n");
                }
            }
        }
```

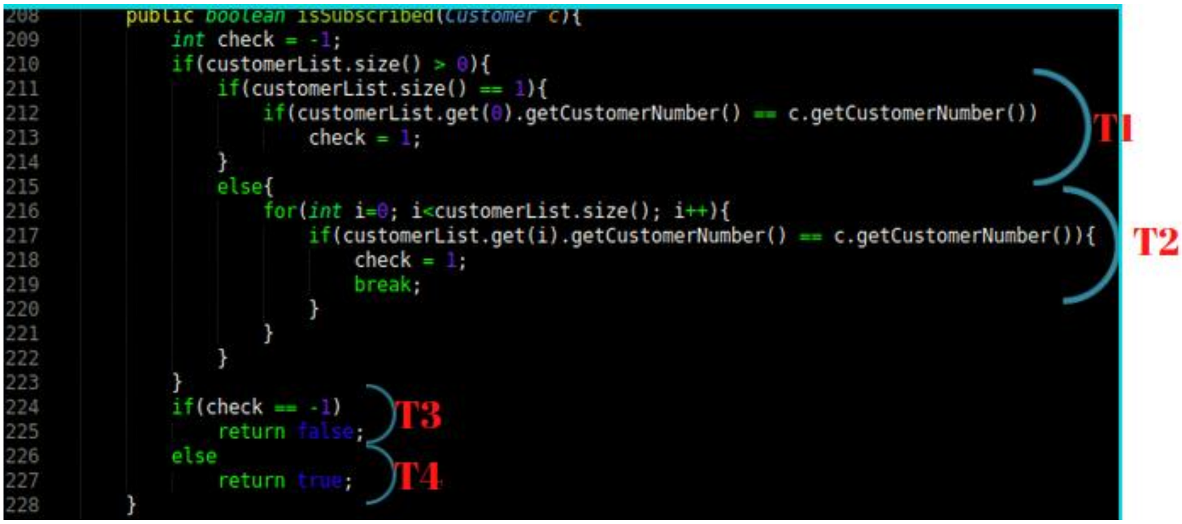
T1(N) = θ(1)

T2(N) = θ(1)(getter,setter)

T3(N) = θ(1)

T(N) = θ(1)

isSubscribed Method



```
        public boolean isSubscribed(Customer c){
            int check = -1;
            if(customerList.size() > 0){
                if(customerList.size() == 1){
                    if(customerList.get(0).getCustomerNumber() == c.getCustomerNumber())
                        check = 1;
                }
                else{
                    for(int i=0; i<customerList.size(); i++){
                        if(customerList.get(i).getCustomerNumber() == c.getCustomerNumber()){
                            check = 1;
                            break;
                        }
                    }
                }
            }
            if(check == -1)
                return false;
            else
                return true;
        }
```

T1(N) = θ(1)(arraylist get method) + other constant time operations = θ(1)

T2(N) = θ**(n) -> Best case =** θ(1)(for loop) * (θ(1)(arraylist get method) + other constant time operations)) = θ(1)

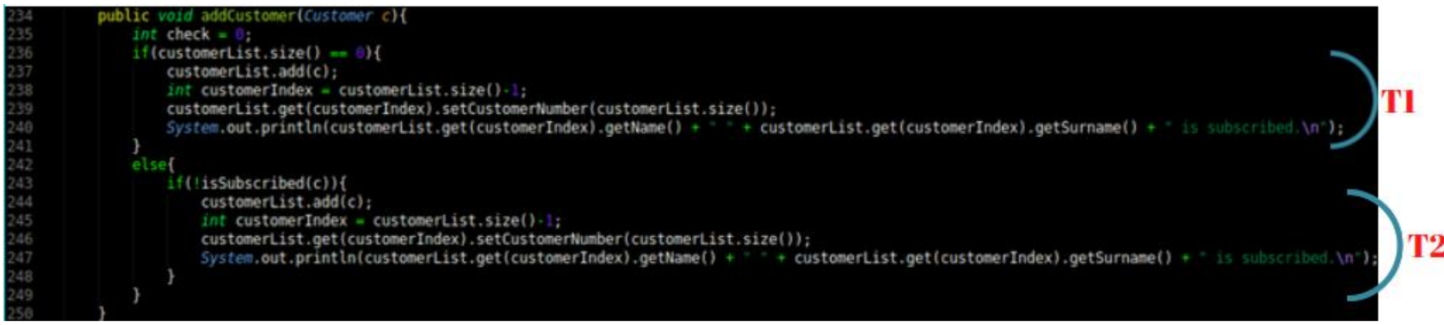
Worst case = θ(n)(for loop) * (θ(1)(arraylist get method) + other constant time operations)) = θ(n)

T3(N) = T4(N) = θ(1)

T(N) = θ **(n) -> Best case = T1(N) + T2(N)(best case) + T3(N) or T4(N) =** θ **(1),**

Worst case = T1(N) + T2(N)(worst case) + T3(N) or T4(N) = θ (n)

addCustomer Method



```
    public void addCustomer(Customer c){
        int check = 0;
        if(customerList.size() == 0){
            customerList.add(c);
            int customerIndex = customerList.size()-1;
            customerList.get(customerIndex).setCustomerNumber(customerList.size());
            System.out.println(customerList.get(customerIndex).getName() + " " + customerList.get(customerIndex).getSurname() + " is subscribed.\n");
        }
        else{
            if(!isSubscribed(c)){
                customerList.add(c);
                int customerIndex = customerList.size()-1;
                customerList.get(customerIndex).setCustomerNumber(customerList.size());
                System.out.println(customerList.get(customerIndex).getName() + " " + customerList.get(customerIndex).getSurname() + " is subscribed.\n");
            }
        }
    }
```

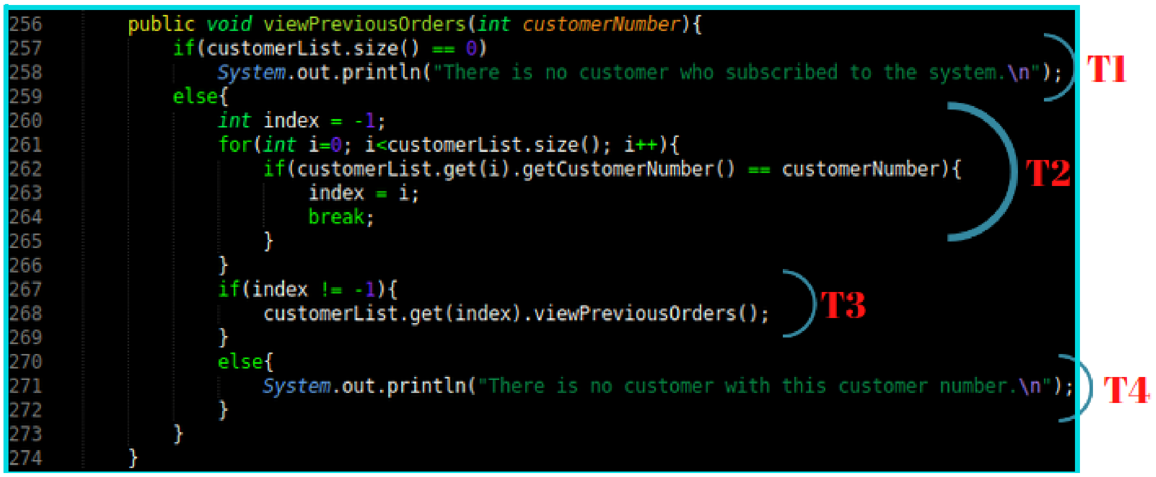
T1(N) = θ(n) because all operations in if are θ(n)

T2(N) = θ(n) -> Best case isSubscribed method best case + constant time operations = θ(1),

Worst case = Worst case isSubscribed method best case + constant time operations = θ(n)

T(N) = θ(n) -> Best case = T1(N) = θ(n), Worst case = T2(N) = θ(n)

viewPreviousOrders Method:



```
public void viewPreviousOrders(int customerNumber){
    if(customerList.size() == 0)
        System.out.println("There is no customer who subscribed to the system.\n");
    else{
        int index = -1;
        for(int i=0; i<customerList.size(); i++){
            if(customerList.get(i).getCustomerNumber() == customerNumber){
                index = i;
                break;
            }
        }
        if(index != -1){
            customerList.get(index).viewPreviousOrders();
        }
        else{
            System.out.println("There is no customer with this customer number.\n");
        }
    }
}
```

T1(N) = θ(1)

T2(N) = θ(n) -> Best case = θ(1) when customer is found at index 0,

Worst case = θ(n)

T3(N) = O(n^3) -> Best case = θ(1) which is best case of Customer Class's viewPreviousOrders method

Worst case = O(n^3) which is worst case of Customer Class's viewPreviousOrders method

T4(N) = θ(1)

T(N) = O(n^3) -> Best case = T1(N) = θ(1)

Worst case = T2(N) + T3(N) = O(n^3)