GIT Department of Computer Engineering CSE 222/505 - Spring 2021 Homework3 # Report

Mustafa Gurler

171044034

System Requirements:

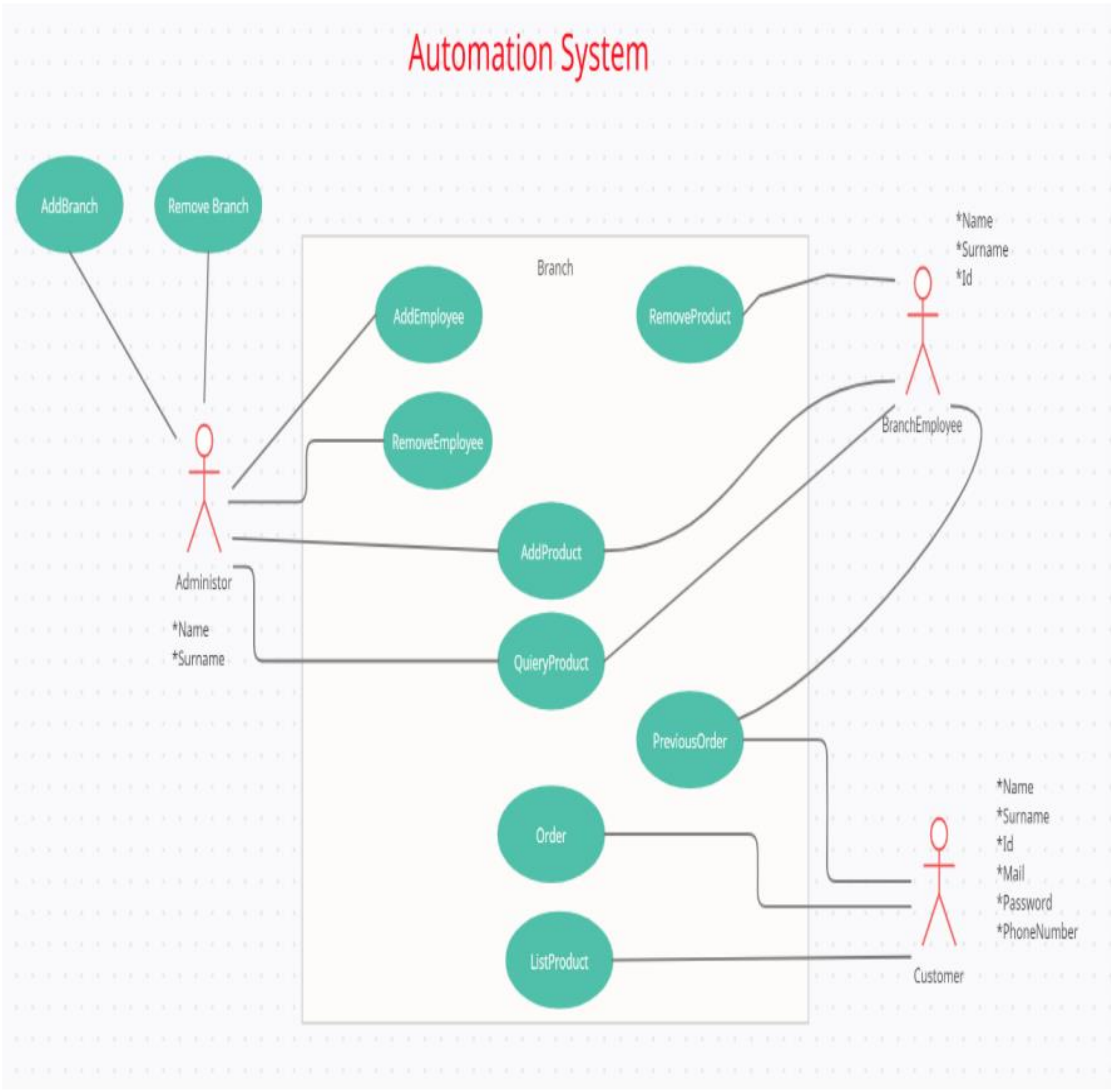
It has been needed to create an automation class for users who finds to add products to branch not easily. I added some features for system. Product class has been used with a structure which is hybrid list.

I designed an interface for automation system for Company which makes sales different model and color of products. Humans in the automation system are administrator, branch employees and customers. These administrators can add and remove branches, employees. Also add new product and inquire product in branches. Branches have their own products. First order of customer has to be a subscription. So employee has to make a new subscription. Also customer can purchase new products. It removes the same product from branches. Customers and employees can have a access feature to old customer orders. Firstly, there is a need for abstract class for the users which is Human. This abstract class should include getter and setter for the user’s name and surname. The class named Company has fields of customers, branches and administrators. These branches have their own products. Admin object has access to manage these branches and employees. So Administrator class has been added with fields. Admin has to add at least one branch and employee to run the other objects like customer and branch employee. Secondly, there is a need for interface for products which is OfficeProduct. This interface has all the data for products. Clients can add more model or color to his/her company.

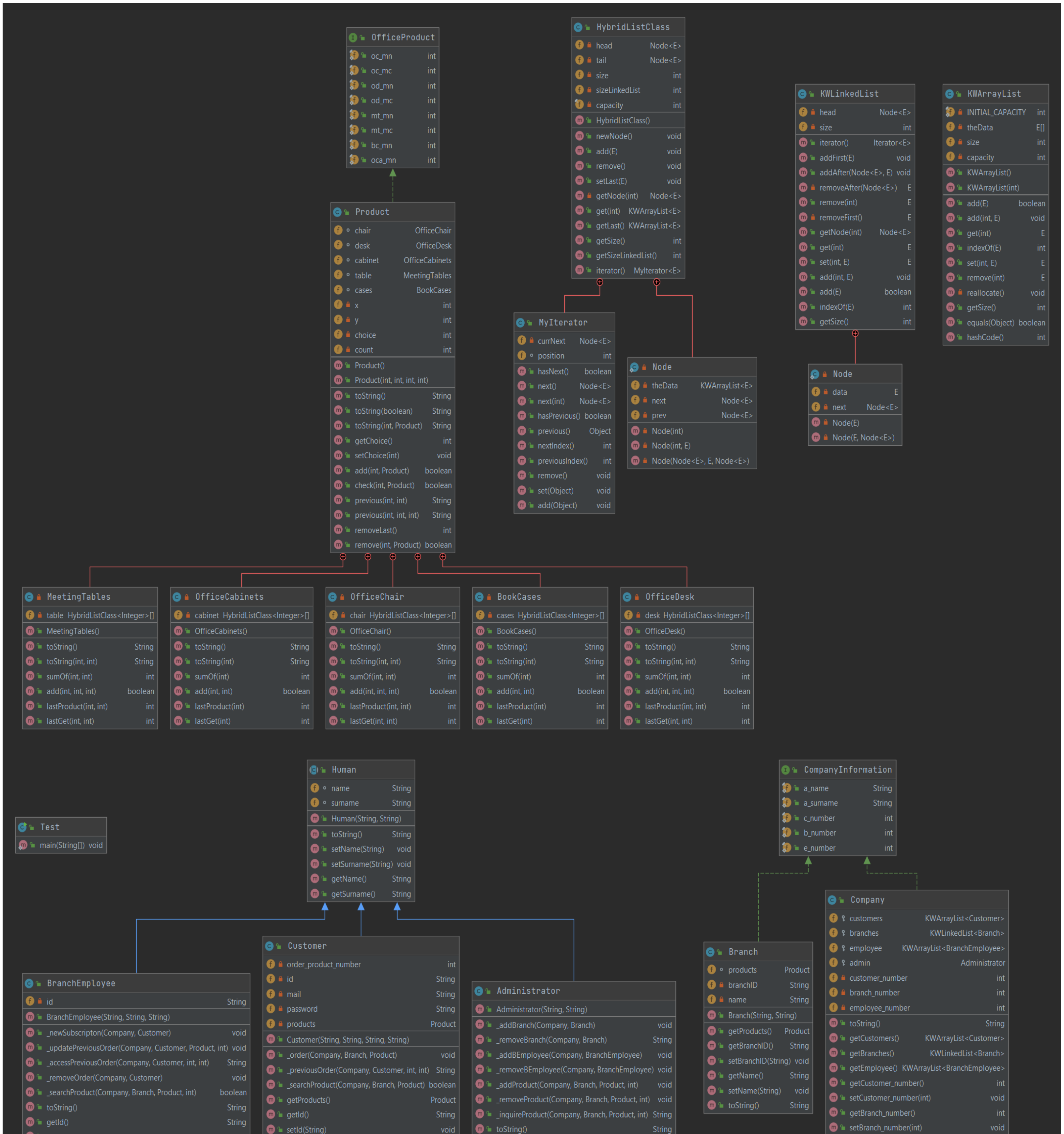
Product class has own 5 class which they are office chair , office desk, meeting tables, office cabinets and book cases. All the product models and colors are in hybrid class which can be add and remove product easily. Hybrid class has a linked list future and holds array list inside of the node.

```
Picked up _JAVA_OPTIONS: -
Dawt.useSystemAAFontSettings=on –

Dswing.aatext=true openjdk version "11.0.11-ea" 2021-04-20
OpenJDK Runtime Environment (build 11.0.11-ea+4-post-
Debian-1) OpenJDK 64-Bit Server VM (build 11.0.11-ea+4-
post-Debian-1, mixed mode, sharing)
```

<u>Use Case Diagram:</u>



UML Diagram:


OfficeProduct
oc_mn int
oc_mc int
od_mn int
od_mc int
mt_mn int
mt_mc int
bc_mn int
oca_mn int
HybridListClass
head Node<E>
tail Node<E>
size int
sizeLinkedList int
capacity int
HybridListClass()
newNode() void
add(E) void
remove() void
setLast(E) void
getNode(int) Node<E>
get(int) KWArrayList<E>
getLast() KWArrayList<E>
getSize() int
getSizeLinkedList() int
iterator() MyIterator<E>
KWLinkedList
head Node<E>
size int
iterator() Iterator<E>
addFirst(E) void
addAfter(Node<E>, E) void
removeAfter(Node<E>) E
remove(int) E
removeFirst() E
getNode(int) Node<E>
get(int) E
set(int, E) E
add(int, E) void
add(E) boolean
indexOf(E) int
getSize() int
KWArrayList
INITIAL_CAPACITY int
theData E[]
size int
capacity int
KWArrayList()
KWArrayList(int)
add(E) boolean
add(int, E) void
get(int) E
indexOf(E) int
set(int, E) E
remove(int) E
reallocate() void
getSize() int
equals(Object) boolean
hashCode() int
Product
chair OfficeChair
desk OfficeDesk
cabinet OfficeCabinets
table MeetingTables
cases BookCases
x int
y int
choice int
count int
Product()
Product(int, int, int, int)
toString() String
toString(boolean) String
toString(int, Product) String
getChoice() int
setChoice(int) void
add(int, Product) boolean
check(int, Product) boolean
previous(int, int) String
previous(int, int, int) String
removeLast() int
remove(int, Product) boolean
MyIterator
currNext Node<E>
position int
hasNext() boolean
next() Node<E>
next(int) Node<E>
hasPrevious() boolean
previous() Object
nextIndex() int
previousIndex() int
remove() void
set(Object) void
add(Object) void
Node
theData KWArrayList<E>
next Node<E>
prev Node<E>
Node(int)
Node(int, E)
Node(Node<E>, E, Node<E>)
Node
data E
next Node<E>
Node(E)
Node(E, Node<E>)
MeetingTables
table HybridListClass<Integer>[]
MeetingTables()
toString() String
toString(int, int) String
sumOf(int, int) int
add(int, int, int) boolean
lastProduct(int, int) int
lastGet(int, int) int
OfficeCabinets
cabinet HybridListClass<Integer>[]
OfficeCabinets()
toString() String
toString(int) String
sumOf(int) int
add(int, int) boolean
lastProduct(int) int
lastGet(int) int
OfficeChair
chair HybridListClass<Integer>[]
OfficeChair()
toString() String
toString(int, int) String
sumOf(int, int) int
add(int, int, int) boolean
lastProduct(int, int) int
lastGet(int, int) int
BookCases
cases HybridListClass<Integer>[]
BookCases()
toString() String
toString(int) String
sumOf(int) int
add(int, int) boolean
lastProduct(int) int
lastGet(int) int
OfficeDesk
desk HybridListClass<Integer>[]
OfficeDesk()
toString() String
toString(int, int) String
sumOf(int, int) int
add(int, int, int) boolean
lastProduct(int, int) int
lastGet(int, int) int
Test
main(String[]) void
Human
name String
surname String
Human(String, String)
toString() String
setName(String) void
setSurname(String) void
getName() String
getSurname() String
CompanyInformation
a_name String
a_surname String
c_number int
b_number int
e_number int
BranchEmployee
id String
BranchEmployee(String, String, String)
_newSubscripton(Company, Customer) void
_updatePreviousOrder(Company, Customer, Product, int) void
_accessPreviousOrder(Company, Customer, int, int) String
_removeOrder(Company, Customer) void
_searchProduct(Company, Branch, Product, int) boolean
toString() String
getId() String
Customer
order_product_number int
id String
mail String
password String
products Product
Customer(String, String, String, String)
_order(Company, Branch, Product) void
_previousOrder(Company, Customer, int, int) String
_searchProduct(Company, Branch, Product) boolean
getProducts() Product
getId() String
setId(String) void
Administrator
Administrator(String, String)
_addBranch(Company, Branch) void
_removeBranch(Company, Branch) String
_addBEmployee(Company, BranchEmployee) void
_removeBEmployee(Company, BranchEmployee) void
_addProduct(Company, Branch, Product, int) void
_removeProduct(Company, Branch, Product, int) void
_inquireProduct(Company, Branch, Product, int) String
toString() String
Branch
products Product
branchID String
name String
Branch(String, String)
getProducts() Product
getBranchID() String
setBranchID(String) void
getName() String
setName(String) void
toString() String
Company
customers KWArrayList<Customer>
branches KWLinkedList<Branch>
employee KWArrayList<BranchEmployee>
admin Administrator
customer_number int
branch_number int
employee_number int
toString() String
getCustomers() KWArrayList<Customer>
getBranches() KWLinkedList<Branch>
getEmployee() KWArrayList<BranchEmployee>
getCustomer_number() int
setCustomer_number(int) void
getBranch_number() int
setBranch_number(int) void

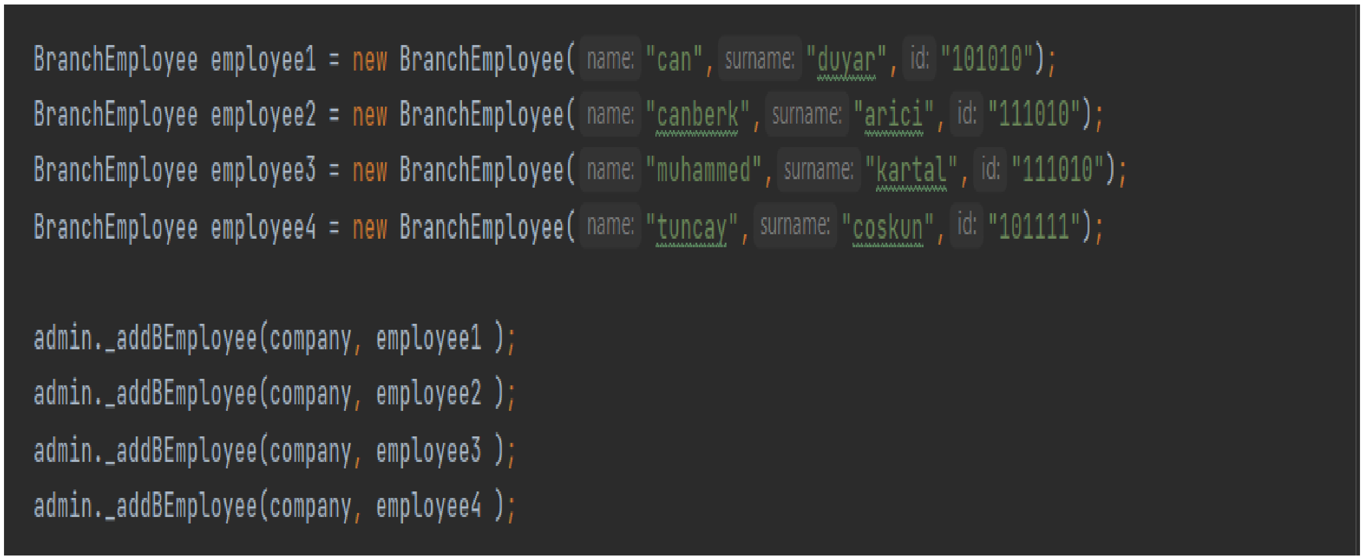
***5 branch has been added to company object**

4.Problem Solution Approach:

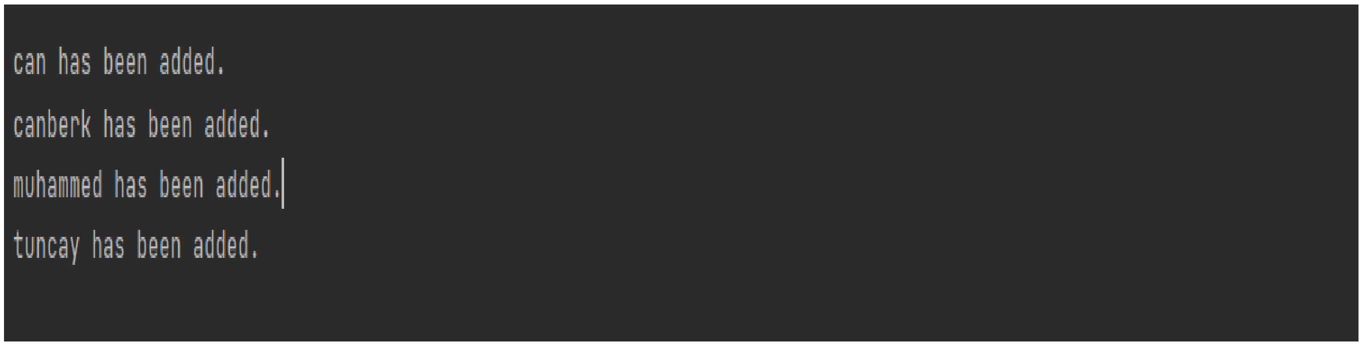
I created an interface for my client, so he/she can add some new models or colors, Also he/she can change his/her name whenever he wants, I simply create two interface, first one is about company information like admin name or customer number, second one creates an office products interface ,so it can be improved by client. I also create a abstract Human class which have getter and setter for my users. I created an company class which I can do my all customers, branches ,admin and employees. My methods are also available in another classes. In my main class I created an admin object and we have to make a branch and customer object. So we can continue with other menus level. Admin level we have six methods which we can add and remove branches and employees. Also add an new product or inquire product. I added to requested methods in customer level and branch employees level. My methods generally do their tasks with product class. Every branches and customers have their own product. I am adding every product to selected branches and customers. I tried to chose an employee during the employees order method. Product class has its own 5 classes. They are office chair, office desk, office cabinet, meeting tables, book cases. I hold them as a hybrid list in the product class. Also Product class has own methods which are previous added products, add product, remove product, check product if there are enough product.

Running and Test Cases:

Add Branch Employee



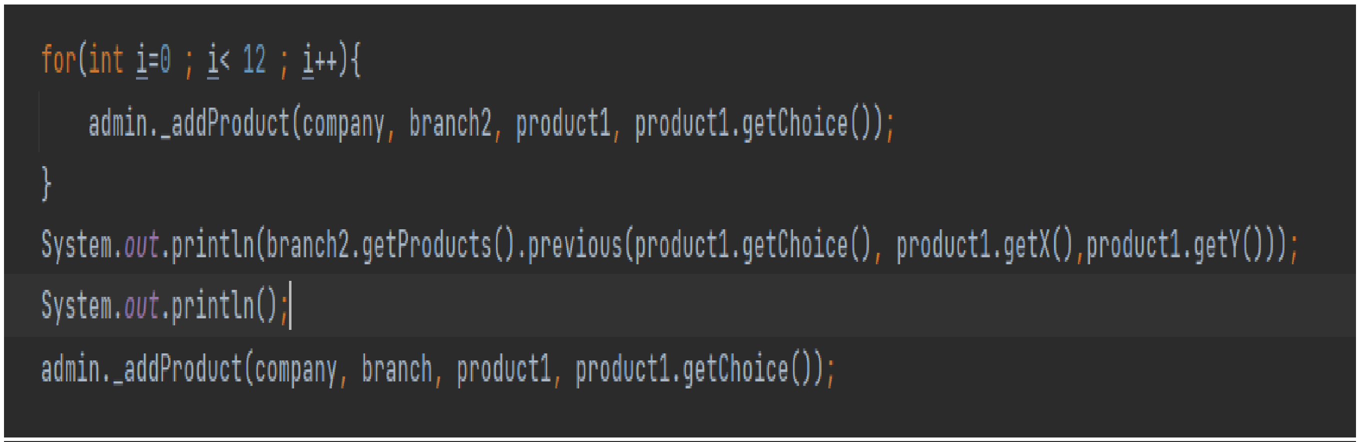
```
BranchEmployee employee1 = new BranchEmployee( name: "can", surname: "duyar", id: "101010");
BranchEmployee employee2 = new BranchEmployee( name: "canberk", surname: "arici", id: "111010");
BranchEmployee employee3 = new BranchEmployee( name: "muhammed", surname: "kartal", id: "111010");
BranchEmployee employee4 = new BranchEmployee( name: "tuncay", surname: "coskun", id: "101111");

admin._addBEmployee(company, employee1 );
admin._addBEmployee(company, employee2 );
admin._addBEmployee(company, employee3 );
admin._addBEmployee(company, employee4 );
```



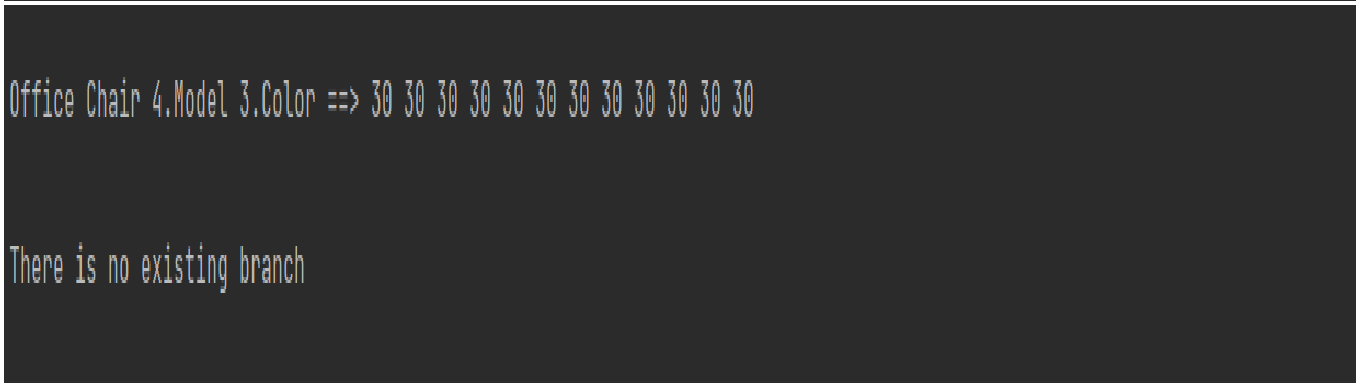
```
can has been added.
canberk has been added.
muhammed has been added.
tuncay has been added.
```

*4 different person has been added to company object

Add Product



```
for(int i=0 ; i< 12 ; i++){
    admin._addProduct(company, branch2, product1, product1.getChoice());
}
System.out.println(branch2.getProducts().previous(product1.getChoice(), product1.getX(),product1.getY()));
System.out.println();
admin._addProduct(company, branch, product1, product1.getChoice());
```

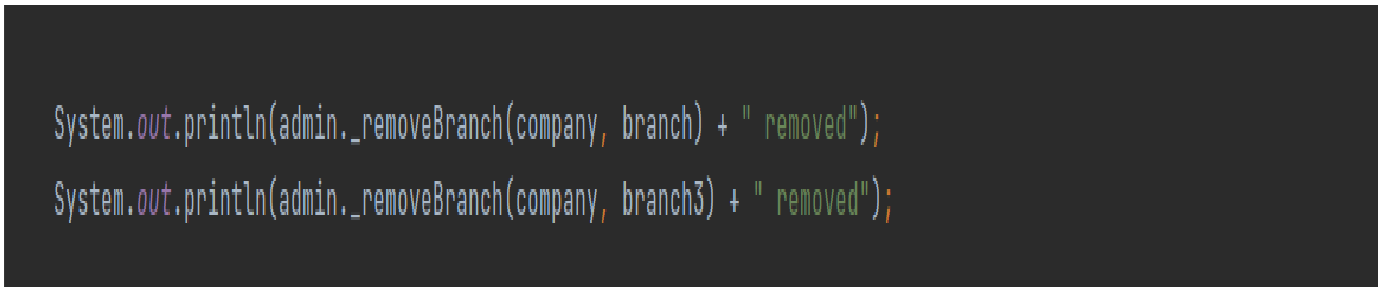


```
Office Chair 4.Model 3.Color ==> 30 30 30 30 30 30 30 30 30 30 30 30


There is no existing branch
```

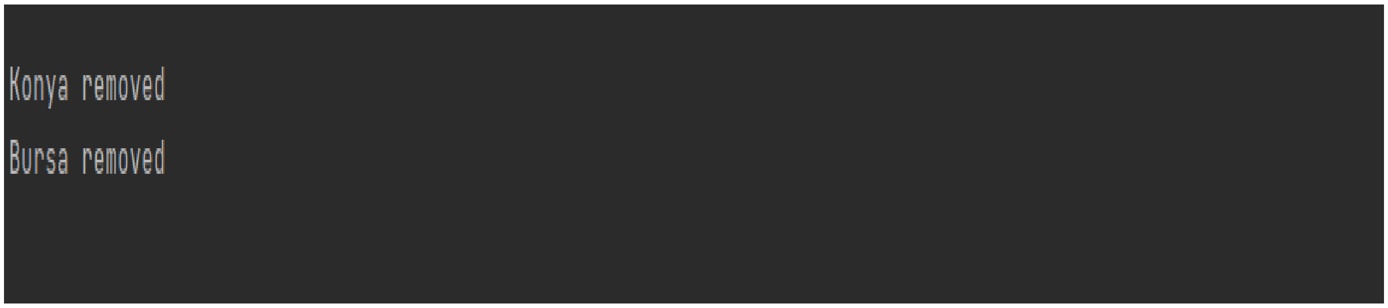
***12 times 30 chair product added to branch 2. Shows to previous order of the branch.**

*** Branch 1 is not exist in the Company object.**

Remove Branch



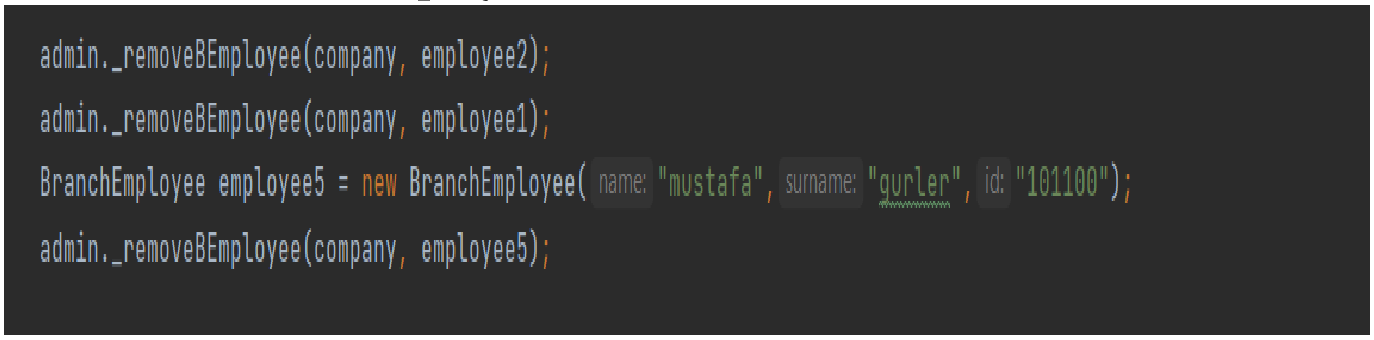
```
System.out.println(admin._removeBranch(company, branch) + " removed");
System.out.println(admin._removeBranch(company, branch3) + " removed");
```



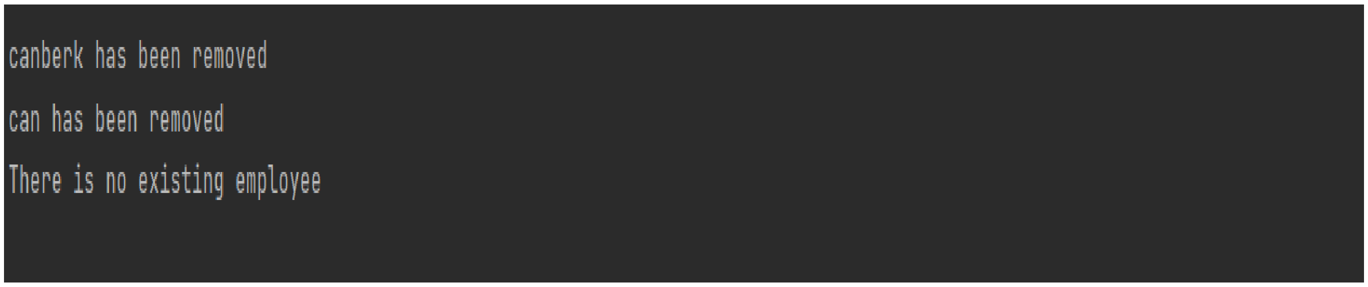
```
Konya removed
Bursa removed
```

*Branch and branch 3 has been removed from company object

Remove Branch Employee



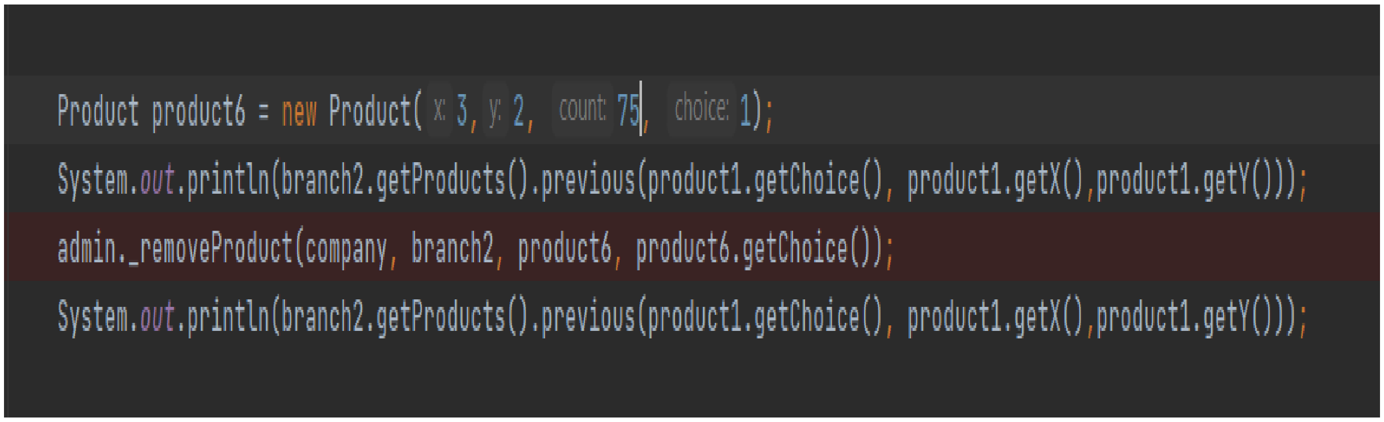
```
admin._removeBEmployee(company, employee2);
admin._removeBEmployee(company, employee1);
BranchEmployee employee5 = new BranchEmployee( name: "mustafa", surname: "gurler", id: "101100");
admin._removeBEmployee(company, employee5);
```



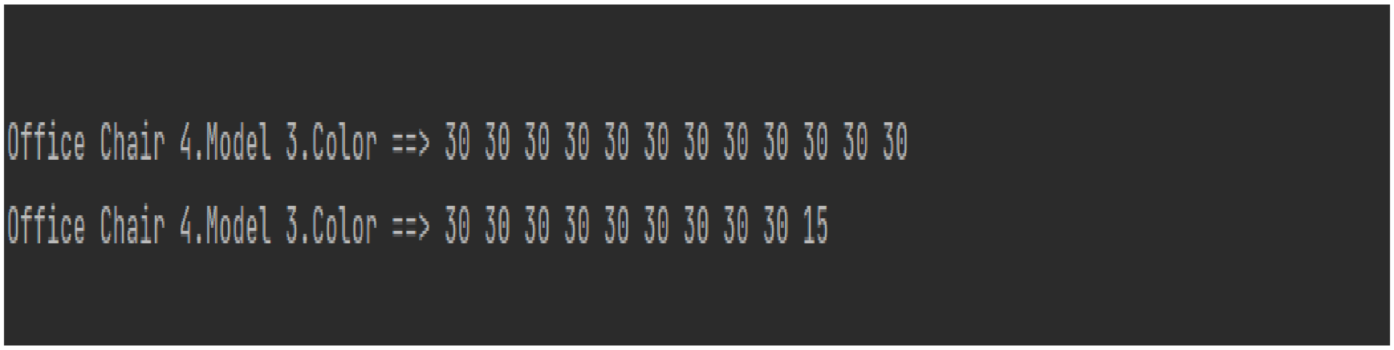
```
canberk has been removed
can has been removed
There is no existing employee
```

***Can and Canberk has been removed from company, but There is no existing Mustafa employee in the list.**

Remove Product



```
Product product6 = new Product( x: 3, y: 2,  count: 75,  choice: 1);
System.out.println(branch2.getProducts().previous(product1.getChoice(), product1.getX(),product1.getY()));
admin._removeProduct(company, branch2, product6, product6.getChoice());
System.out.println(branch2.getProducts().previous(product1.getChoice(), product1.getX(),product1.getY()));
```

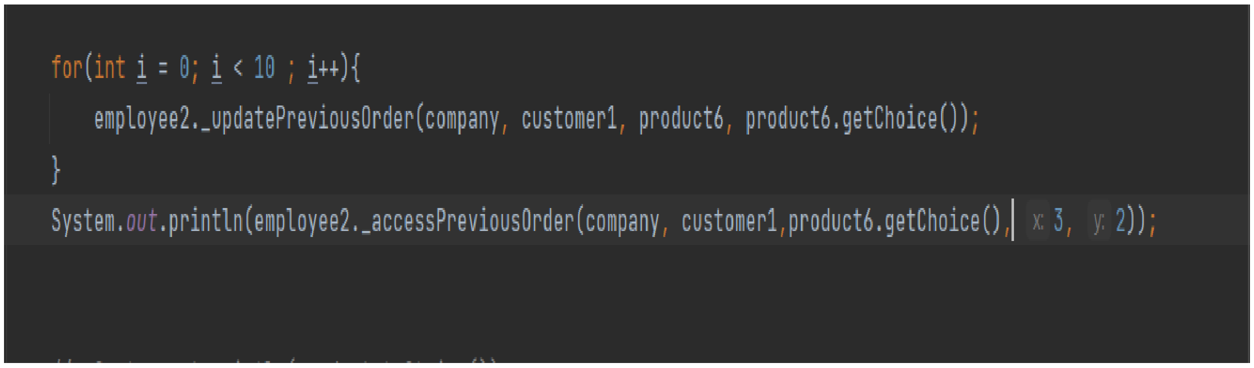


```
Office Chair 4.Model 3.Color ==> 30 30 30 30 30 30 30 30 30 30 30 30
Office Chair 4.Model 3.Color ==> 30 30 30 30 30 30 30 30 30 15
```

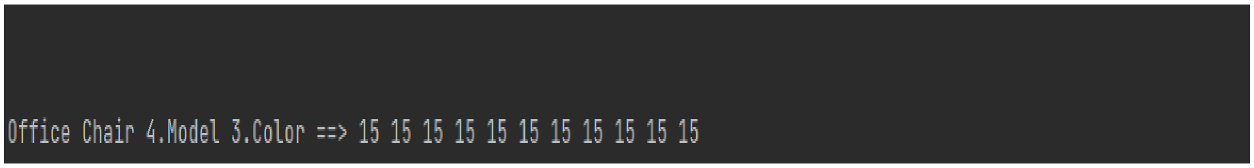
***12 times 30 4.model 3.color chair has been added to branch2. And removed 75 chair.**

Branch Employee Class:

Access Previous Order Class



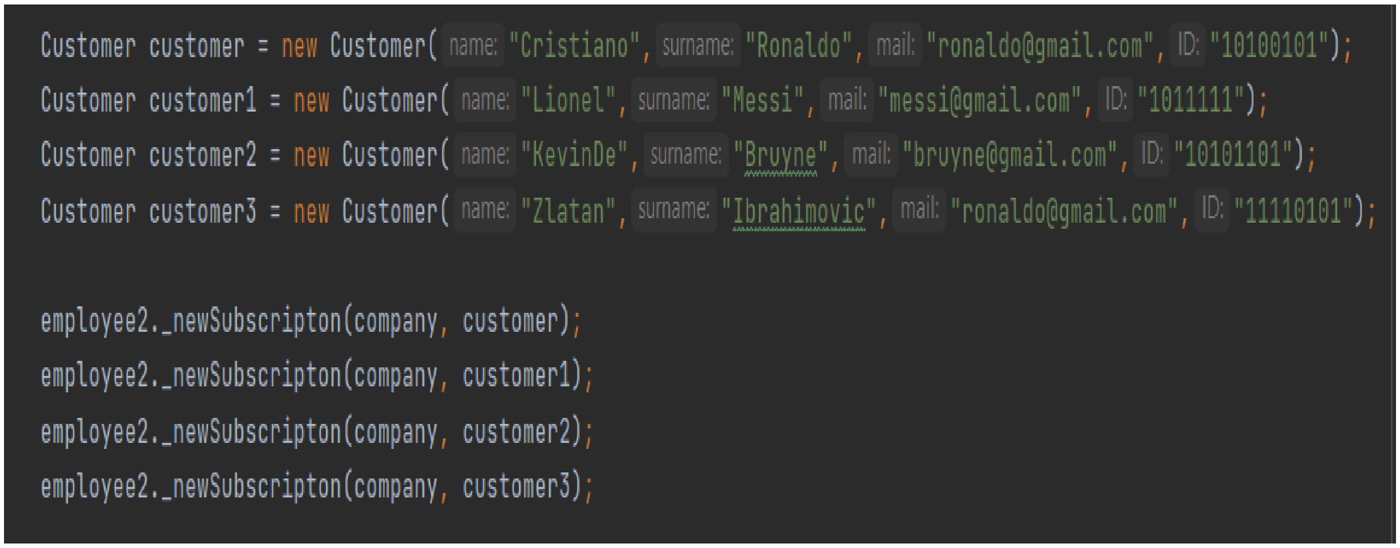
```
for(int i = 0; i < 10 ; i++){
    employee2._updatePreviousOrder(company, customer1, product6, product6.getChoice());
}
System.out.println(employee2._accessPreviousOrder(company, customer1,product6.getChoice(), x: 3, y: 2));
```



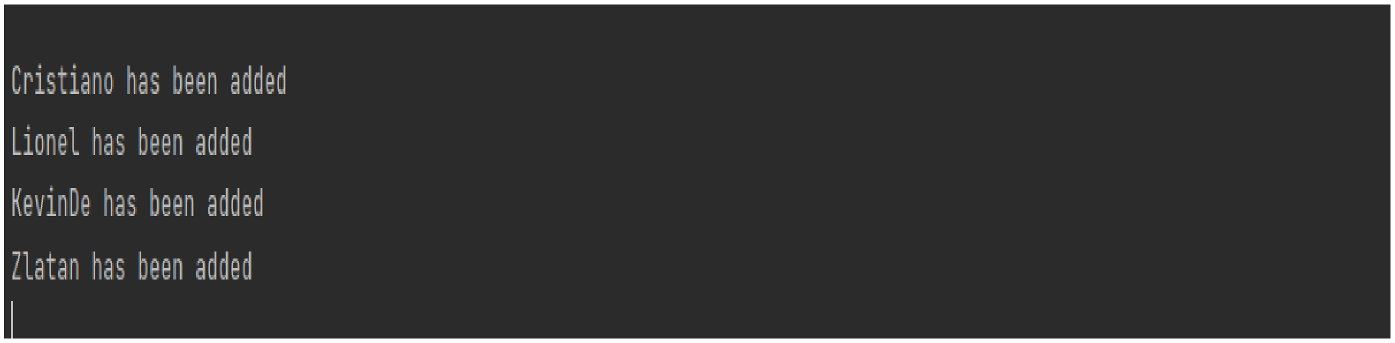
```
Office Chair 4.Model 3.Color ==> 15 15 15 15 15 15 15 15 15 15 15
```

***10 times 4.model 3.color Chair has been added and access to previous order.**

New Subscription



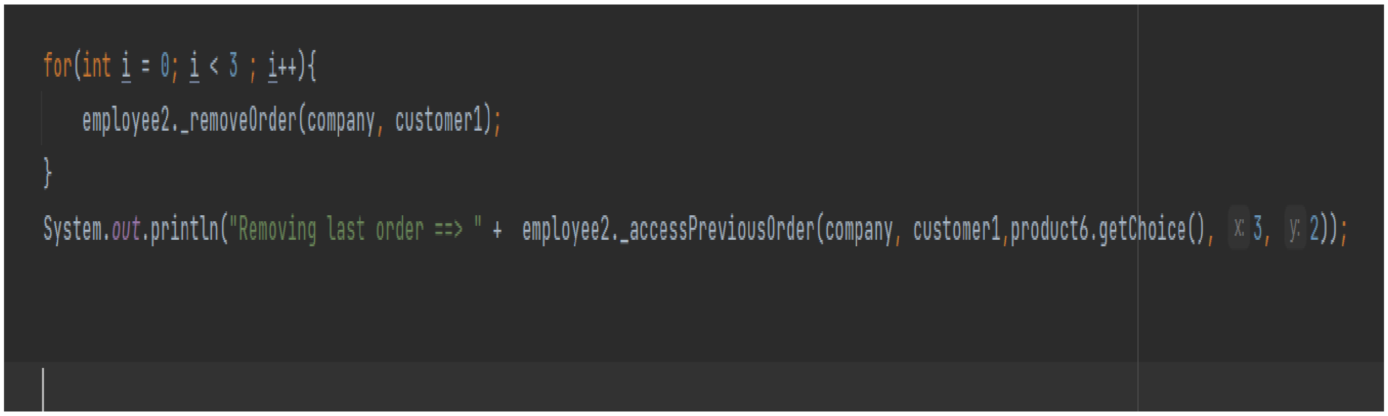
```
Customer customer = new Customer( name: "Cristiano", surname: "Ronaldo", mail: "ronaldo@gmail.com", ID: "10100101");
Customer customer1 = new Customer( name: "Lionel", surname: "Messi", mail: "messi@gmail.com", ID: "1011111");
Customer customer2 = new Customer( name: "KevinDe", surname: "Bruyne", mail: "bruyne@gmail.com", ID: "10101101");
Customer customer3 = new Customer( name: "Zlatan", surname: "Ibrahimovic", mail: "ronaldo@gmail.com", ID: "11110101");

employee2._newSubscripton(company, customer);
employee2._newSubscripton(company, customer1);
employee2._newSubscripton(company, customer2);
employee2._newSubscripton(company, customer3);
```



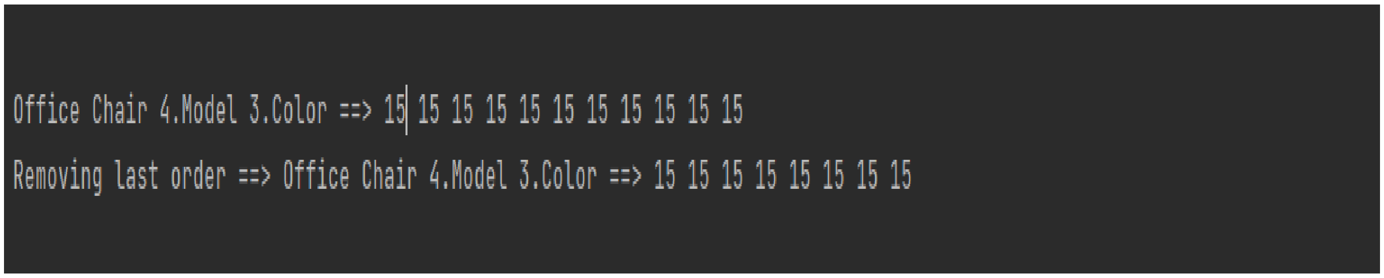
```
Cristiano has been added
Lionel has been added
KevinDe has been added
Zlatan has been added
```

***4 different customer has been added to company.**

Removing Last Order



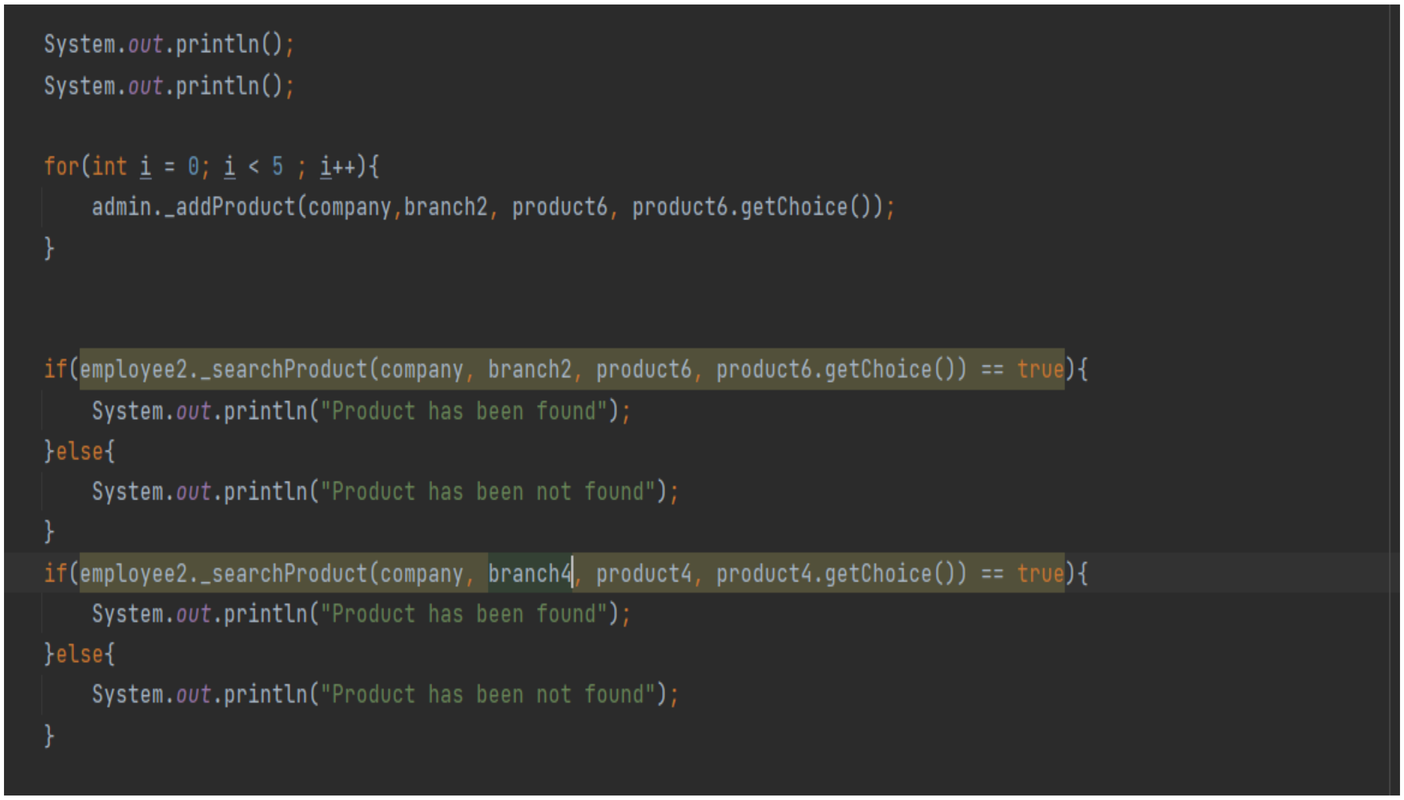
```
for(int i = 0; i < 3 ; i++){
    employee2._removeOrder(company, customer1);
}
System.out.println("Removing last order ==> " +  employee2._accessPreviousOrder(company, customer1,product6.getChoice(), x: 3, y: 2));
```



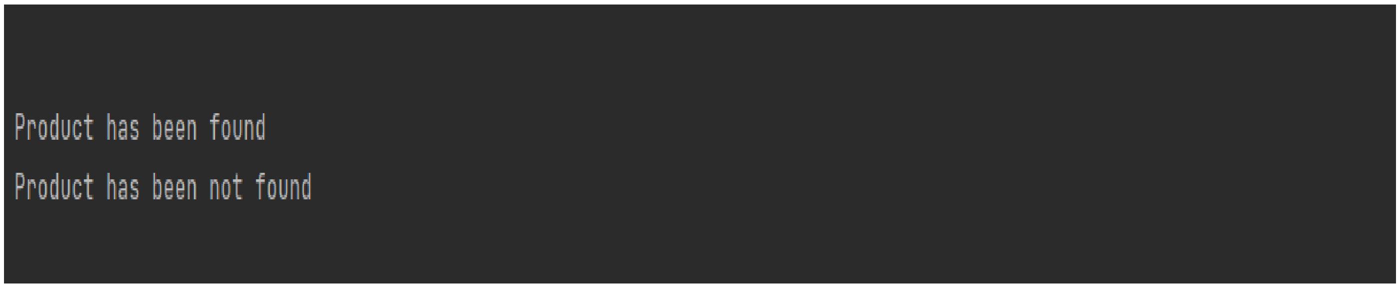
```
Office Chair 4.Model 3.Color ==> 15 15 15 15 15 15 15 15 15 15 15
Removing last order ==> Office Chair 4.Model 3.Color ==> 15 15 15 15 15 15 15 15
```

***11 chair has been added to customer 1 and 3 of them have been removed.**

Search Product



```java
System.out.println();
System.out.println();

for(int i = 0; i < 5 ; i++){
    admin._addProduct(company,branch2, product6, product6.getChoice());
}


if(employee2._searchProduct(company, branch2, product6, product6.getChoice()) == true){
    System.out.println("Product has been found");
}else{
    System.out.println("Product has been not found");
}
if(employee2._searchProduct(company, branch4, product4, product4.getChoice()) == true){
    System.out.println("Product has been found");
}else{
    System.out.println("Product has been not found");
}
```

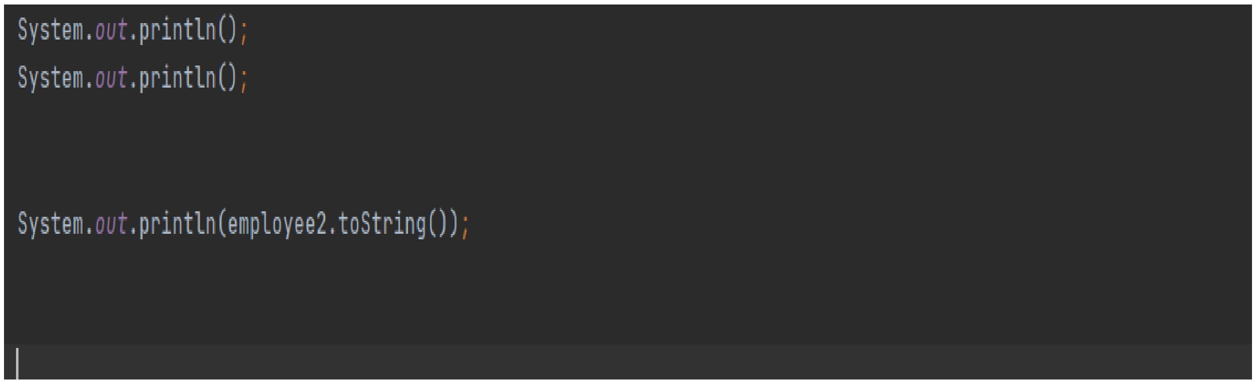


```
Product has been found
Product has been not found
```

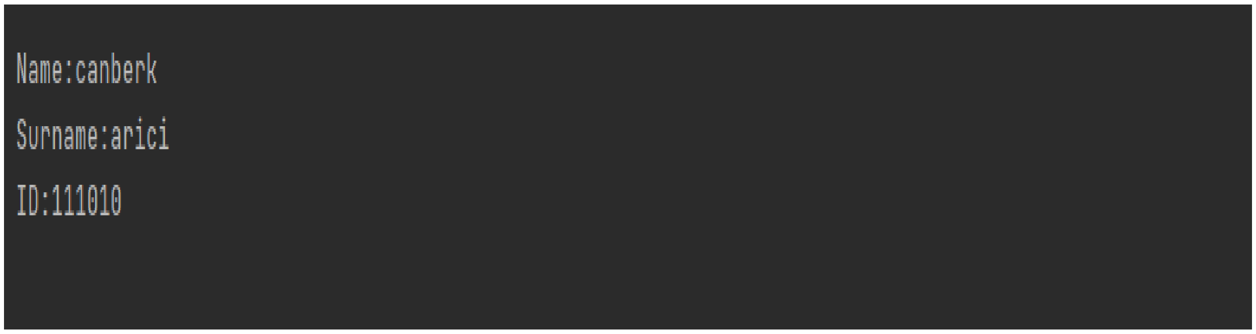
***5 times product 6’s product has been added to branch 2. _searchProduct method shows that product6’ s product is in branch2.**

***_searchProduct method shows that product4’s product is not inside of the company**

toString method



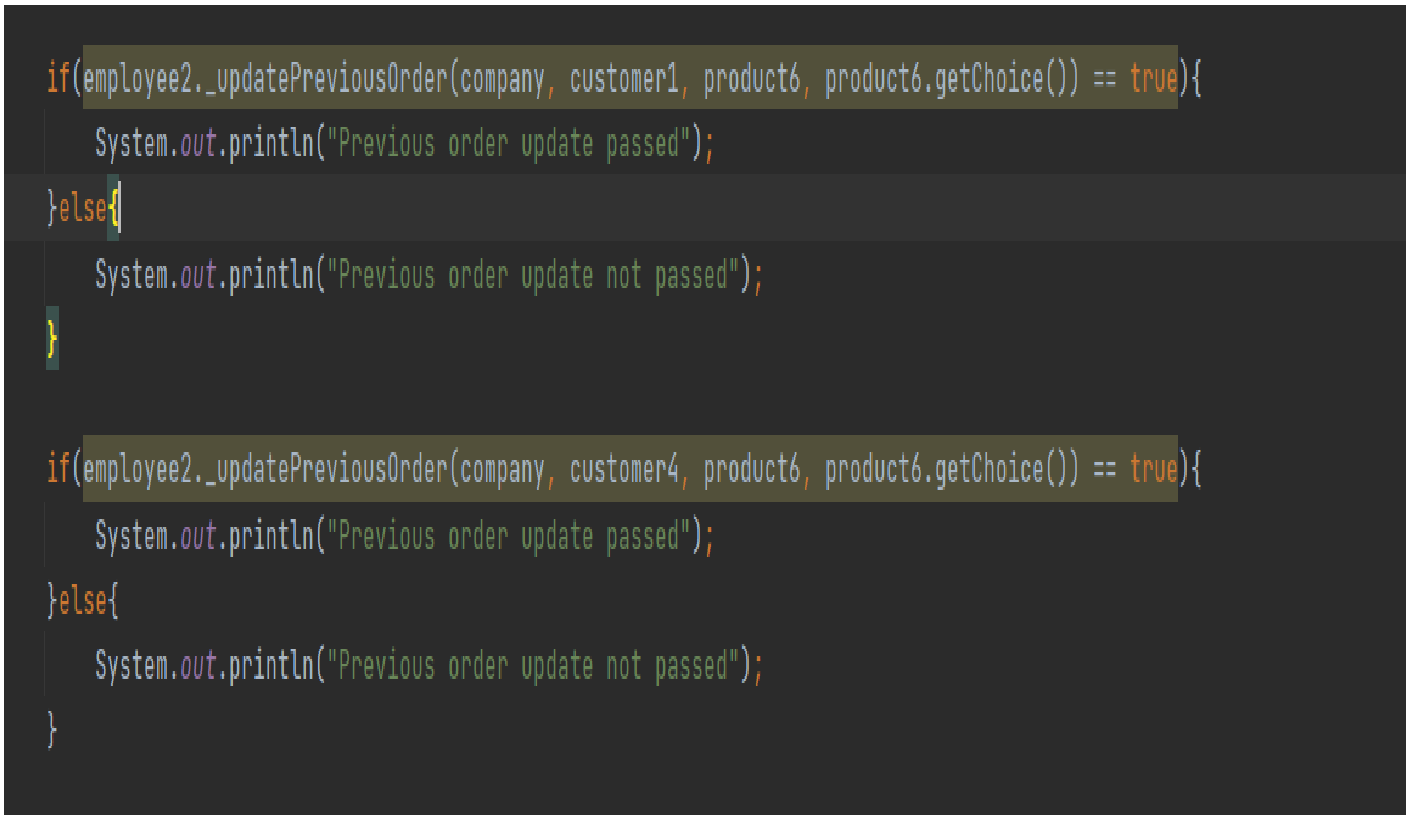
```
System.out.println();
System.out.println();


System.out.println(employee2.toString());
```



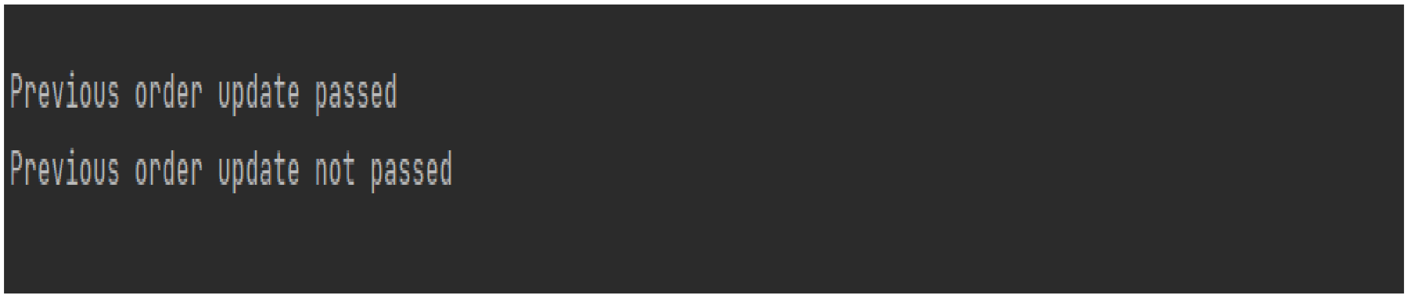
```
Name:canberk
Surname:arici
ID:111010
```

***Information about employee**

Update Previous Order



```
if(employee2._updatePreviousOrder(company, customer1, product6, product6.getChoice()) == true){
    System.out.println("Previous order update passed");
}else{
    System.out.println("Previous order update not passed");
}

if(employee2._updatePreviousOrder(company, customer4, product6, product6.getChoice()) == true){
    System.out.println("Previous order update passed");
}else{
    System.out.println("Previous order update not passed");
}
```



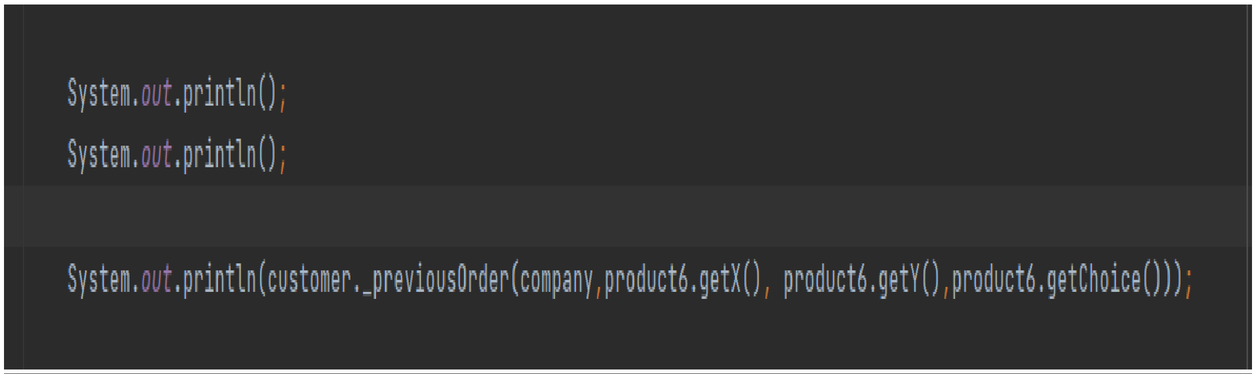
```
Previous order update passed
Previous order update not passed
```

***It adds to new product to customer.**

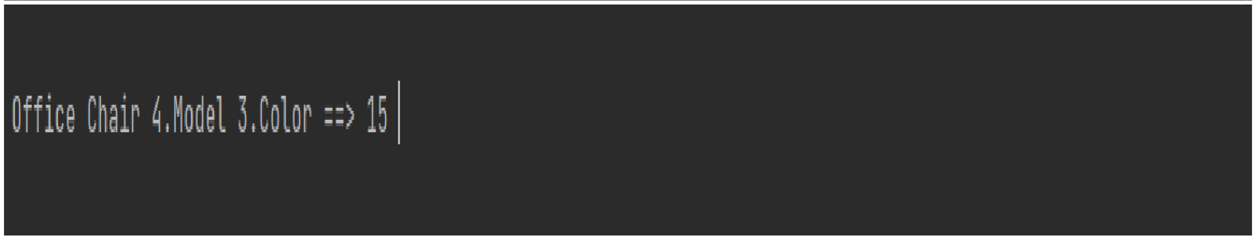
***It does not add any product to customer if branch is not in the list**

Customer Class

Access Previous Order

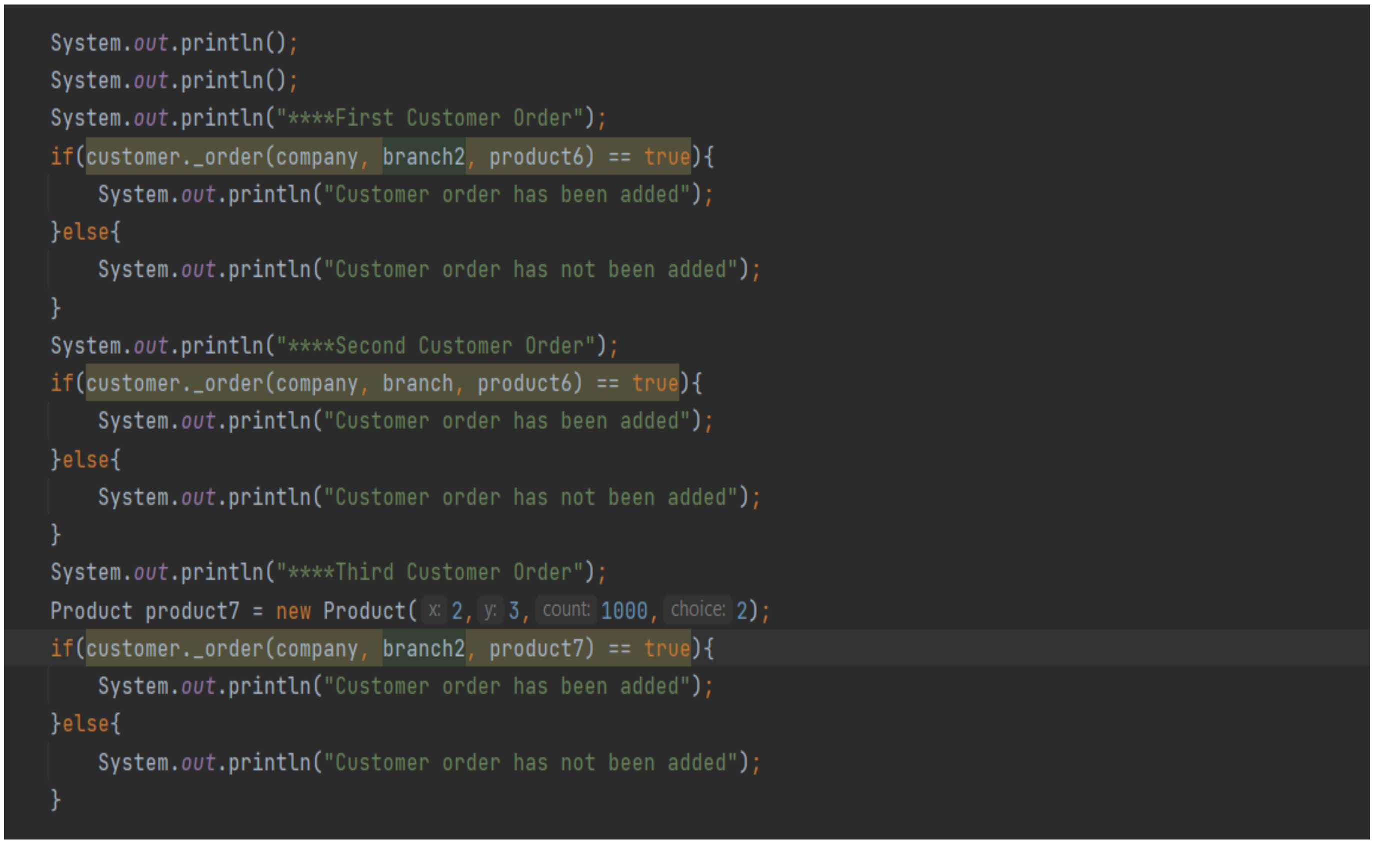


```
System.out.println();
System.out.println();


System.out.println(customer._previousOrder(company,product6.getX(), product6.getY(),product6.getChoice()));
```

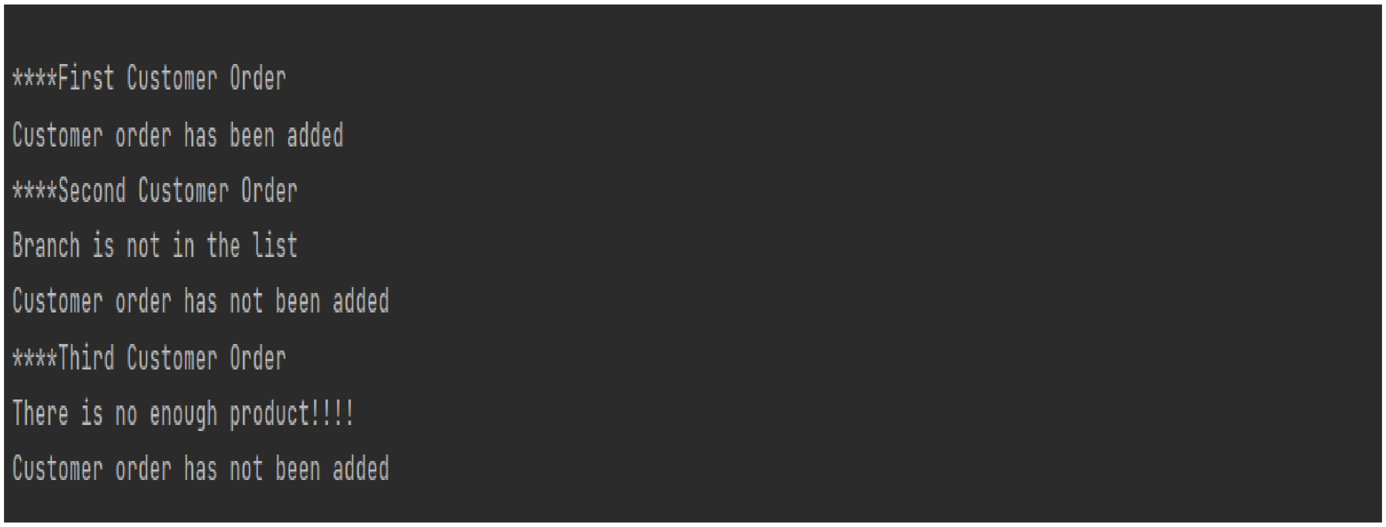


```
Office Chair 4.Model 3.Color ==> 15
```

Order Product



```java
System.out.println();
System.out.println();
System.out.println("****First Customer Order");
if(customer._order(company, branch2, product6) == true){
    System.out.println("Customer order has been added");
}else{
    System.out.println("Customer order has not been added");
}
System.out.println("****Second Customer Order");
if(customer._order(company, branch, product6) == true){
    System.out.println("Customer order has been added");
}else{
    System.out.println("Customer order has not been added");
}
System.out.println("****Third Customer Order");
Product product7 = new Product( x: 2, y: 3, count: 1000, choice: 2);
if(customer._order(company, branch2, product7) == true){
    System.out.println("Customer order has been added");
}else{
    System.out.println("Customer order has not been added");
}
```

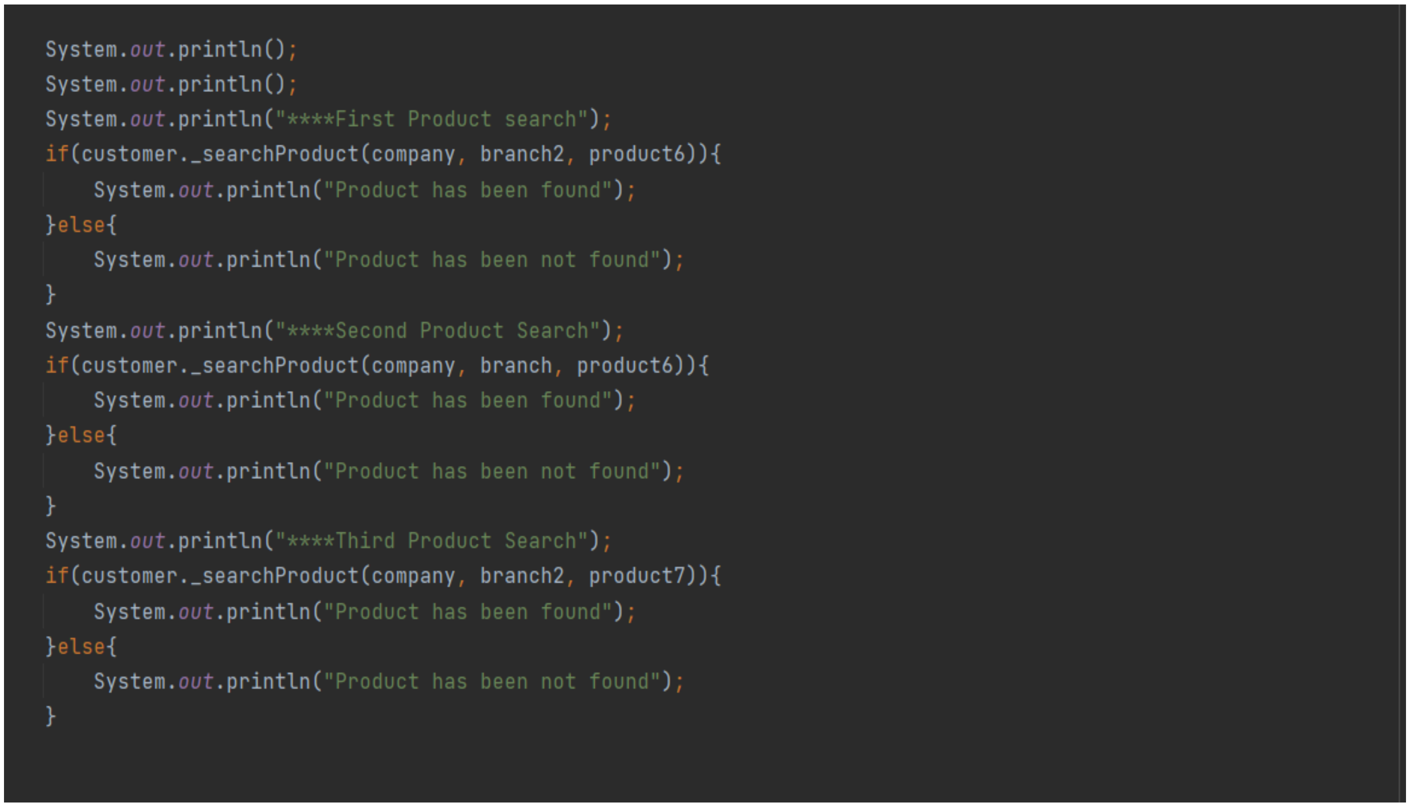

```
****First Customer Order
Customer order has been added
****Second Customer Order
Branch is not in the list
Customer order has not been added
****Third Customer Order
There is no enough product!!!!
Customer order has not been added
```

***Created product has been added to customer and removed from branch.**

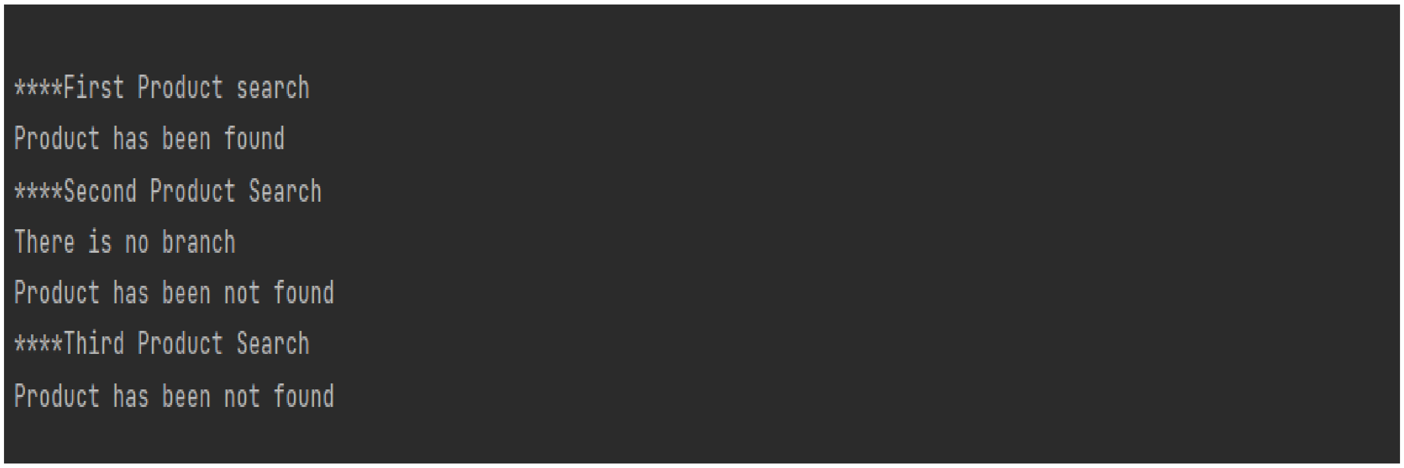
***If branch is not in the list. It does not happen**

***If product number is not enough for the customer. Admin has to add new product to branch**

Search Product



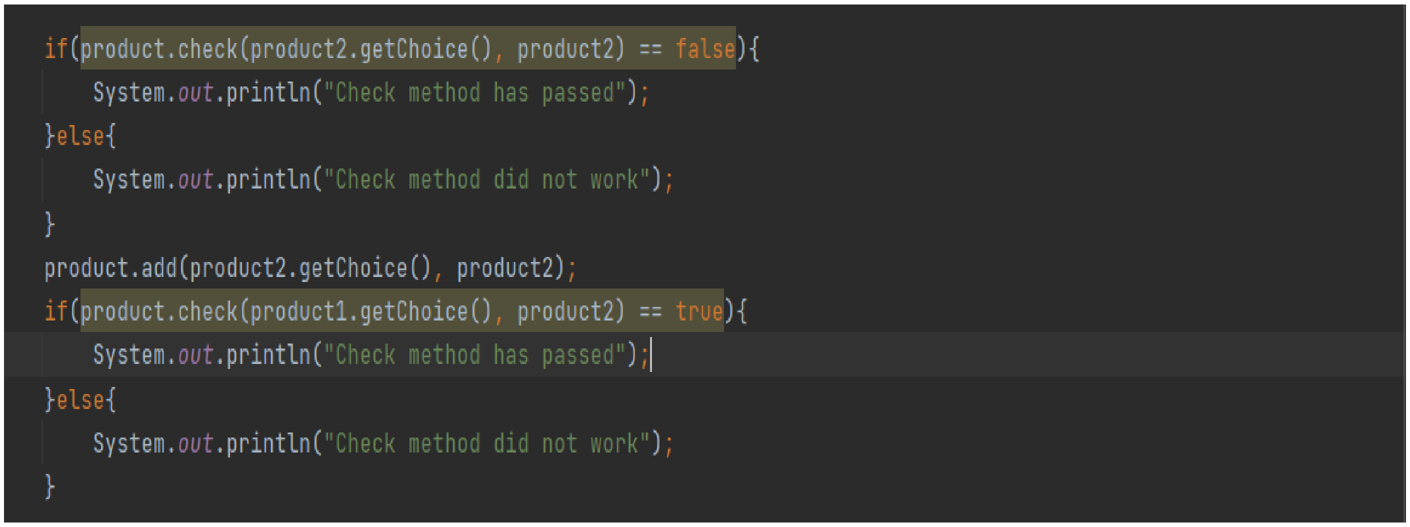
```
System.out.println();
System.out.println();
System.out.println("****First Product search");
if(customer._searchProduct(company, branch2, product6)){
    System.out.println("Product has been found");
}else{
    System.out.println("Product has been not found");
}
System.out.println("****Second Product Search");
if(customer._searchProduct(company, branch, product6)){
    System.out.println("Product has been found");
}else{
    System.out.println("Product has been not found");
}
System.out.println("****Third Product Search");
if(customer._searchProduct(company, branch2, product7)){
    System.out.println("Product has been found");
}else{
    System.out.println("Product has been not found");
}
```



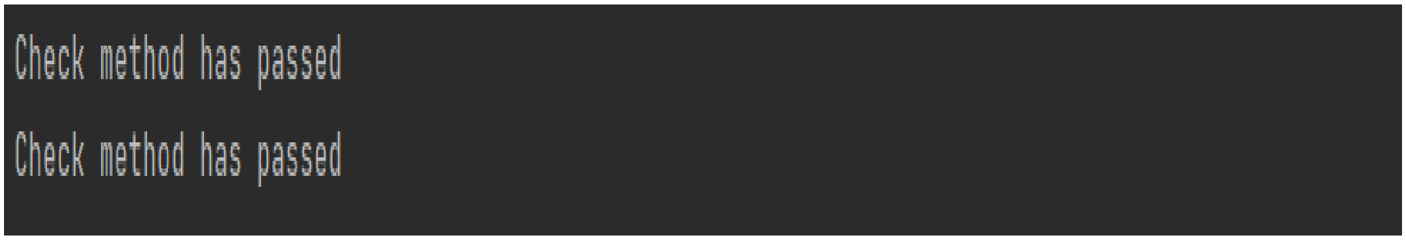
```
****First Product search
Product has been found
****Second Product Search
There is no branch
Product has been not found
****Third Product Search
Product has been not found
```

Product Class:

Check Product



```
if(product.check(product2.getChoice(), product2) == false){
    System.out.println("Check method has passed");
}else{
    System.out.println("Check method did not work");
}
product.add(product2.getChoice(), product2);
if(product.check(product1.getChoice(), product2) == true){
    System.out.println("Check method has passed");
}else{
    System.out.println("Check method did not work");
}
```

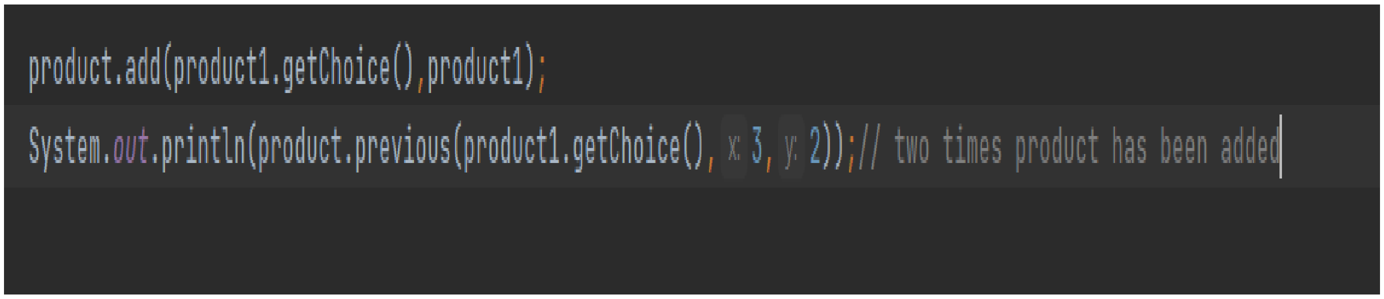


```
Check method has passed

Check method has passed
```

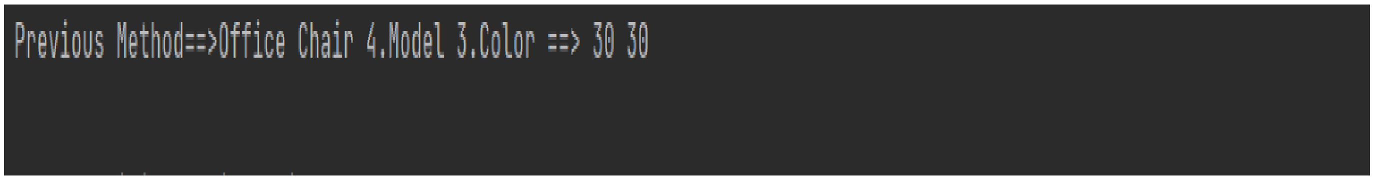
***Compares if product has contains product 2 . It does not contain product 2 so it returns false**

*** Compares if product has contains product 1 . It does contain product 1 so it returns true**

Previous Product



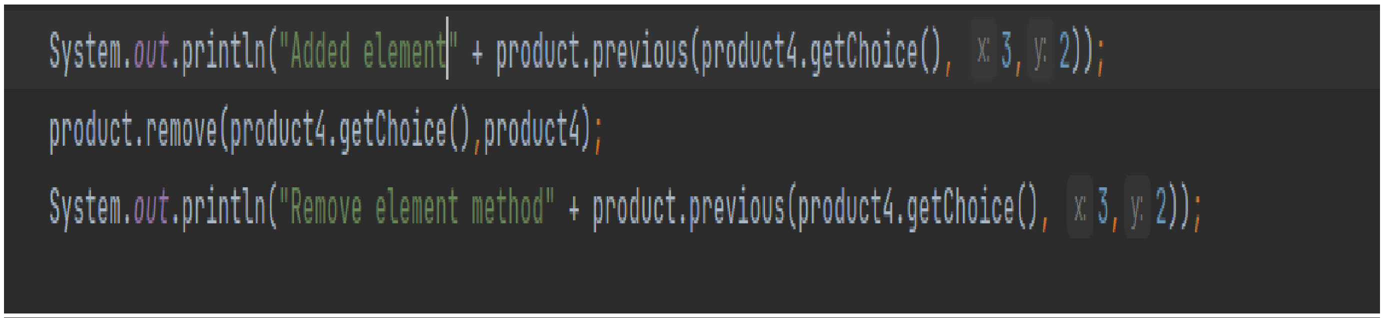
```
product.add(product1.getChoice(),product1);
System.out.println(product.previous(product1.getChoice(), x: 3, y: 2));// two times product has been added
```



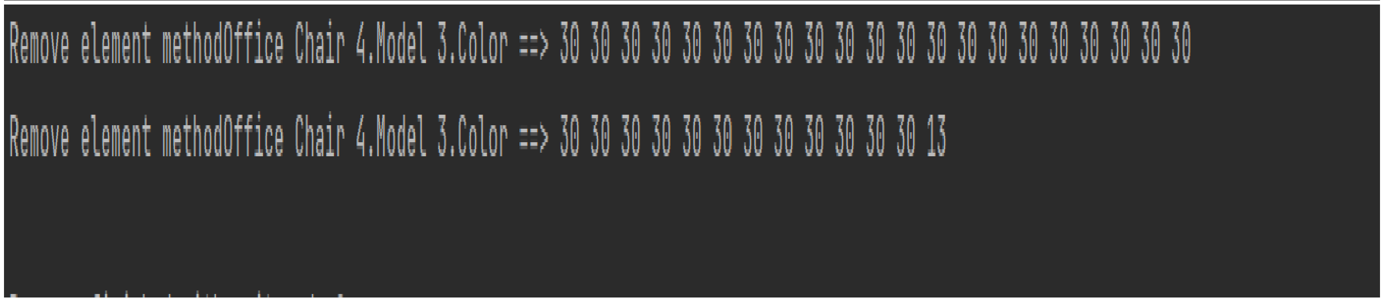
```
Previous Method==>Office Chair 4.Model 3.Color ==> 30 30
```

***It shows all the added products.**

Remove



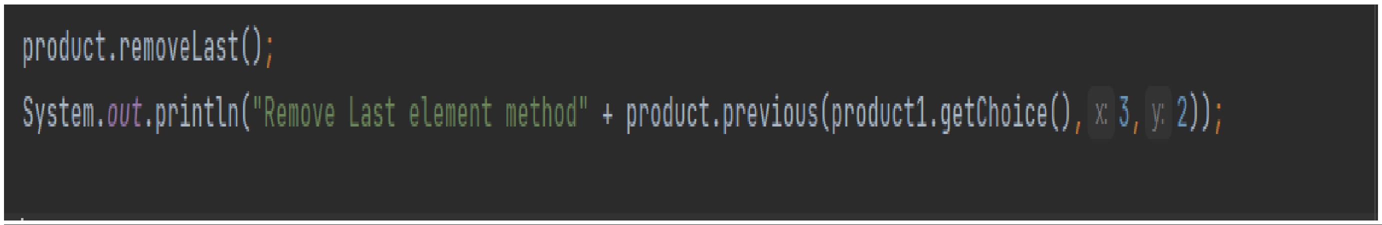
```
System.out.println("Added element" + product.previous(product4.getChoice(),  x: 3, y: 2));
product.remove(product4.getChoice(),product4);
System.out.println("Remove element method" + product.previous(product4.getChoice(),  x: 3, y: 2));
```



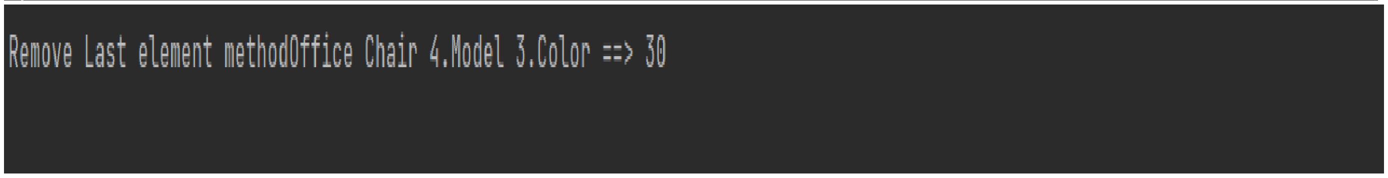
```
Remove element methodOffice Chair 4.Model 3.Color ==> 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30

Remove element methodOffice Chair 4.Model 3.Color ==> 30 30 30 30 30 30 30 30 30 30 30 30 13
```

***257 Chair removed from product.**

Remove Last



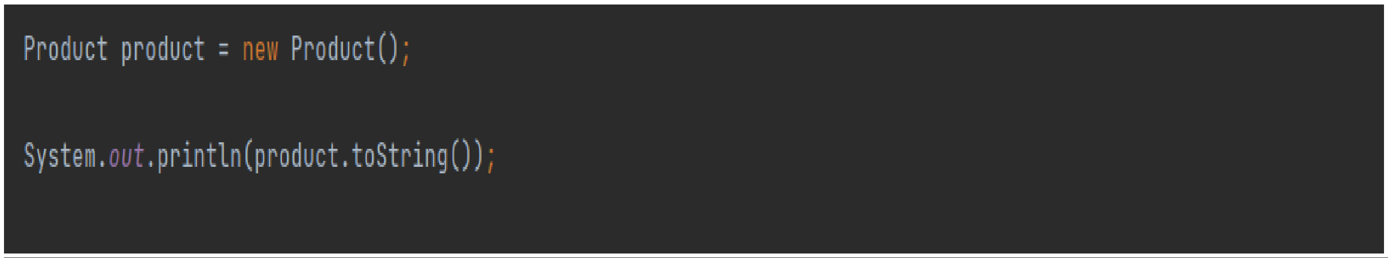
```
product.removeLast();
System.out.println("Remove Last element method" + product.previous(product1.getChoice(), x: 3, y: 2));
```



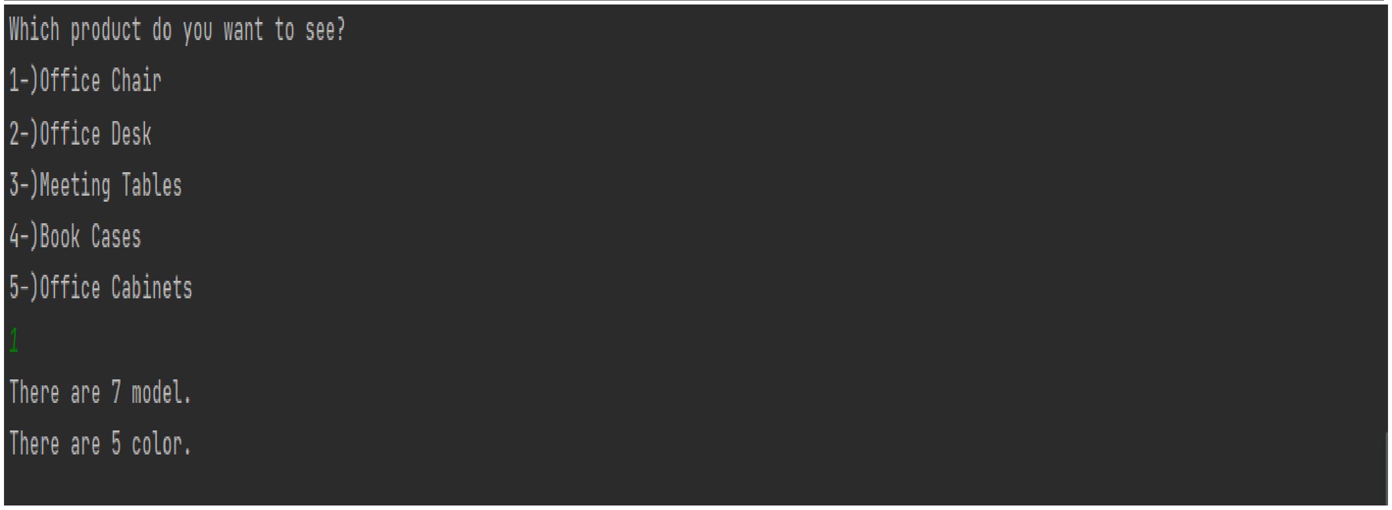
```
Remove Last element methodOffice Chair 4.Model 3.Color ==> 30
```

***Removes last added element in the product.**

toString



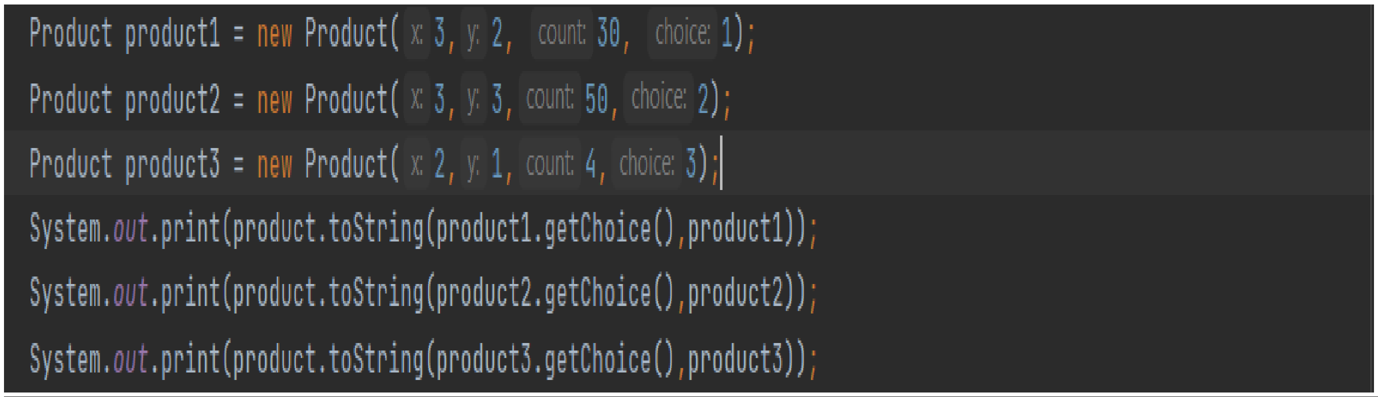
```
Product product = new Product();

System.out.println(product.toString());
```



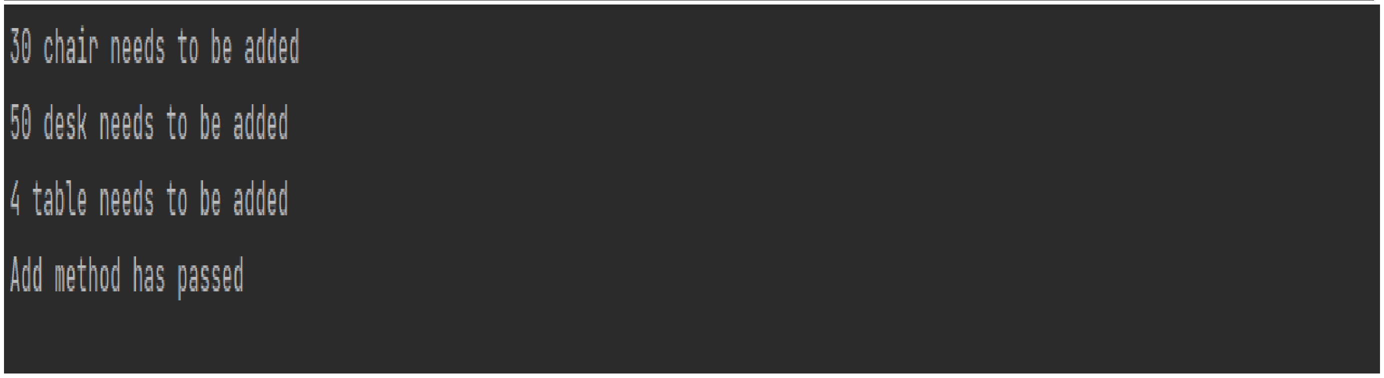
```
Which product do you want to see?
1-)Office Chair
2-)Office Desk
3-)Meeting Tables
4-)Book Cases
5-)Office Cabinets
1
There are 7 model.
There are 5 color.
```

***It just shows model number and color number in the products.**

toString(product)



```java
Product product1 = new Product( x: 3, y: 2,  count: 30,  choice: 1);
Product product2 = new Product( x: 3, y: 3, count: 50, choice: 2);
Product product3 = new Product( x: 2, y: 1, count: 4, choice: 3);
System.out.print(product.toString(product1.getChoice(),product1));
System.out.print(product.toString(product2.getChoice(),product2));
System.out.print(product.toString(product3.getChoice(),product3));
```



```
30 chair needs to be added

50 desk needs to be added

4 table needs to be added

Add method has passed
```

***It shows how many office product needs to be added to product object.**