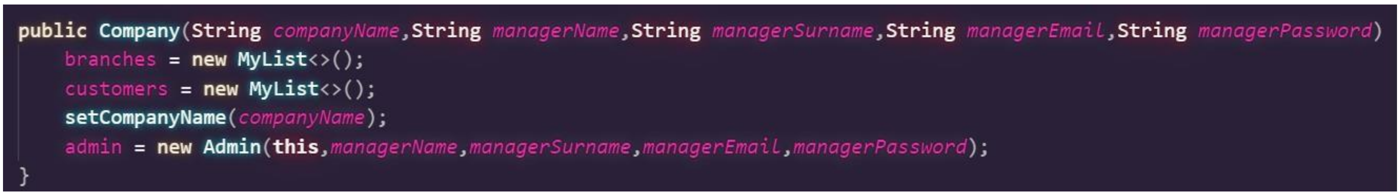
GIT Department of Computer Engineering
CSE 222/505 - Spring 2021
Homework #3 Report

ATAKAN ALTIN
1801042668

1. SYSTEM REQUIREMENTS

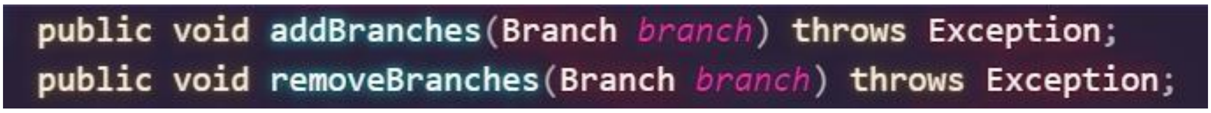
First things first, we should create a **company** object.



```
public Company(String companyName,String managerName,String managerSurname,String managerEmail,String managerPassword)
    branches = new MyList<>();
    customers = new MyList<>();
    setCompanyName(companyName);
    admin = new Admin(this,managerName,managerSurname,managerEmail,managerPassword);
}
```

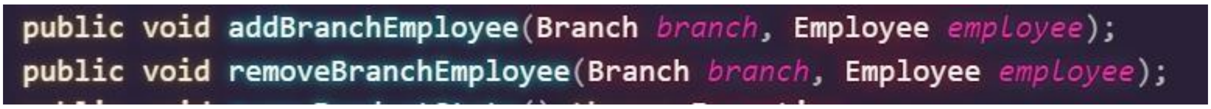
Company constructor initializes company’s name and authorizes administrator.

Administrator can add/remove branches and employees.



```
public void addBranches(Branch branch) throws Exception;
public void removeBranches(Branch branch) throws Exception;
```

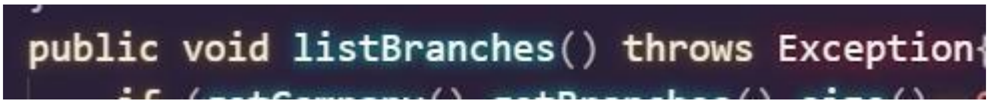
Add/remove branch functions takes branch objects as a parameter.



```
public void addBranchEmployee(Branch branch, Employee employee);
public void removeBranchEmployee(Branch branch, Employee employee);
```

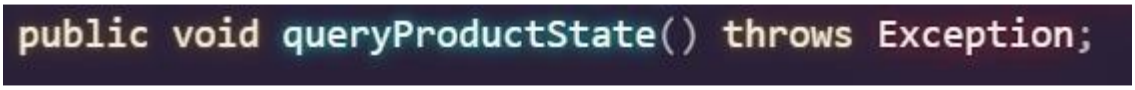
When the admin adding a new employee, he/she must know which store to add it to.

Add/remove employee functions also takes employee objects as a parameter.



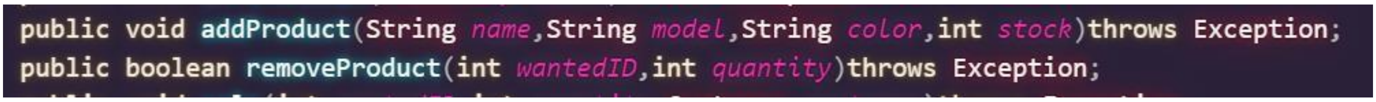
```
public void listBranches() throws Exception{
```

Admin also can list company’s all branches and can view if there are products that need to be supplied.



```
public void queryProductState() throws Exception;
```

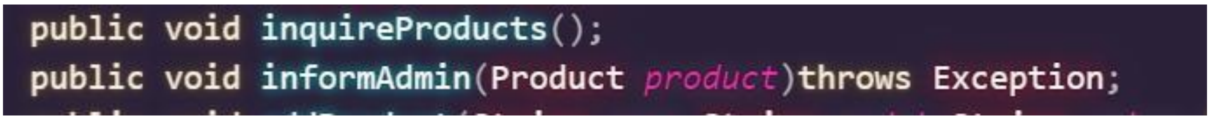
Employees can add/remove products to his/her company.



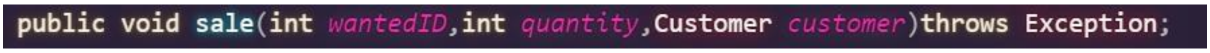
```
public void addProduct(String name,String model,String color,int stock)throws Exception;
public boolean removeProduct(int wantedID,int quantity)throws Exception;
```

Employees can sale products to customers in store. They add products to customer’s order history. If customer is not a part of automation system then employee can add customer to the system. It takes product’s id, quantity and customer object.

They also inquire all products and they can observe products status. They can inform the admin about product’s status.

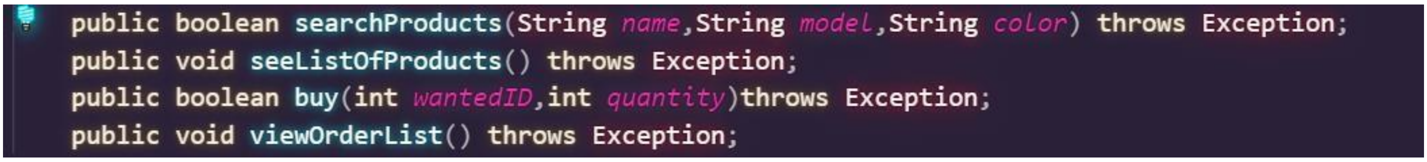


```
public void inquireProducts();
public void informAdmin(Product product)throws Exception;
```



```
public void sale(int wantedID,int quantity,Customer customer)throws Exception;
```

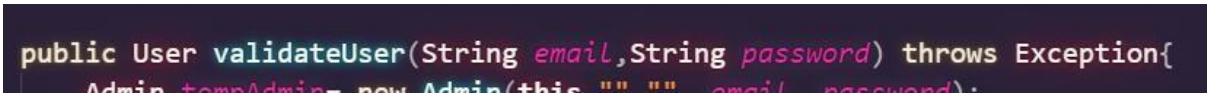
Customer’s able to subscribe to the automation system. It can use sign up function after the customer’s object is created.



```
public boolean searchProducts(String name,String model,String color) throws Exception;
public void seeListOfProducts() throws Exception;
public boolean buy(int wantedID,int quantity)throws Exception;
public void viewOrderList() throws Exception;
```

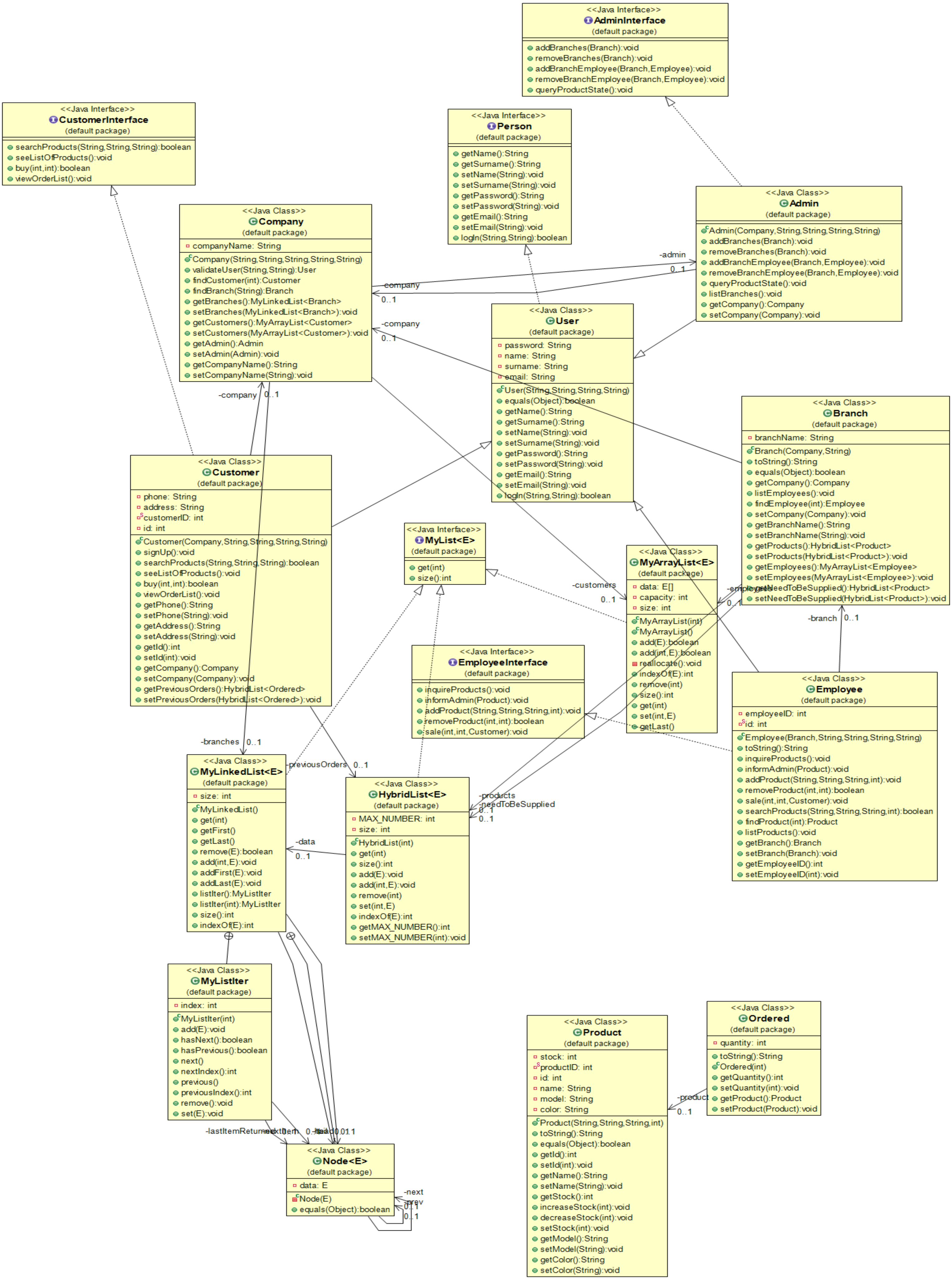
They can search individual products, see all of the products in the system, buy products and view order history.

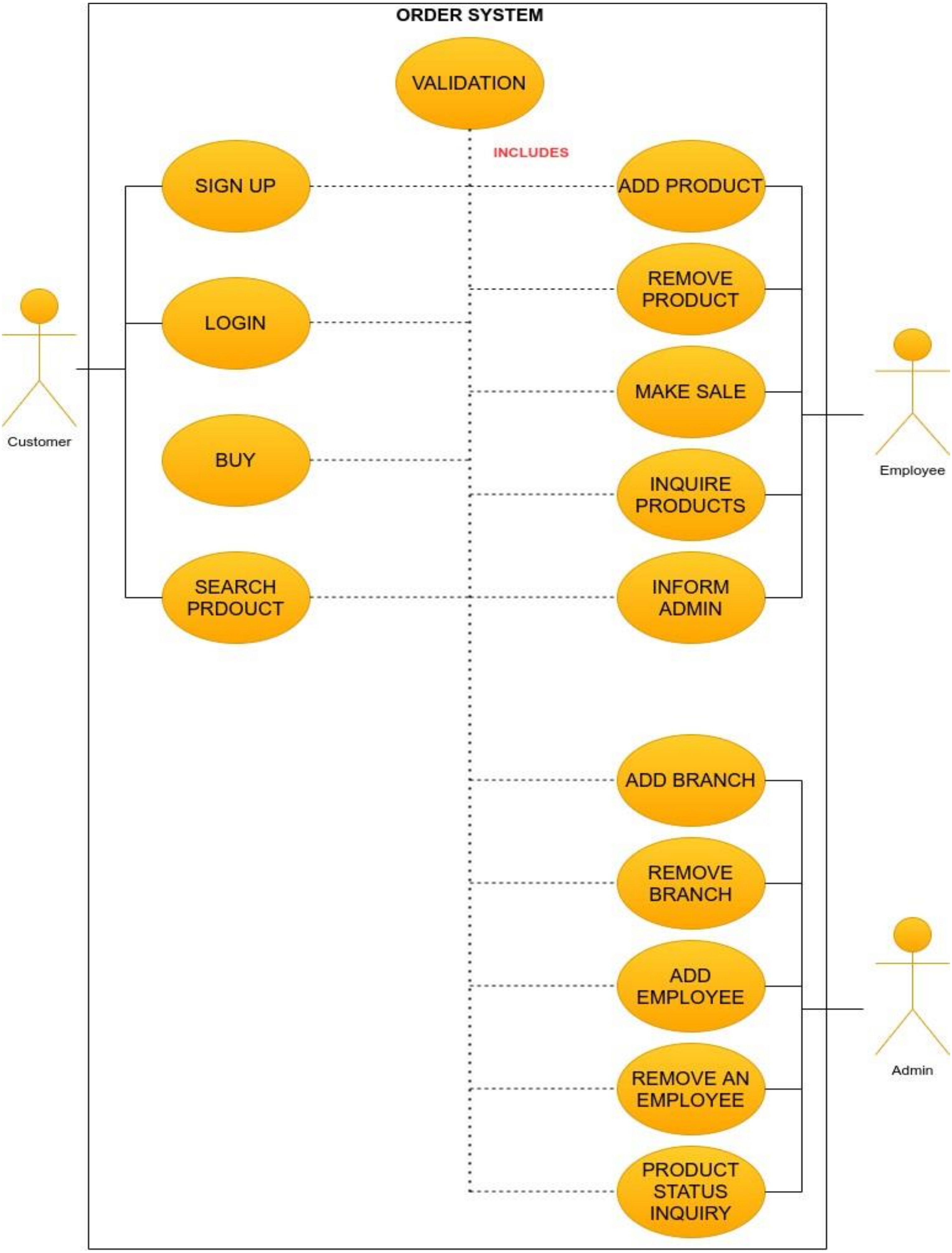
All users can login to the system and it is checked by company’s validation function.



```
public User validateUser(String email,String password) throws Exception{
    Admin tempAdmin= new Admin(this,"","", email, password);
```

2. USE CASE AND CLASS DIAGRAMS

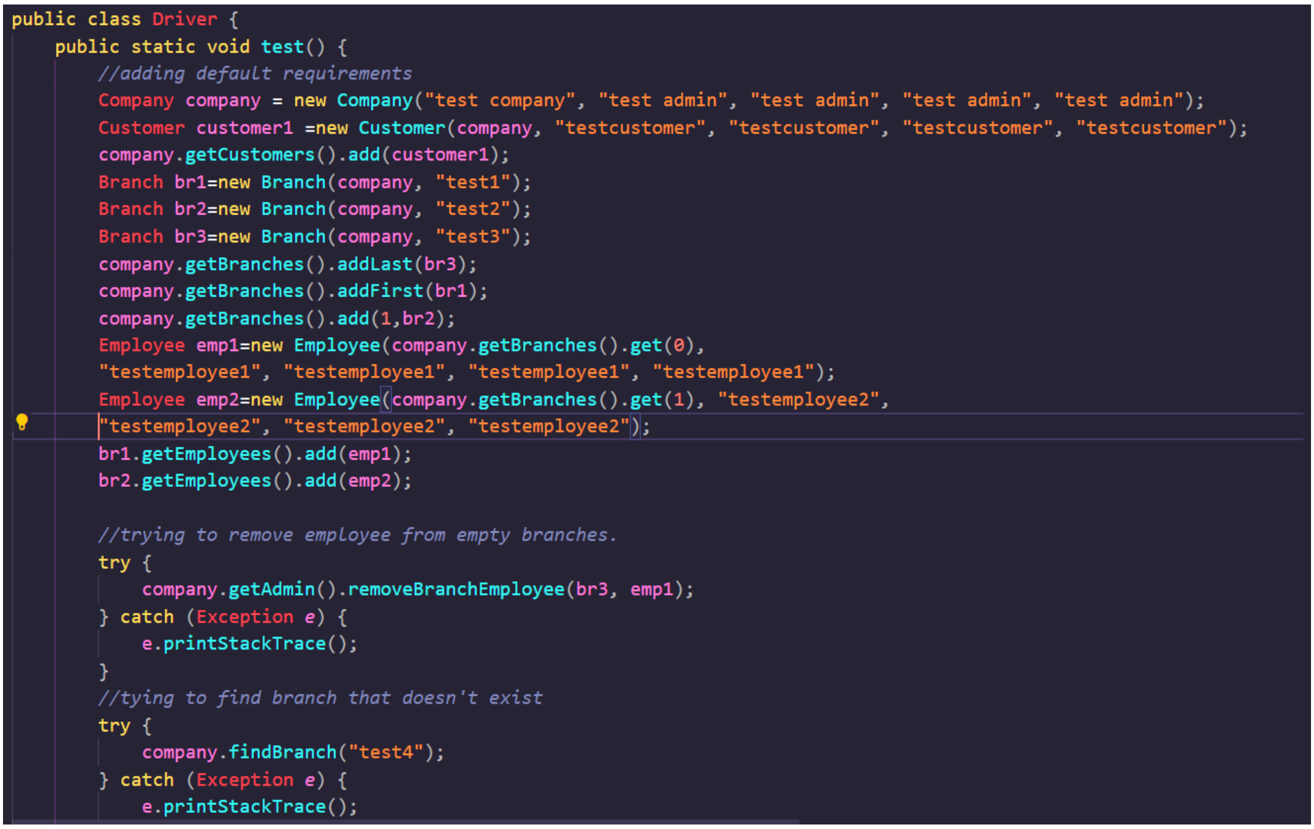



ORDER SYSTEM
VALIDATION
INCLUDES
SIGN UP
LOGIN
BUY
SEARCH PRDOUCT
Customer
ADD PRODUCT
REMOVE PRODUCT
MAKE SALE
INQUIRE PRODUCTS
INFORM ADMIN
Employee
ADD BRANCH
REMOVE BRANCH
ADD EMPLOYEE
REMOVE AN EMPLOYEE
PRODUCT STATUS INQUIRY
Admin

3. PROBLEM SOLUTION APPROACH

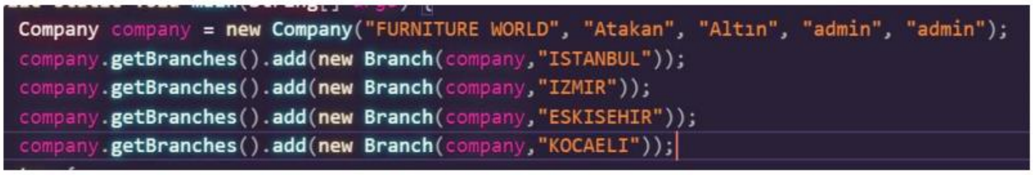
Since we need to store the data, we must use data structures. In this problem i have used some of data structures such as MyArrayList, MyLinkedList and HybridList. I've implemented each of these structures individually and combine them with MyList Interface. I tried to implement functions of these structures with best time complexity for the sake of efficiency. I create Company class to record all system data easily. I combined customers,employees and admins in the User class which implements Person Interface.

4. TEST CASES

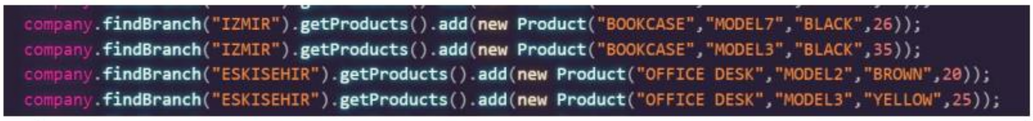


```java
public class Driver {
    public static void test() {
        //adding default requirements
        Company company = new Company("test company", "test admin", "test admin", "test admin", "test admin");
        Customer customer1 =new Customer(company, "testcustomer", "testcustomer", "testcustomer", "testcustomer");
        company.getCustomers().add(customer1);
        Branch br1=new Branch(company, "test1");
        Branch br2=new Branch(company, "test2");
        Branch br3=new Branch(company, "test3");
        company.getBranches().addLast(br3);
        company.getBranches().addFirst(br1);
        company.getBranches().add(1,br2);
        Employee emp1=new Employee(company.getBranches().get(0),
        "testemployee1", "testemployee1", "testemployee1", "testemployee1");
        Employee emp2=new Employee(company.getBranches().get(1), "testemployee2",
        "testemployee2", "testemployee2", "testemployee2");
        br1.getEmployees().add(emp1);
        br2.getEmployees().add(emp2);

        //trying to remove employee from empty branches.
        try {
            company.getAdmin().removeBranchEmployee(br3, emp1);
        } catch (Exception e) {
            e.printStackTrace();
        }
        //tying to find branch that doesn't exist
        try {
            company.findBranch("test4");
        } catch (Exception e) {
            e.printStackTrace();
```

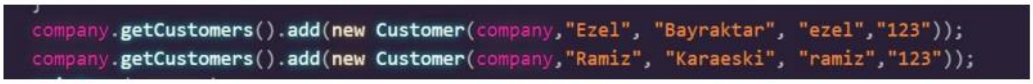
I create one company, one customer,3 branches, 2 employees,3 products to test the entire system. I tried to test all cases of all functions in the test function. I also create menu system for you to test all functions yourselves.



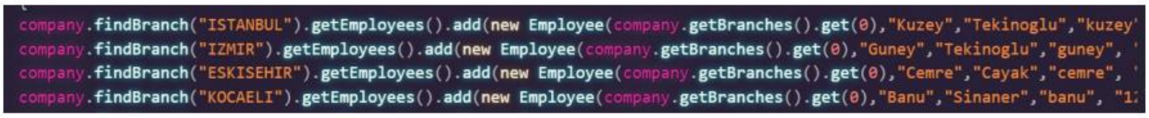
```java
Company company = new Company("FURNITURE WORLD", "Atakan", "Altın", "admin", "admin");
company.getBranches().add(new Branch(company,"ISTANBUL"));
company.getBranches().add(new Branch(company,"IZMIR"));
company.getBranches().add(new Branch(company,"ESKISEHIR"));
company.getBranches().add(new Branch(company,"KOCAELI"));
```



```java
company.findBranch("IZMIR").getProducts().add(new Product("BOOKCASE","MODEL7","BLACK",26));
company.findBranch("IZMIR").getProducts().add(new Product("BOOKCASE","MODEL3","BLACK",35));
company.findBranch("ESKISEHIR").getProducts().add(new Product("OFFICE DESK","MODEL2","BROWN",20));
company.findBranch("ESKISEHIR").getProducts().add(new Product("OFFICE DESK","MODEL3","YELLOW",25));
```

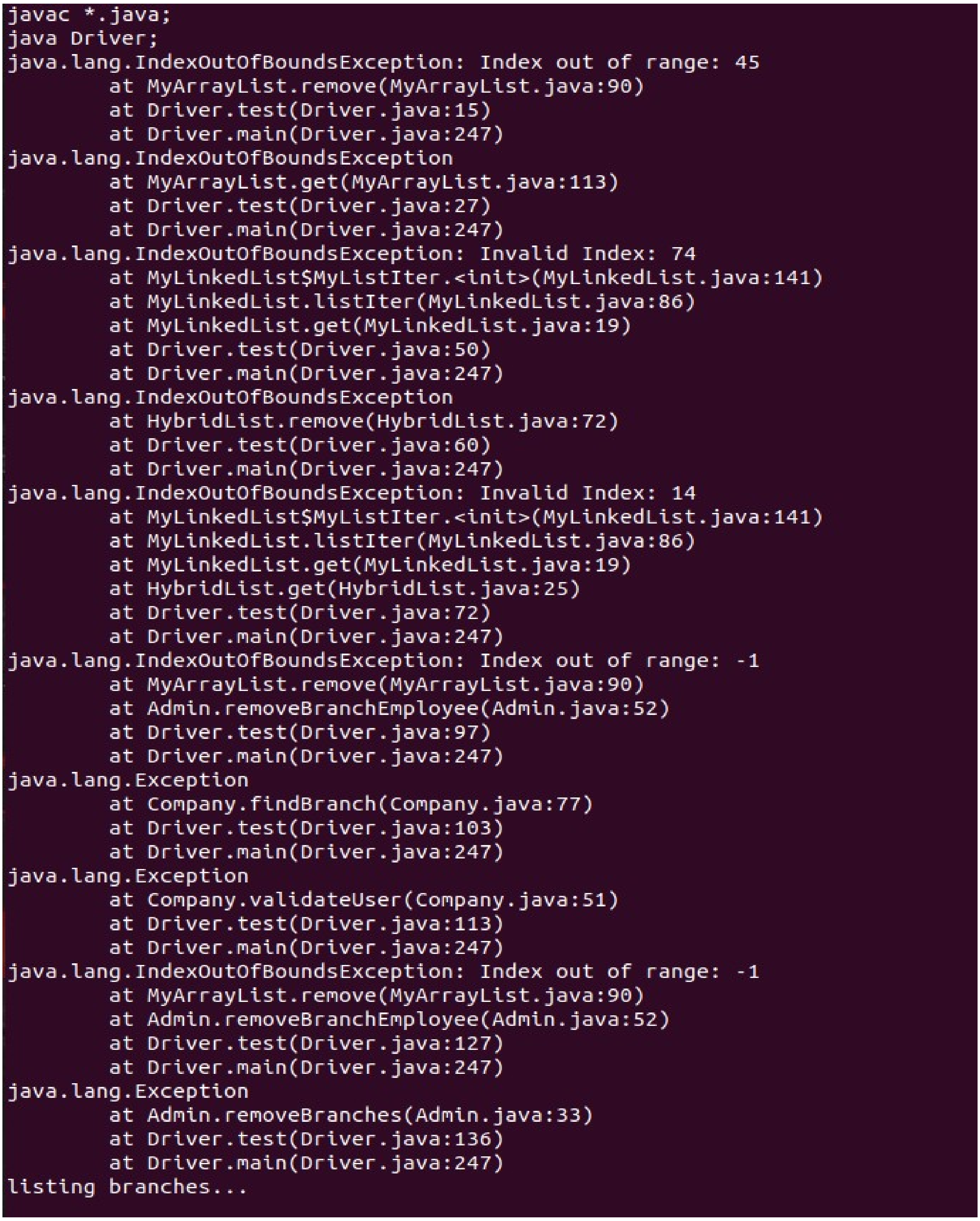


```java
company.getCustomers().add(new Customer(company,"Ezel", "Bayraktar", "ezel","123"));
company.getCustomers().add(new Customer(company,"Ramiz", "Karaeski", "ramiz","123"));
```



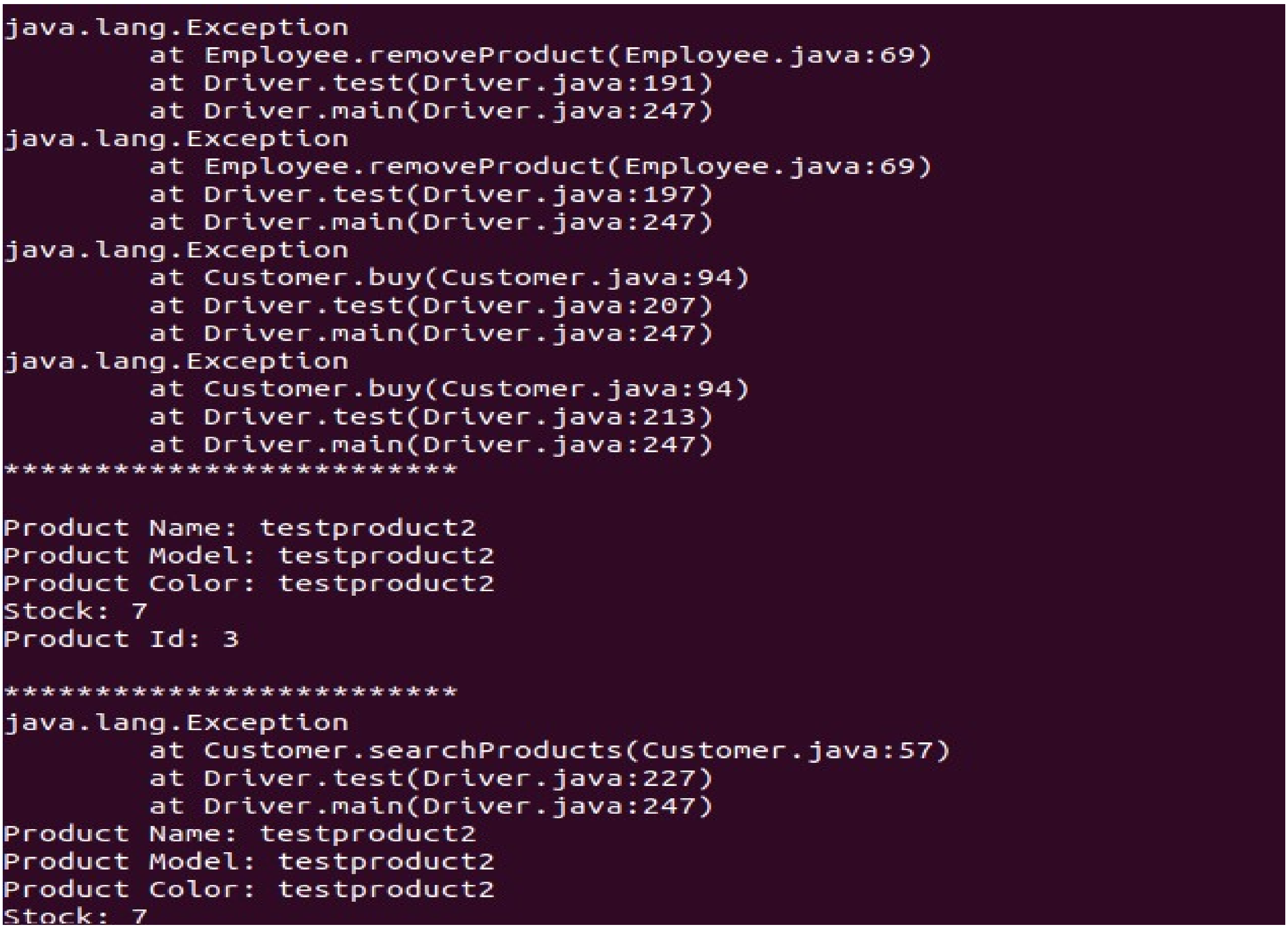
```java
company.findBranch("ISTANBUL").getEmployees().add(new Employee(company.getBranches().get(0),"Kuzey","Tekinoglu","kuzey'
company.findBranch("IZMIR").getEmployees().add(new Employee(company.getBranches().get(0),"Guney","Tekinoglu","guney",
company.findBranch("ESKISEHIR").getEmployees().add(new Employee(company.getBranches().get(0),"Cemre","Cayak","cemre",
company.findBranch("KOCAELI").getEmployees().add(new Employee(company.getBranches().get(0),"Banu","Sinaner","banu", "1:
```

5.RUNNING AND TEST RESULTS

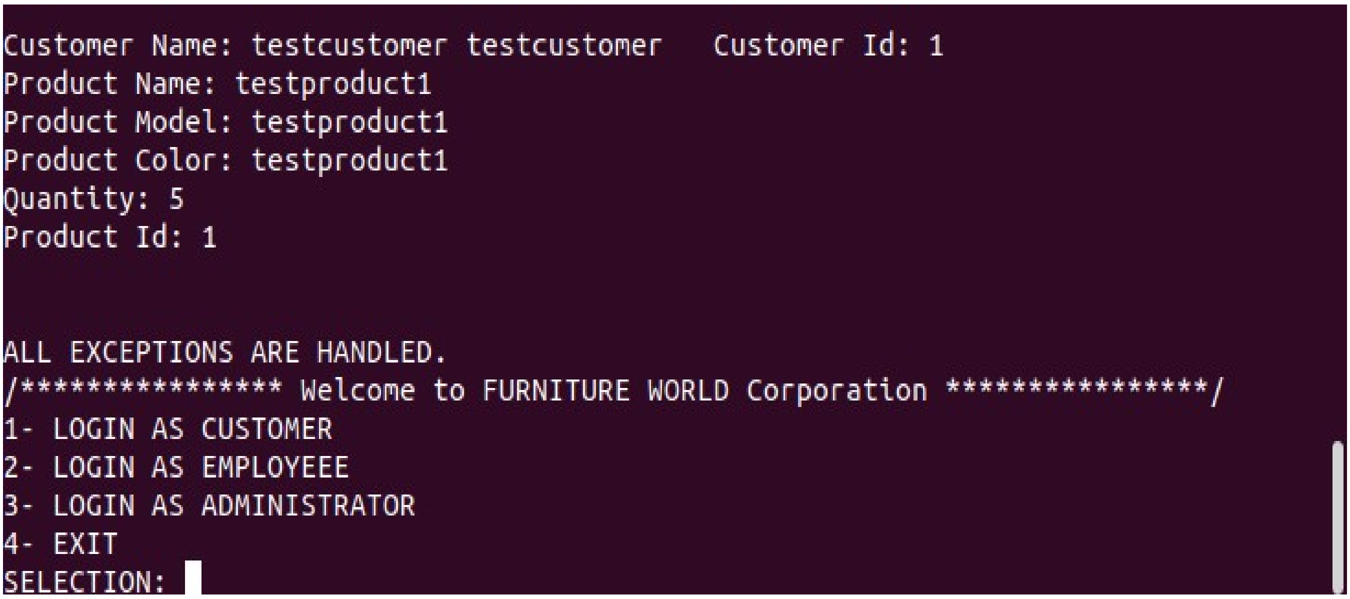


```
javac *.java;
java Driver;
java.lang.IndexOutOfBoundsException: Index out of range: 45
        at MyArrayList.remove(MyArrayList.java:90)
        at Driver.test(Driver.java:15)
        at Driver.main(Driver.java:247)
java.lang.IndexOutOfBoundsException
        at MyArrayList.get(MyArrayList.java:113)
        at Driver.test(Driver.java:27)
        at Driver.main(Driver.java:247)
java.lang.IndexOutOfBoundsException: Invalid Index: 74
        at MyLinkedList$MyListIter.<init>(MyLinkedList.java:141)
        at MyLinkedList.listIter(MyLinkedList.java:86)
        at MyLinkedList.get(MyLinkedList.java:19)
        at Driver.test(Driver.java:50)
        at Driver.main(Driver.java:247)
java.lang.IndexOutOfBoundsException
        at HybridList.remove(HybridList.java:72)
        at Driver.test(Driver.java:60)
        at Driver.main(Driver.java:247)
java.lang.IndexOutOfBoundsException: Invalid Index: 14
        at MyLinkedList$MyListIter.<init>(MyLinkedList.java:141)
        at MyLinkedList.listIter(MyLinkedList.java:86)
        at MyLinkedList.get(MyLinkedList.java:19)
        at HybridList.get(HybridList.java:25)
        at Driver.test(Driver.java:72)
        at Driver.main(Driver.java:247)
java.lang.IndexOutOfBoundsException: Index out of range: -1
        at MyArrayList.remove(MyArrayList.java:90)
        at Admin.removeBranchEmployee(Admin.java:52)
        at Driver.test(Driver.java:97)
        at Driver.main(Driver.java:247)
java.lang.Exception
        at Company.findBranch(Company.java:77)
        at Driver.test(Driver.java:103)
        at Driver.main(Driver.java:247)
java.lang.Exception
        at Company.validateUser(Company.java:51)
        at Driver.test(Driver.java:113)
        at Driver.main(Driver.java:247)
java.lang.IndexOutOfBoundsException: Index out of range: -1
        at MyArrayList.remove(MyArrayList.java:90)
        at Admin.removeBranchEmployee(Admin.java:52)
        at Driver.test(Driver.java:127)
        at Driver.main(Driver.java:247)
java.lang.Exception
        at Admin.removeBranches(Admin.java:33)
        at Driver.test(Driver.java:136)
        at Driver.main(Driver.java:247)
listing branches...
```

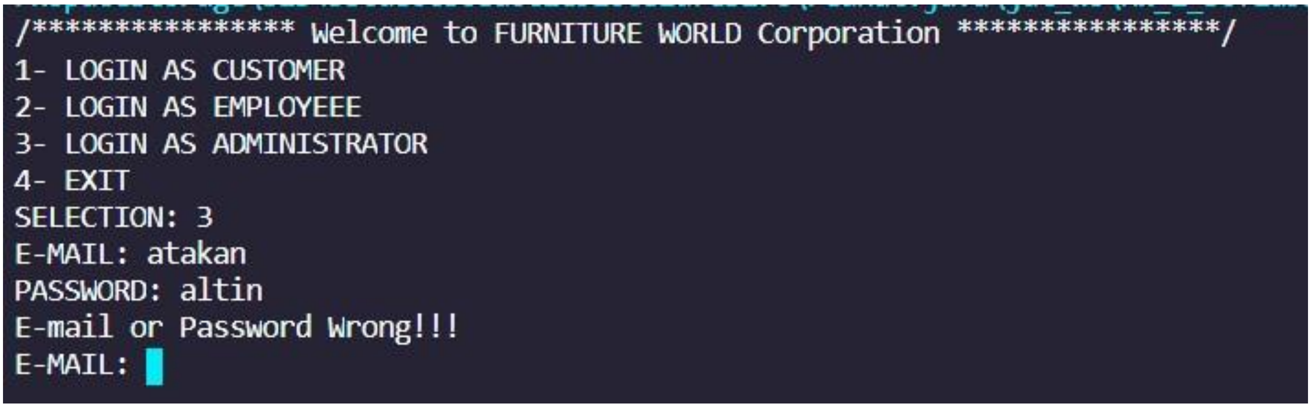
When i tried to functions of data structure and automation system i intentionally catch exceptions and handle them.



```
java.lang.Exception
        at Employee.removeProduct(Employee.java:69)
        at Driver.test(Driver.java:191)
        at Driver.main(Driver.java:247)
java.lang.Exception
        at Employee.removeProduct(Employee.java:69)
        at Driver.test(Driver.java:197)
        at Driver.main(Driver.java:247)
java.lang.Exception
        at Customer.buy(Customer.java:94)
        at Driver.test(Driver.java:207)
        at Driver.main(Driver.java:247)
java.lang.Exception
        at Customer.buy(Customer.java:94)
        at Driver.test(Driver.java:213)
        at Driver.main(Driver.java:247)
*************************

Product Name: testproduct2
Product Model: testproduct2
Product Color: testproduct2
Stock: 7
Product Id: 3

*************************
java.lang.Exception
        at Customer.searchProducts(Customer.java:57)
        at Driver.test(Driver.java:227)
        at Driver.main(Driver.java:247)
Product Name: testproduct2
Product Model: testproduct2
Product Color: testproduct2
Stock: 7
```

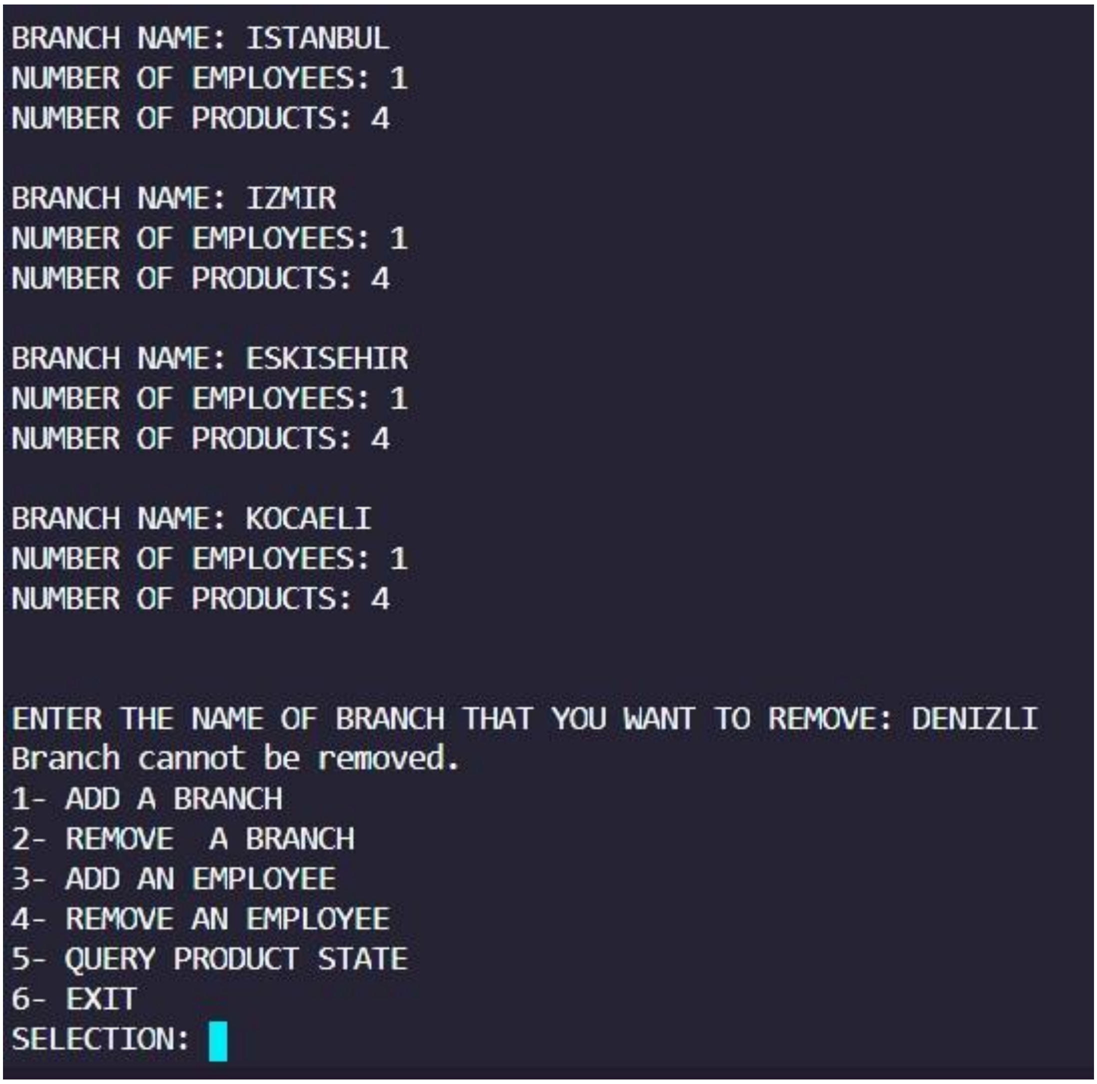


```
Customer Name: testcustomer testcustomer   Customer Id: 1
Product Name: testproduct1
Product Model: testproduct1
Product Color: testproduct1
Quantity: 5
Product Id: 1


ALL EXCEPTIONS ARE HANDLED.
/**************** Welcome to FURNITURE WORLD Corporation ****************/
1- LOGIN AS CUSTOMER
2- LOGIN AS EMPLOYEEE
3- LOGIN AS ADMINISTRATOR
4- EXIT
SELECTION: 
```



```
/**************** Welcome to FURNITURE WORLD Corporation ****************/
1- LOGIN AS CUSTOMER
2- LOGIN AS EMPLOYEEE
3- LOGIN AS ADMINISTRATOR
4- EXIT
SELECTION: 3
E-MAIL: atakan
PASSWORD: altin
E-mail or Password Wrong!!!
E-MAIL: 
```

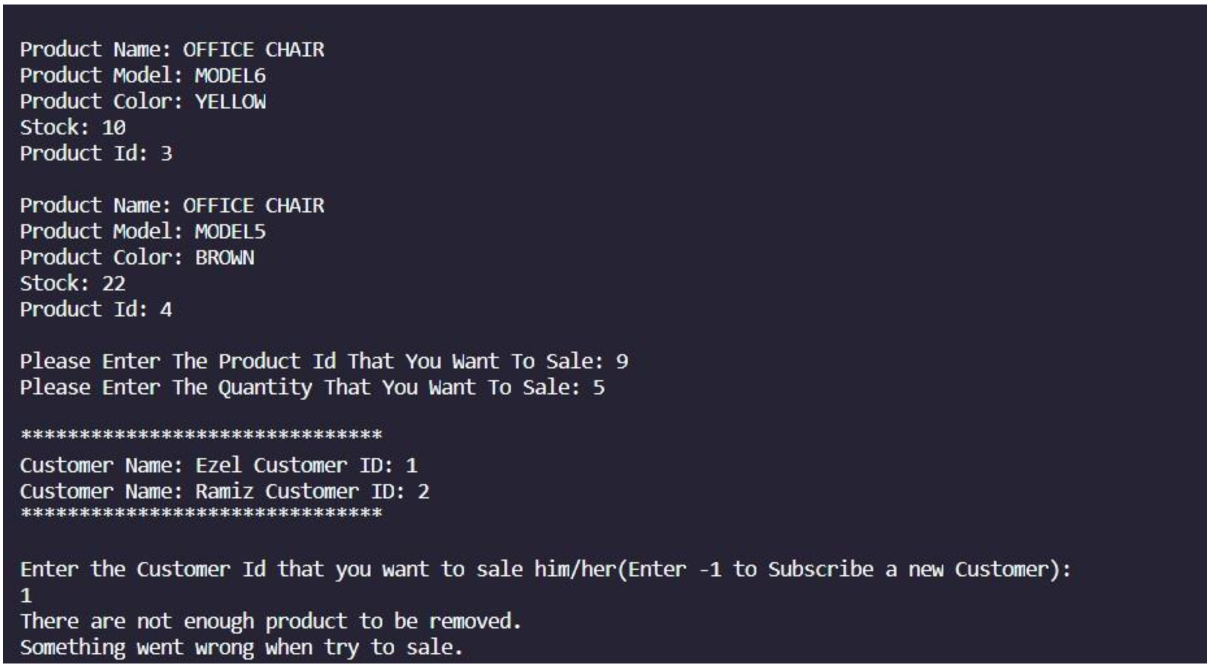
Login validation works for all user types.



```
BRANCH NAME: ISTANBUL
NUMBER OF EMPLOYEES: 1
NUMBER OF PRODUCTS: 4

BRANCH NAME: IZMIR
NUMBER OF EMPLOYEES: 1
NUMBER OF PRODUCTS: 4

BRANCH NAME: ESKISEHIR
NUMBER OF EMPLOYEES: 1
NUMBER OF PRODUCTS: 4

BRANCH NAME: KOCAELI
NUMBER OF EMPLOYEES: 1
NUMBER OF PRODUCTS: 4


ENTER THE NAME OF BRANCH THAT YOU WANT TO REMOVE: DENIZLI
Branch cannot be removed.
1- ADD A BRANCH
2- REMOVE  A BRANCH
3- ADD AN EMPLOYEE
4- REMOVE AN EMPLOYEE
5- QUERY PRODUCT STATE
6- EXIT
SELECTION: 
```

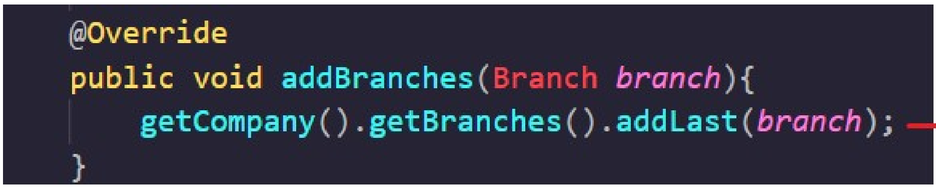
Adding and removing branches, employees and products works fine.

Try to buy product that is not exist handled perfectly.



```
Product Name: OFFICE CHAIR
Product Model: MODEL6
Product Color: YELLOW
Stock: 10
Product Id: 3

Product Name: OFFICE CHAIR
Product Model: MODEL5
Product Color: BROWN
Stock: 22
Product Id: 4

Please Enter The Product Id That You Want To Sale: 9
Please Enter The Quantity That You Want To Sale: 5

*******************************
Customer Name: Ezel Customer ID: 1
Customer Name: Ramiz Customer ID: 2
*******************************

Enter the Customer Id that you want to sale him/her(Enter -1 to Subscribe a new Customer):
1
There are not enough product to be removed.
Something went wrong when try to sale.
```

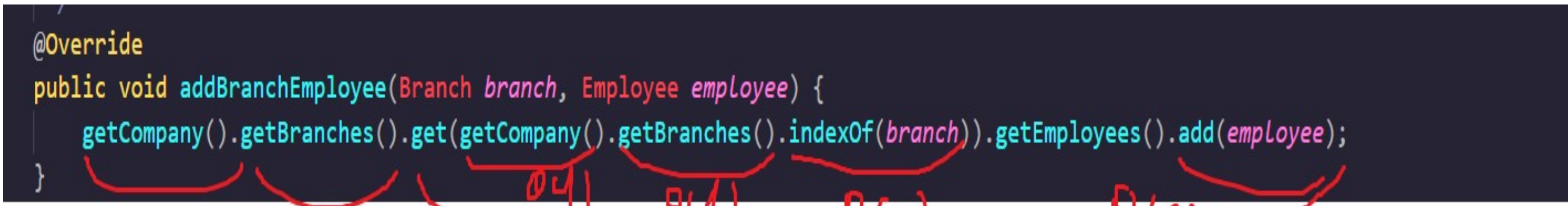
6. TIME COMPLEXITY



```
@Override
public void addBranches(Branch branch){
    getCompany().getBranches().addLast(branch);
}
```

→ O(n)

T(n) = O(n)

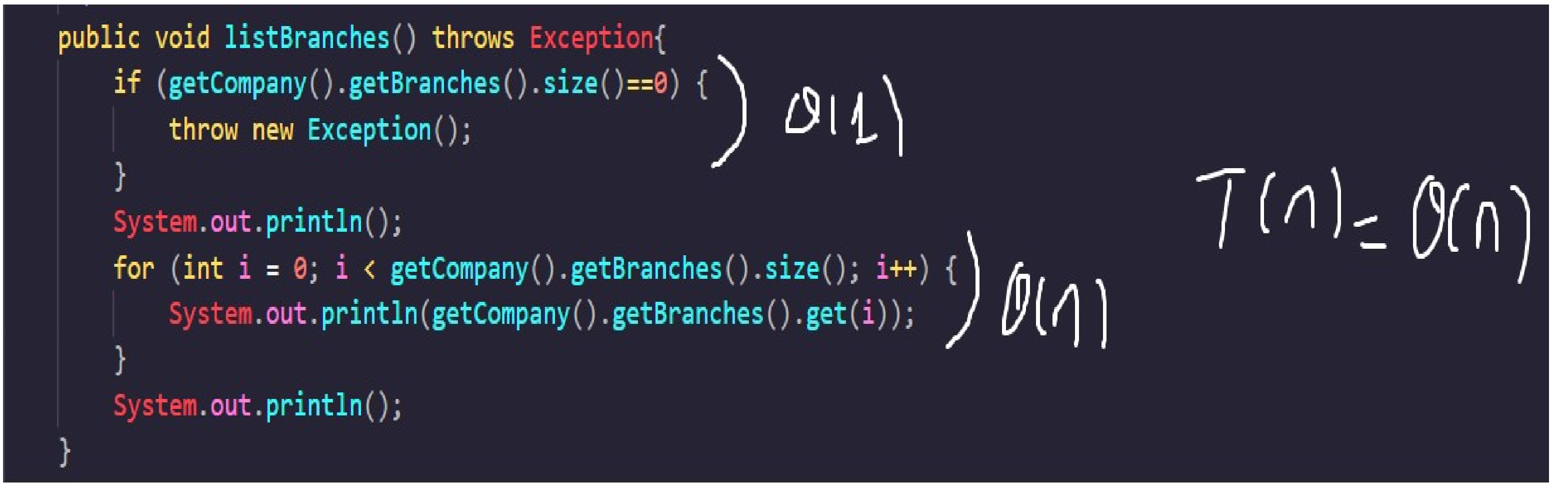


```
@Override
public void addBranchEmployee(Branch branch, Employee employee) {
    getCompany().getBranches().get(getCompany().getBranches().indexOf(branch)).getEmployees().add(employee);
}
```

O(1) O(1) O(1) O(1) O(n) O(n)

O(n)

T(n) = O(n) #

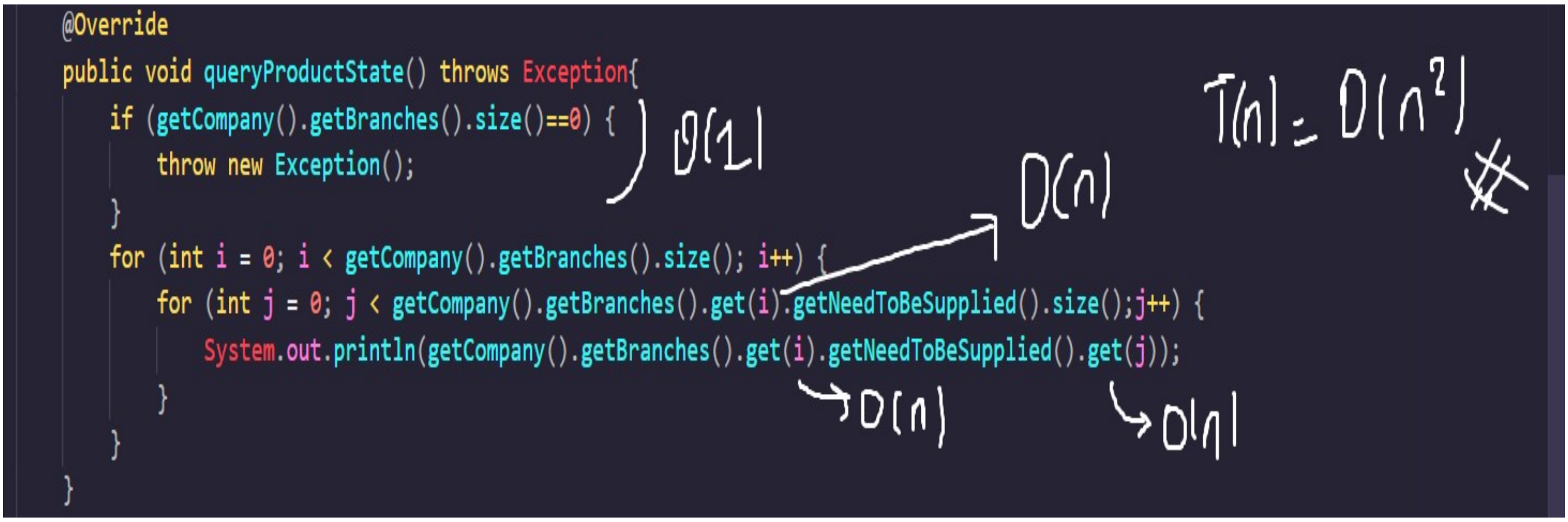


```java
public void listBranches() throws Exception{
    if (getCompany().getBranches().size()==0) {
        throw new Exception();
    }
    System.out.println();
    for (int i = 0; i < getCompany().getBranches().size(); i++) {
        System.out.println(getCompany().getBranches().get(i));
    }
    System.out.println();
}
```

$O(1)$

$O(n)$

$T(n) = O(n)$



```java
@Override
public void queryProductState() throws Exception{
    if (getCompany().getBranches().size()==0) {
        throw new Exception();
    }
    for (int i = 0; i < getCompany().getBranches().size(); i++) {
        for (int j = 0; j < getCompany().getBranches().get(i).getNeedToBeSupplied().size();j++) {
            System.out.println(getCompany().getBranches().get(i).getNeedToBeSupplied().get(j));
        }
    }
}
```

$O(1)$

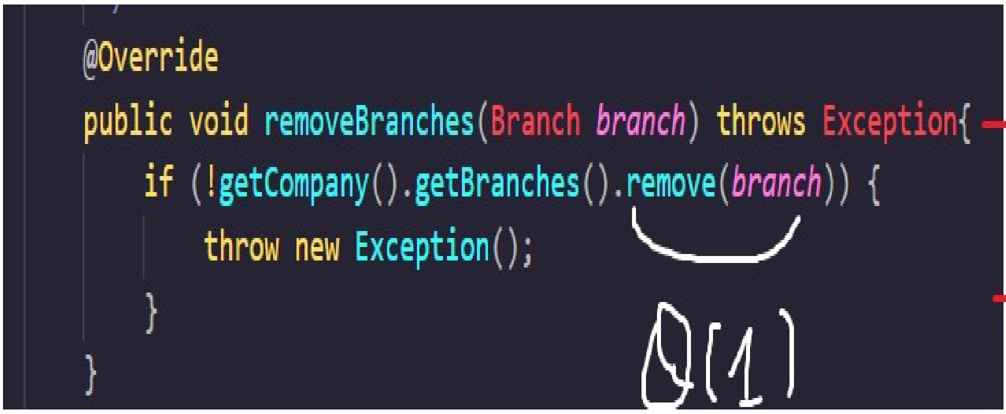
$O(n)$

$O(n)$

$O(n)$

$T(n) = O(n^2)$

#

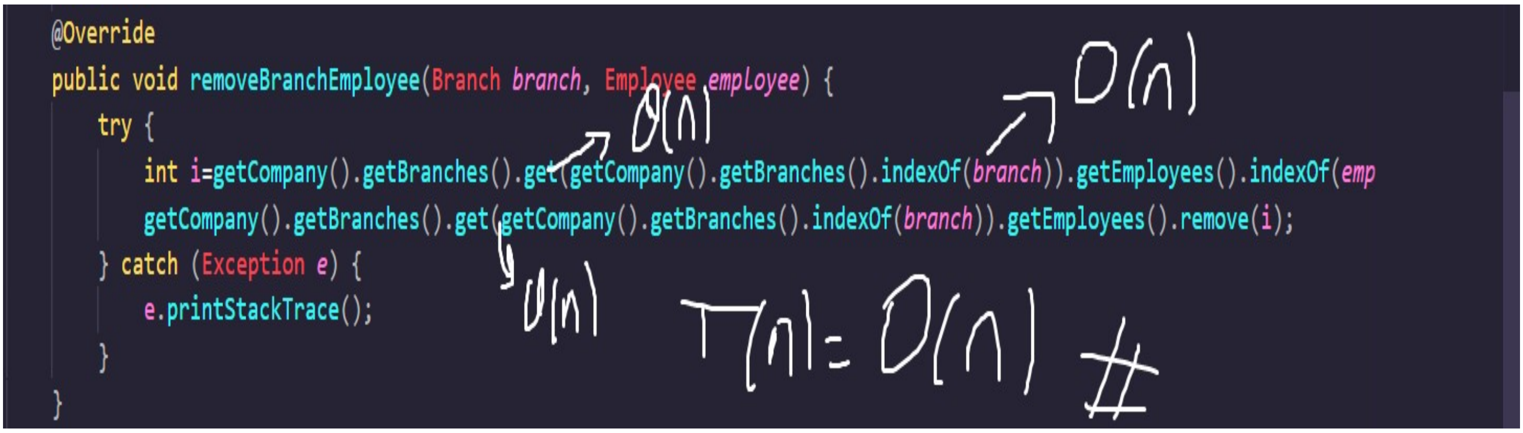


```java
@Override
public void removeBranches(Branch branch) throws Exception{
    if (!getCompany().getBranches().remove(branch)) {
        throw new Exception();
    }
}
```

$O(1)$

$T(n) = O(1)$

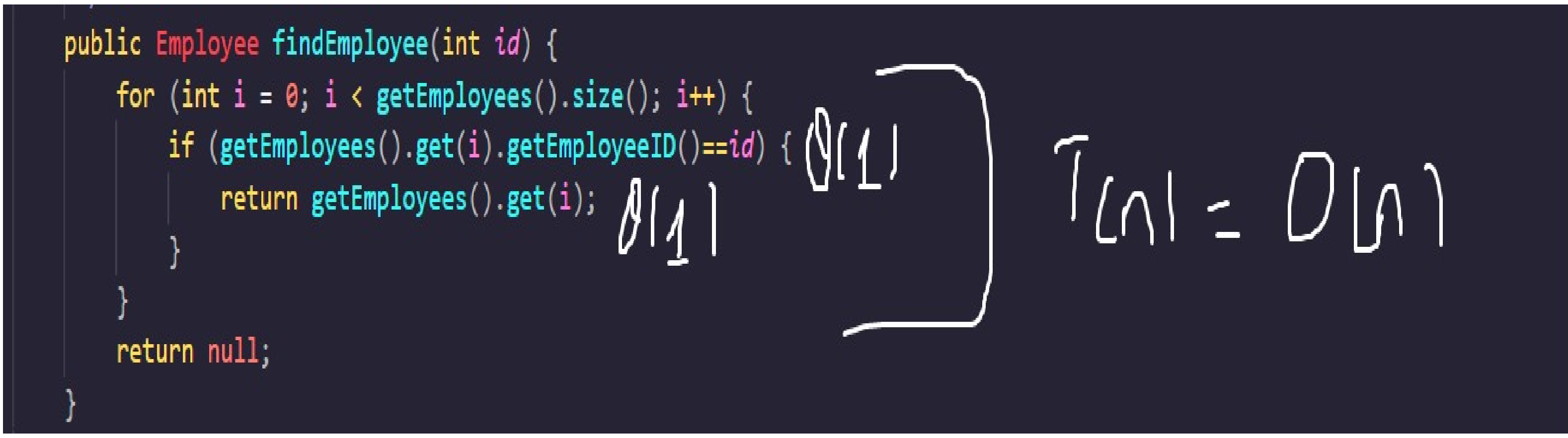
#



```
@Override
public void removeBranchEmployee(Branch branch, Employee employee) {
    try {
        int i=getCompany().getBranches().get(getCompany().getBranches().indexOf(branch)).getEmployees().indexOf(emp
        getCompany().getBranches().get(getCompany().getBranches().indexOf(branch)).getEmployees().remove(i);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

$\Theta(n)$ $O(n)$ $O(n)$

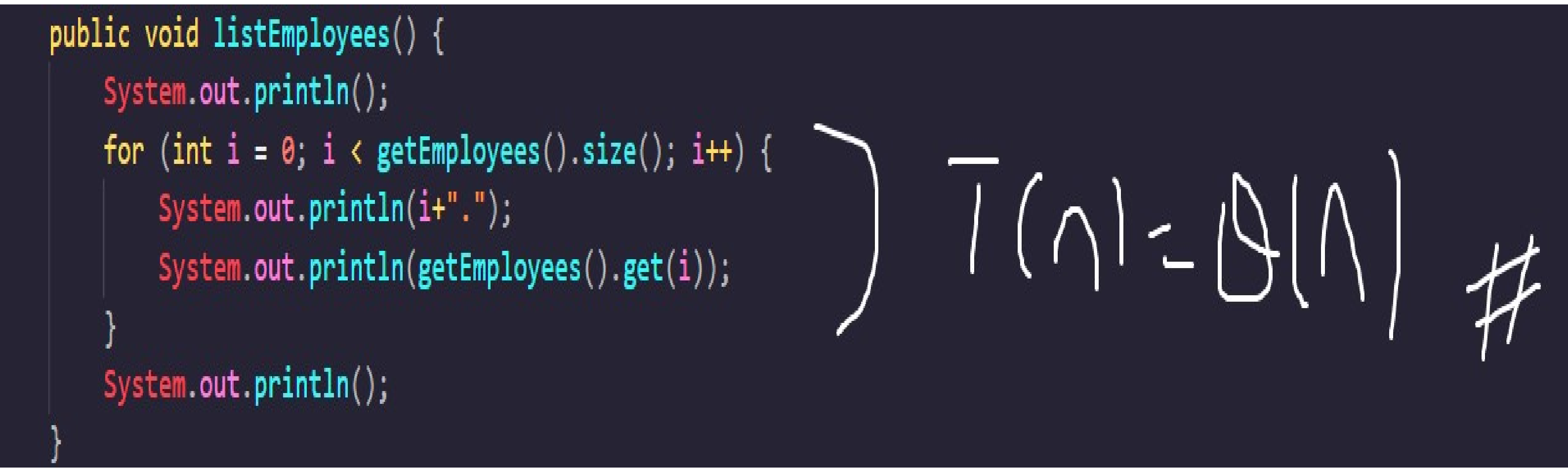
$T(n) = O(n)$ #



```
public Employee findEmployee(int id) {
    for (int i = 0; i < getEmployees().size(); i++) {
        if (getEmployees().get(i).getEmployeeID()==id) {
            return getEmployees().get(i);
        }
    }
    return null;
}
```

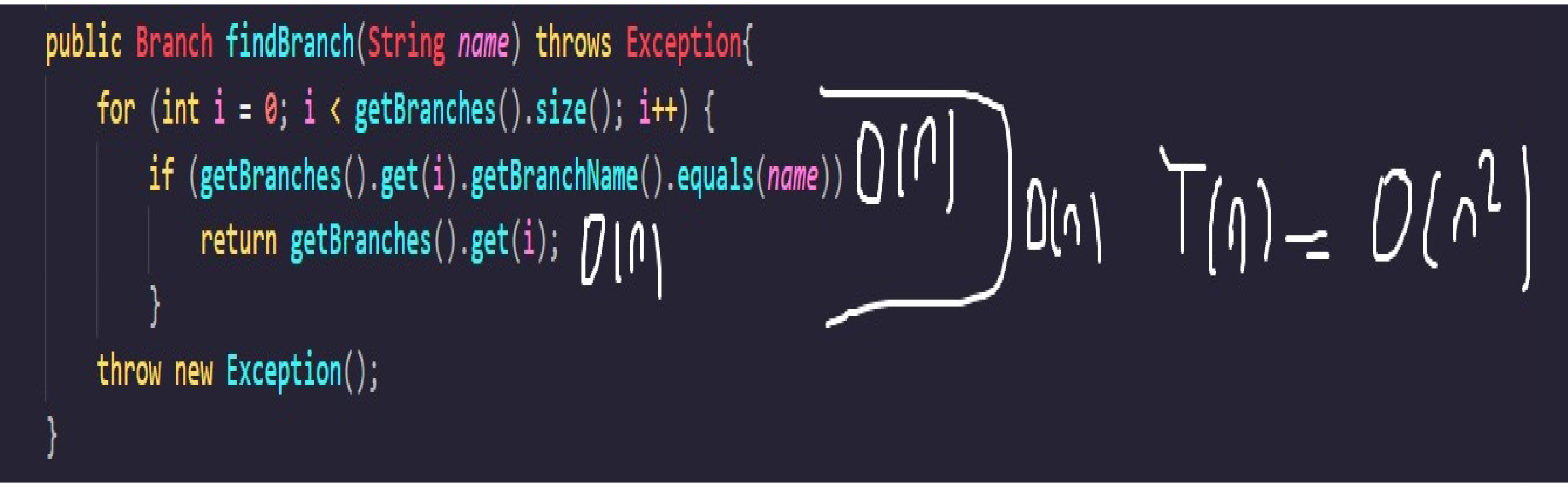
$\Theta(1)$ $\Theta(1)$

$T(n) = O(n)$



```
public void listEmployees() {
    System.out.println();
    for (int i = 0; i < getEmployees().size(); i++) {
        System.out.println(i+".");
        System.out.println(getEmployees().get(i));
    }
    System.out.println();
}
```

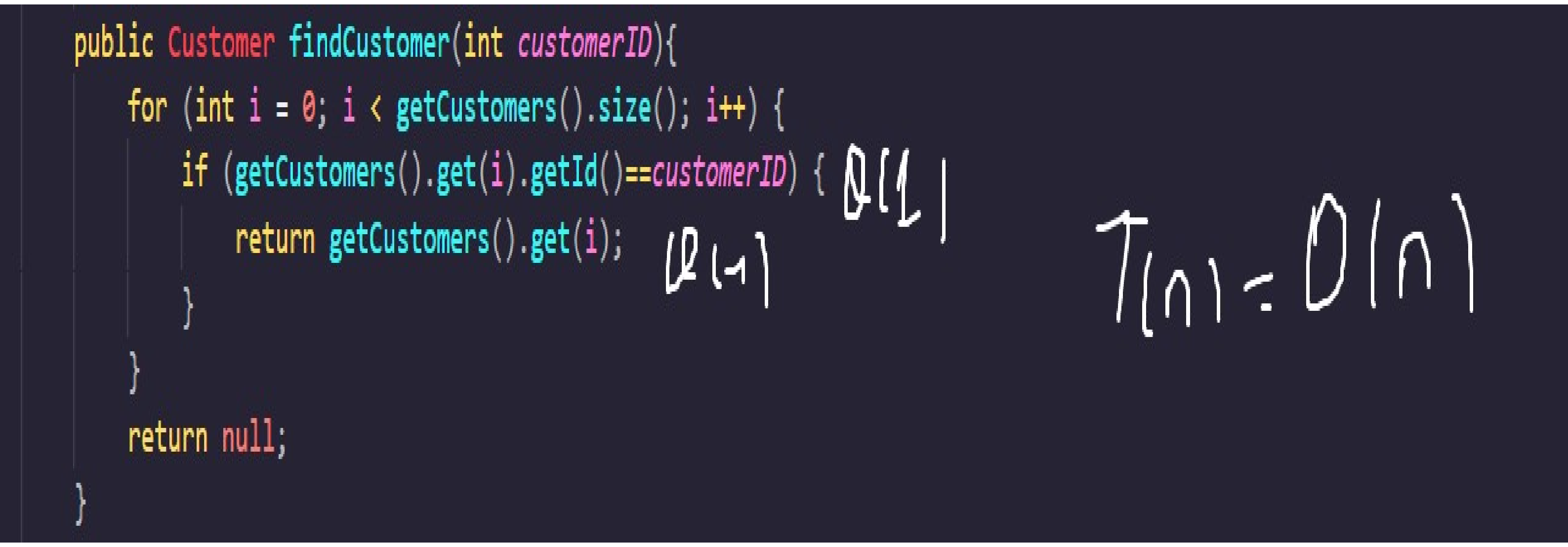
$T(n) = \Theta(n)$ #



```
public Branch findBranch(String name) throws Exception{
    for (int i = 0; i < getBranches().size(); i++) {
        if (getBranches().get(i).getBranchName().equals(name))
            return getBranches().get(i);
        }
    throw new Exception();
}
```

$O(n)$ $O(n)$ $O(n)$

$T(n) = O(n^2)$

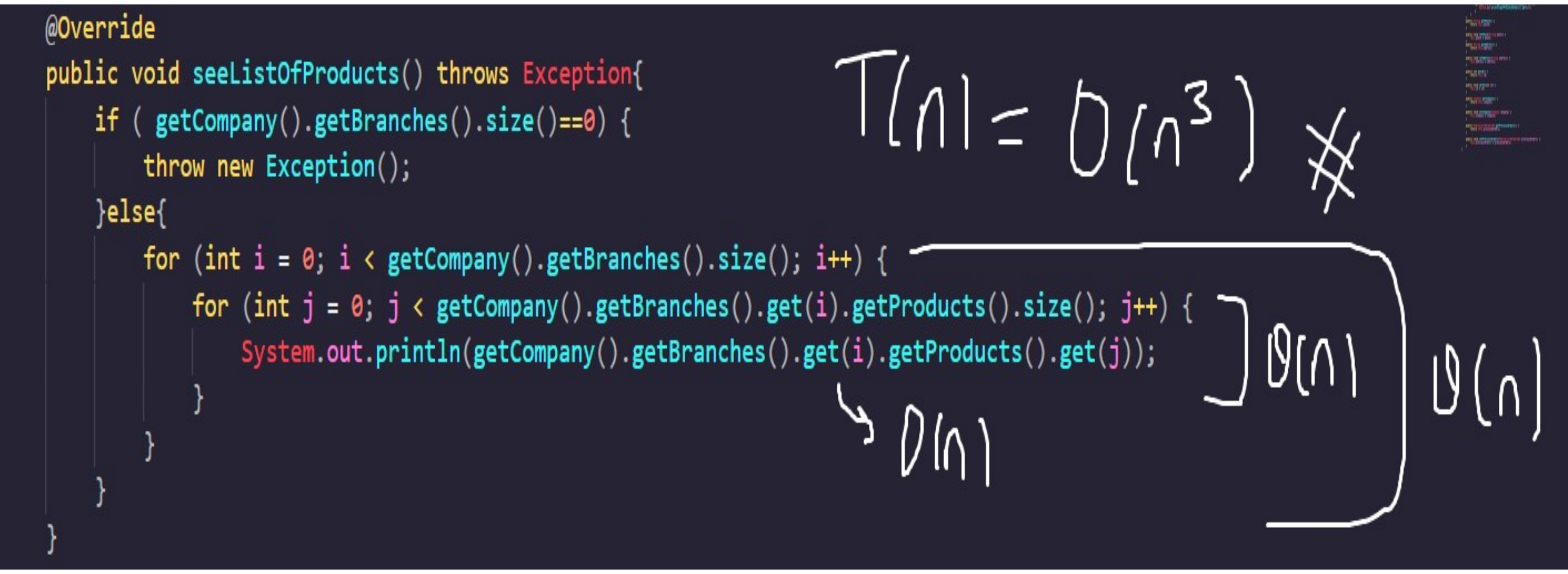


```
public Customer findCustomer(int customerID){
    for (int i = 0; i < getCustomers().size(); i++) {
        if (getCustomers().get(i).getId()==customerID) {
            return getCustomers().get(i);
        }
    }
    return null;
}
```

$O(1)$

$O(1)$

$T(n) = O(n)$



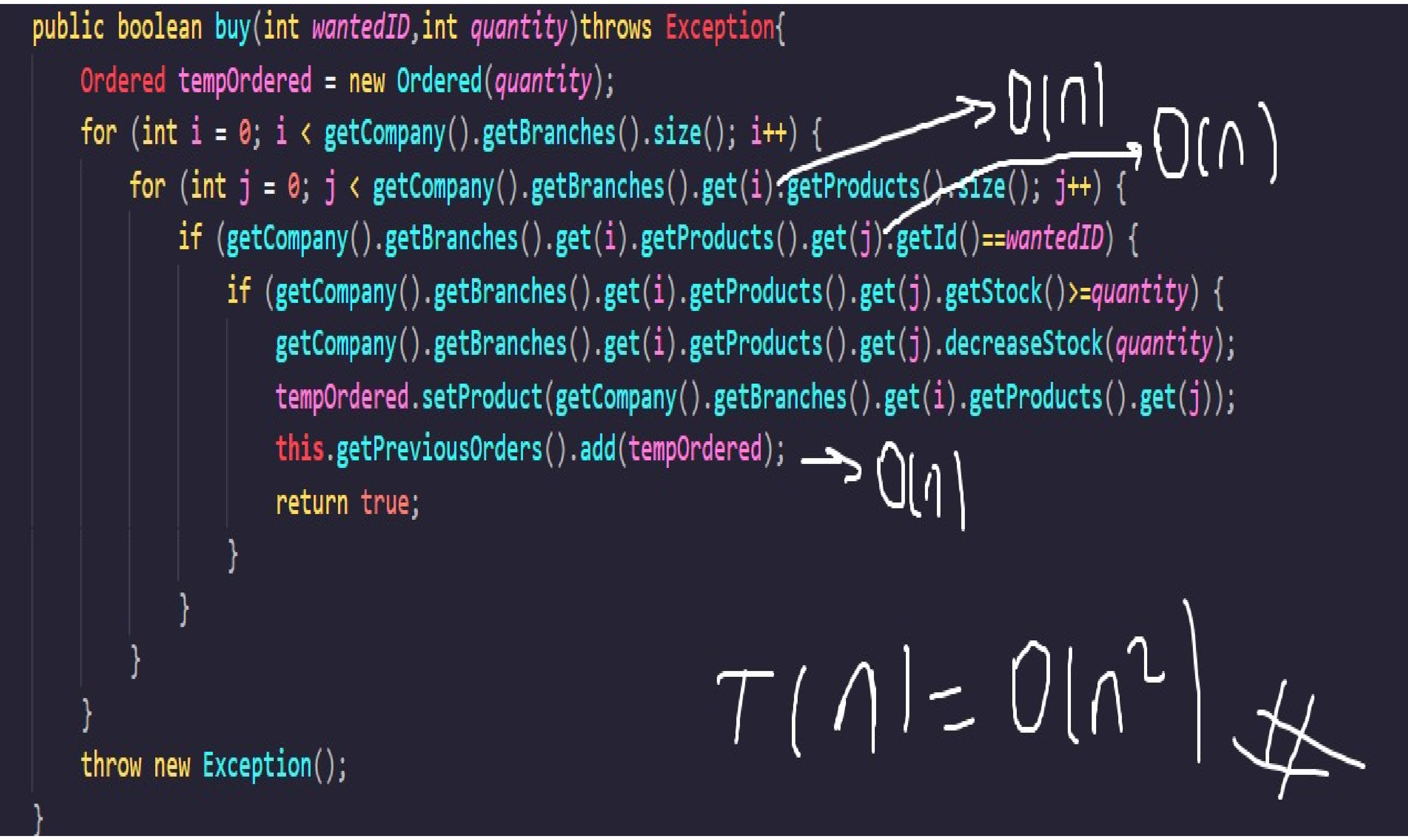
```
@Override
public void seeListOfProducts() throws Exception{
    if ( getCompany().getBranches().size()==0) {
        throw new Exception();
    }else{
        for (int i = 0; i < getCompany().getBranches().size(); i++) {
            for (int j = 0; j < getCompany().getBranches().get(i).getProducts().size(); j++) {
                System.out.println(getCompany().getBranches().get(i).getProducts().get(j));
            }
        }
    }
}
```

$T(n) = O(n^3)$ #

$O(n)$

$O(n)$

$O(n)$



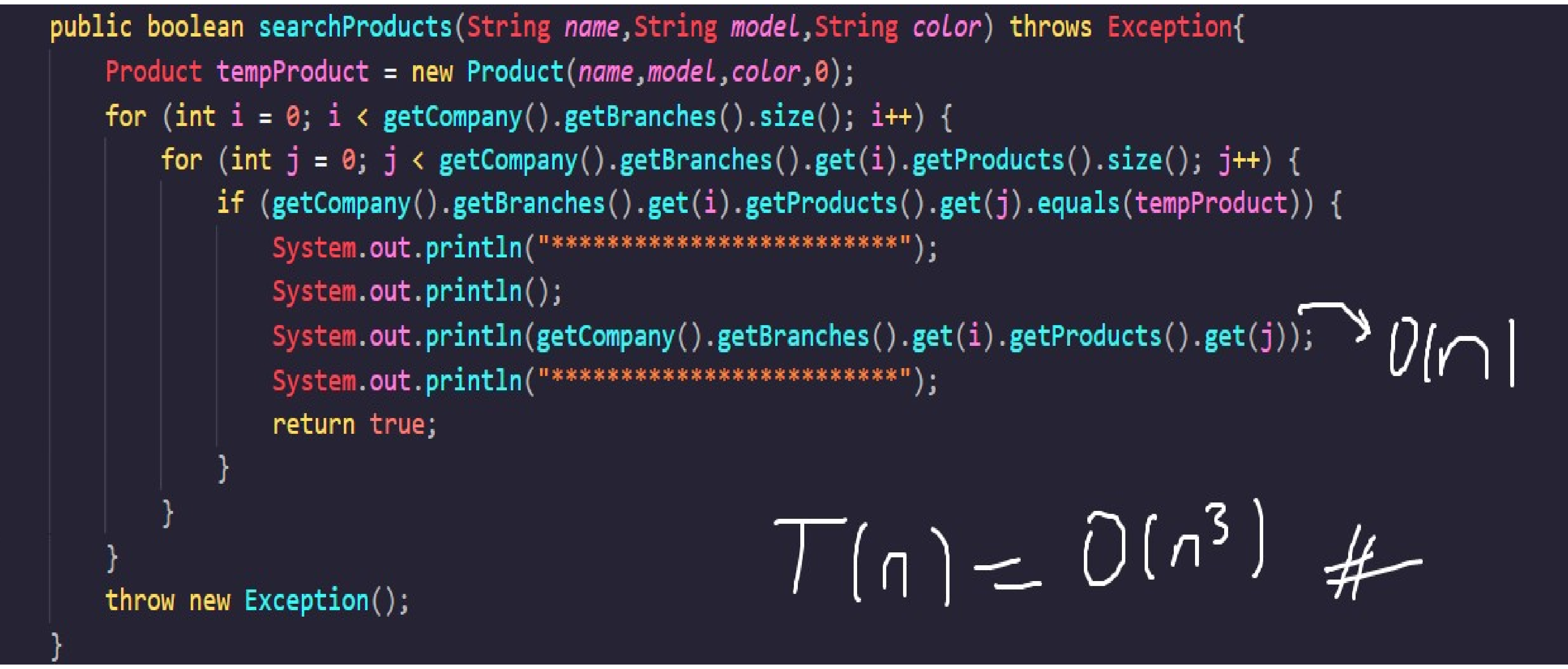
```
public boolean buy(int wantedID,int quantity)throws Exception{
    Ordered tempOrdered = new Ordered(quantity);
    for (int i = 0; i < getCompany().getBranches().size(); i++) {
        for (int j = 0; j < getCompany().getBranches().get(i).getProducts().size(); j++) {
            if (getCompany().getBranches().get(i).getProducts().get(j).getId()==wantedID) {
                if (getCompany().getBranches().get(i).getProducts().get(j).getStock()>=quantity) {
                    getCompany().getBranches().get(i).getProducts().get(j).decreaseStock(quantity);
                    tempOrdered.setProduct(getCompany().getBranches().get(i).getProducts().get(j));
                    this.getPreviousOrders().add(tempOrdered);
                    return true;
                }
            }
        }
    }
    throw new Exception();
}
```

$O(n)$

$O(n)$

$O(1)$

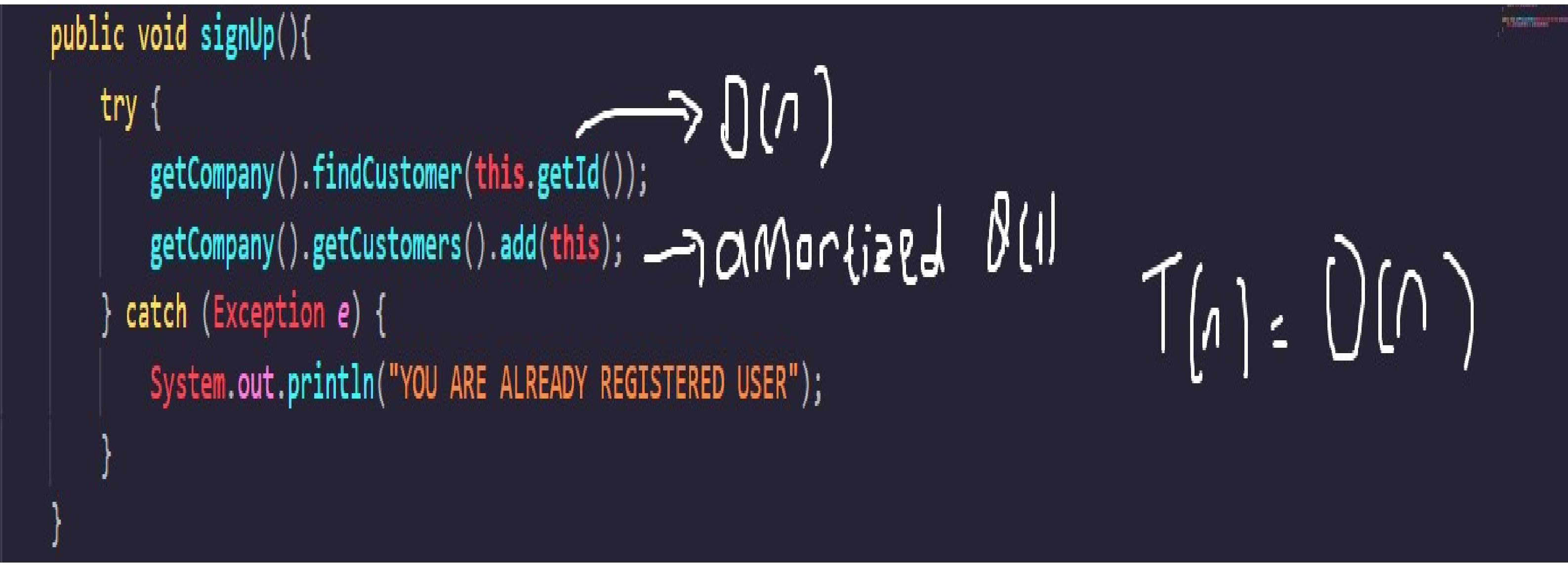
$T(n) = O(n^2)$ #



```java
public boolean searchProducts(String name,String model,String color) throws Exception{
    Product tempProduct = new Product(name,model,color,0);
    for (int i = 0; i < getCompany().getBranches().size(); i++) {
        for (int j = 0; j < getCompany().getBranches().get(i).getProducts().size(); j++) {
            if (getCompany().getBranches().get(i).getProducts().get(j).equals(tempProduct)) {
                System.out.println("*************************");
                System.out.println();
                System.out.println(getCompany().getBranches().get(i).getProducts().get(j));
                System.out.println("*************************");
                return true;
            }
        }
    }
    throw new Exception();
}
```

$\rightarrow O(n)$

$T(n) = O(n^3)$ #

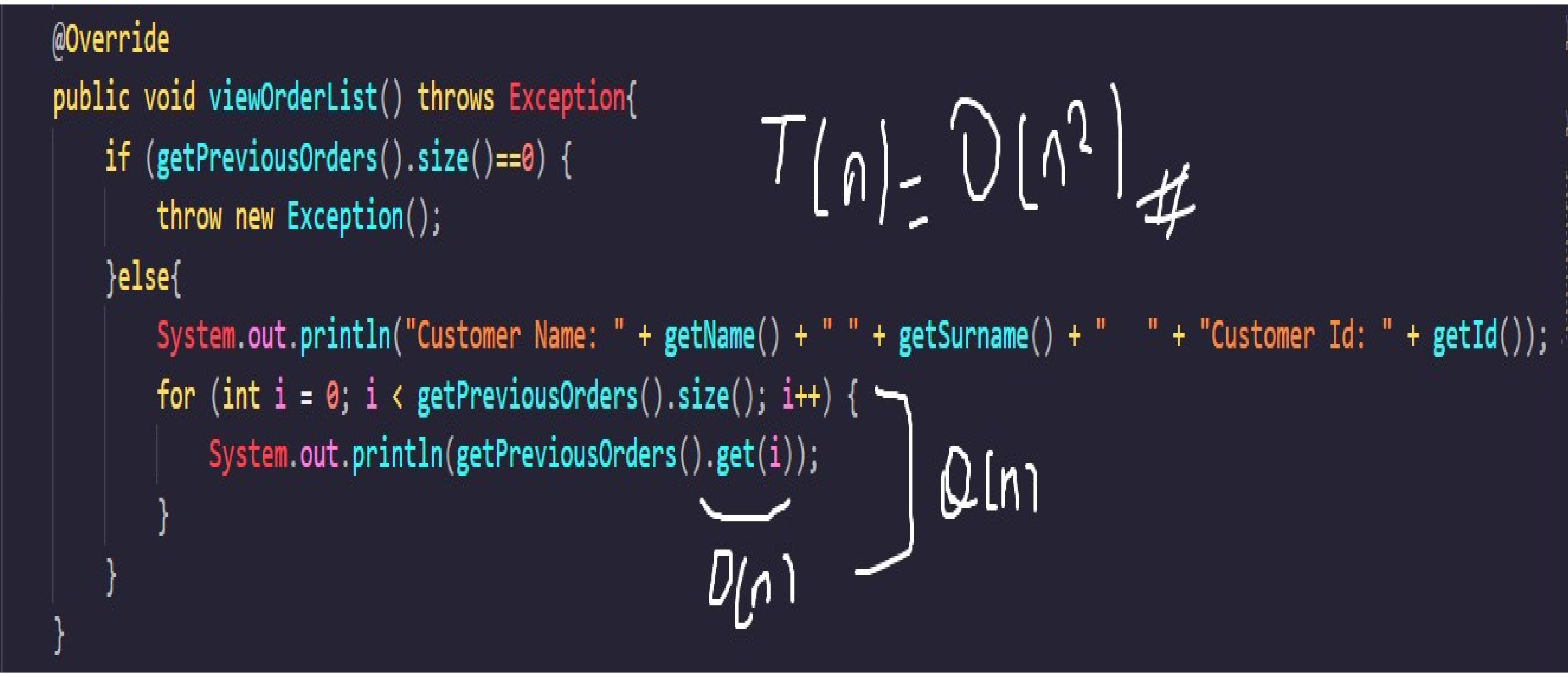


```java
public void signUp(){
    try {
        getCompany().findCustomer(this.getId());
        getCompany().getCustomers().add(this);
    } catch (Exception e) {
        System.out.println("YOU ARE ALREADY REGISTERED USER");
    }
}
```

$\rightarrow O(n)$

$\rightarrow$ amortized $O(1)$

$T(n) = O(n)$

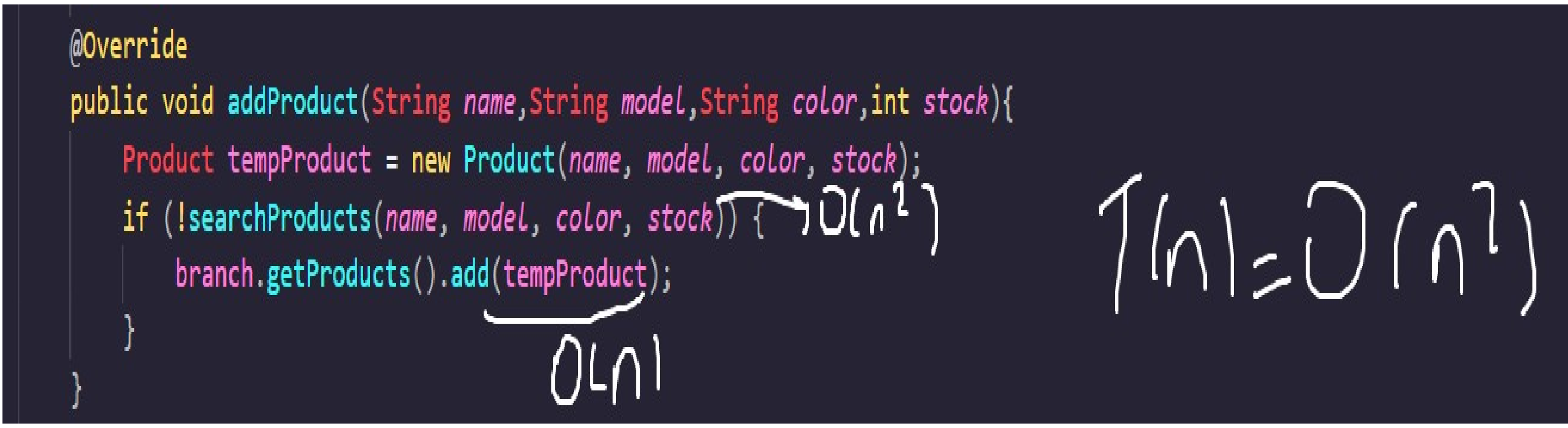


```java
@Override
public void viewOrderList() throws Exception{
    if (getPreviousOrders().size()==0) {
        throw new Exception();
    }else{
        System.out.println("Customer Name: " + getName() + " " + getSurname() + "    " + "Customer Id: " + getId());
        for (int i = 0; i < getPreviousOrders().size(); i++) {
            System.out.println(getPreviousOrders().get(i));
        }
    }
}
```

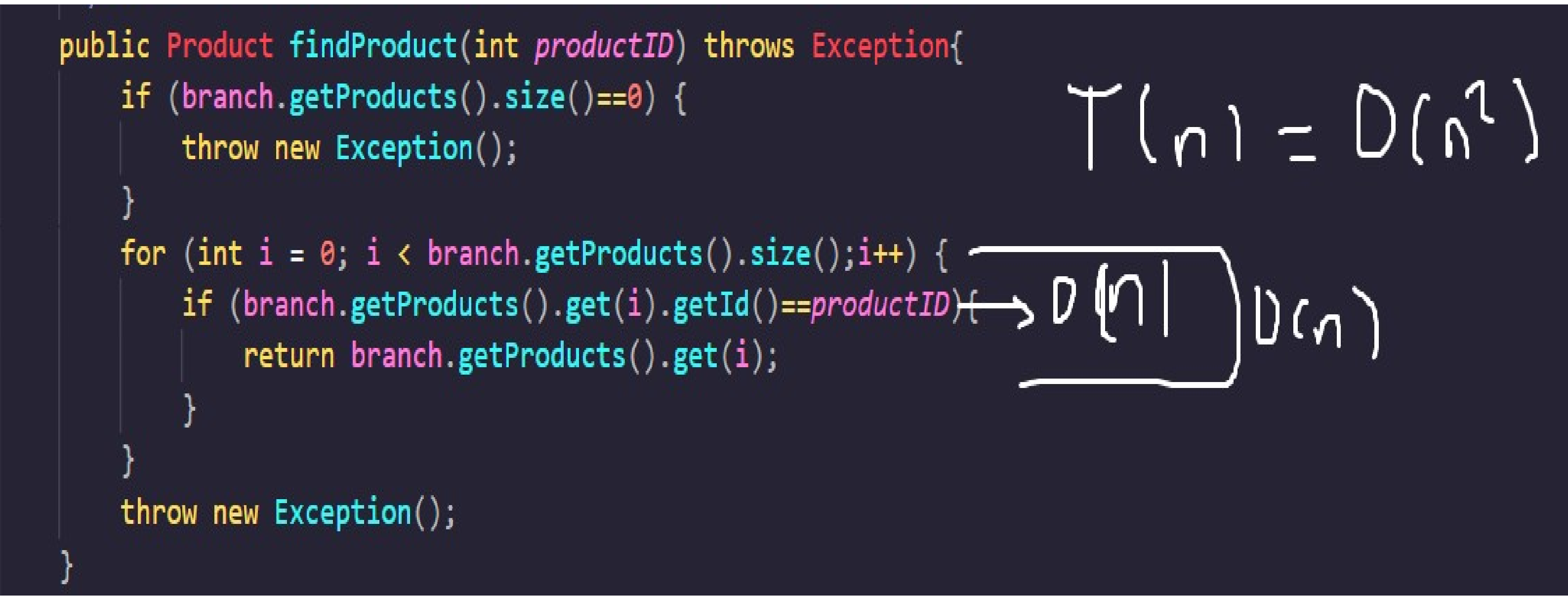
$T(n) = O(n^2)$ #

$O(n)$

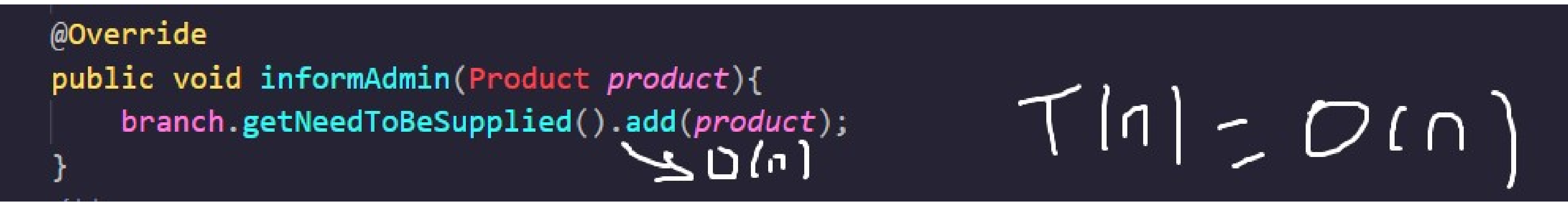
$O(n)$



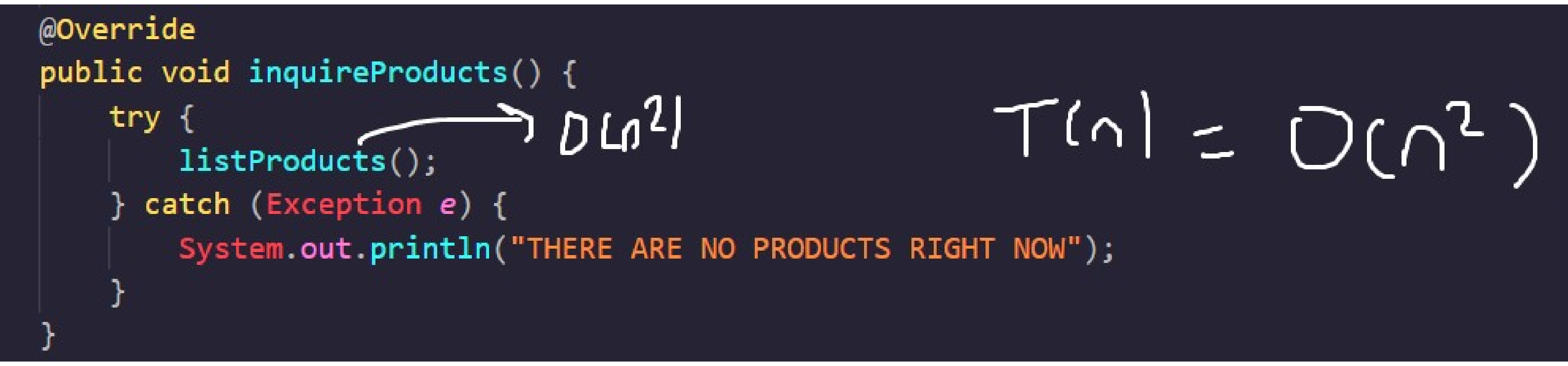
```java
@Override
public void addProduct(String name,String model,String color,int stock){
    Product tempProduct = new Product(name, model, color, stock);
    if (!searchProducts(name, model, color, stock)) {
        branch.getProducts().add(tempProduct);
    }
}
```



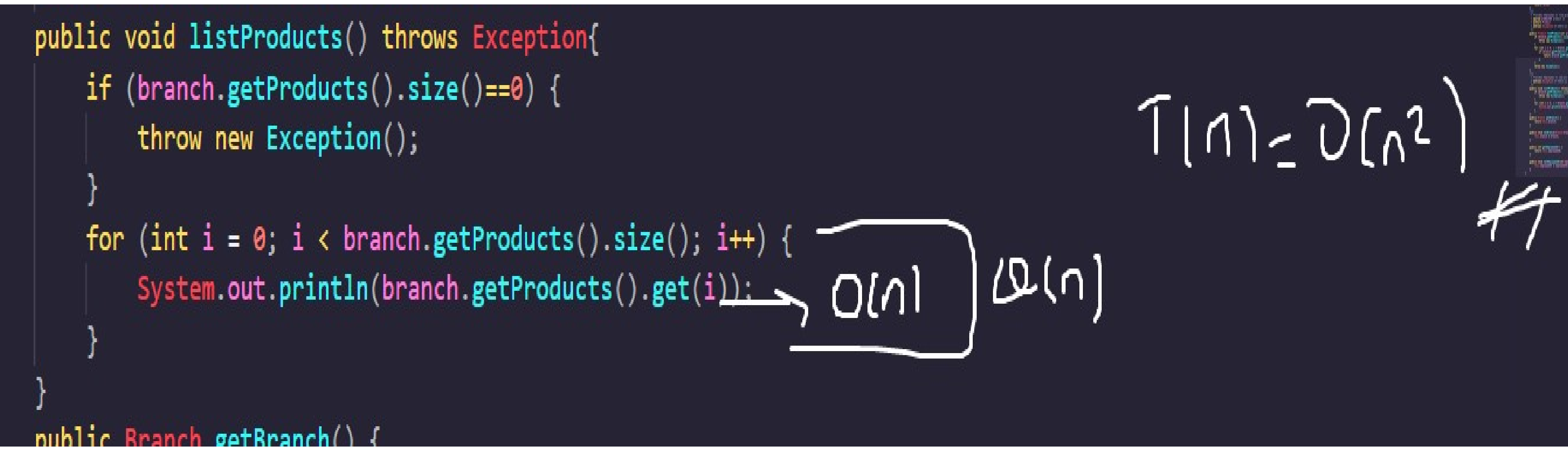
```java
public Product findProduct(int productID) throws Exception{
    if (branch.getProducts().size()==0) {
        throw new Exception();
    }
    for (int i = 0; i < branch.getProducts().size();i++) {
        if (branch.getProducts().get(i).getId()==productID){
            return branch.getProducts().get(i);
        }
    }
    throw new Exception();
}
```



```java
@Override
public void informAdmin(Product product){
    branch.getNeedToBeSupplied().add(product);
}
```



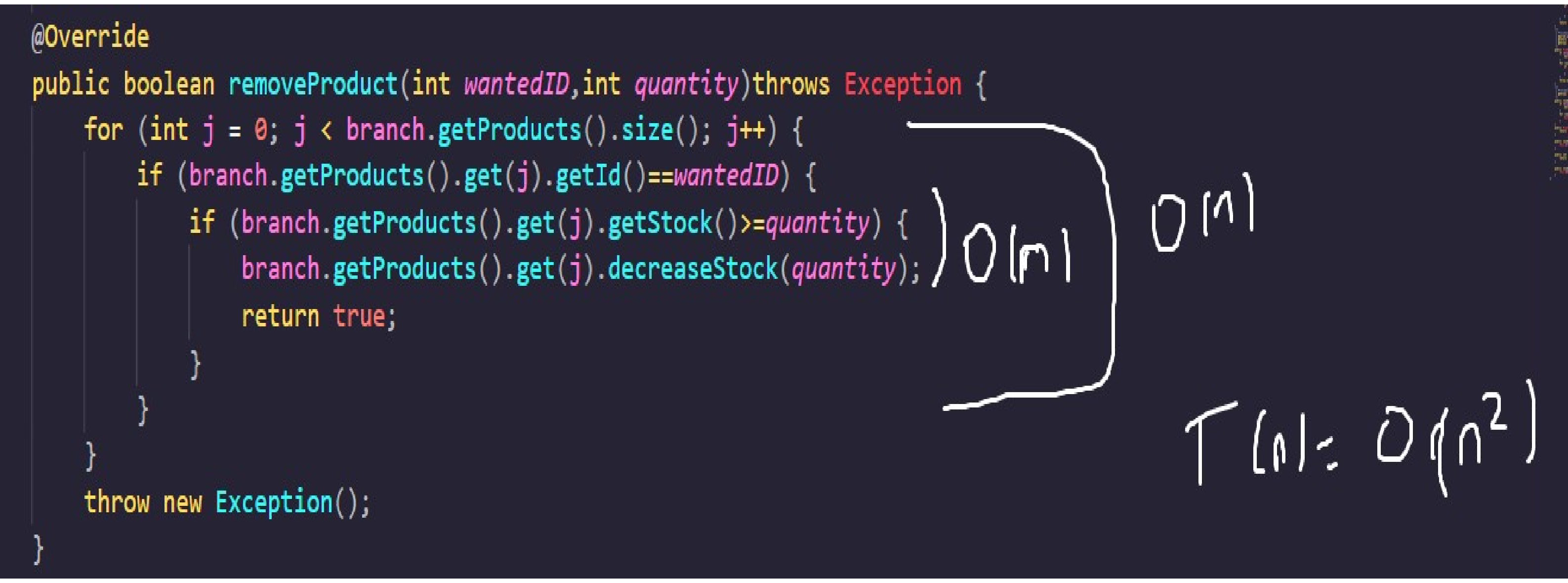
```java
@Override
public void inquireProducts() {
    try {
        listProducts();
    } catch (Exception e) {
        System.out.println("THERE ARE NO PRODUCTS RIGHT NOW");
    }
}
```



```
public void listProducts() throws Exception{
    if (branch.getProducts().size()==0) {
        throw new Exception();
    }
    for (int i = 0; i < branch.getProducts().size(); i++) {
        System.out.println(branch.getProducts().get(i));
    }
}
public Branch getBranch() {
```

$O(n)$ $\Theta(n)$

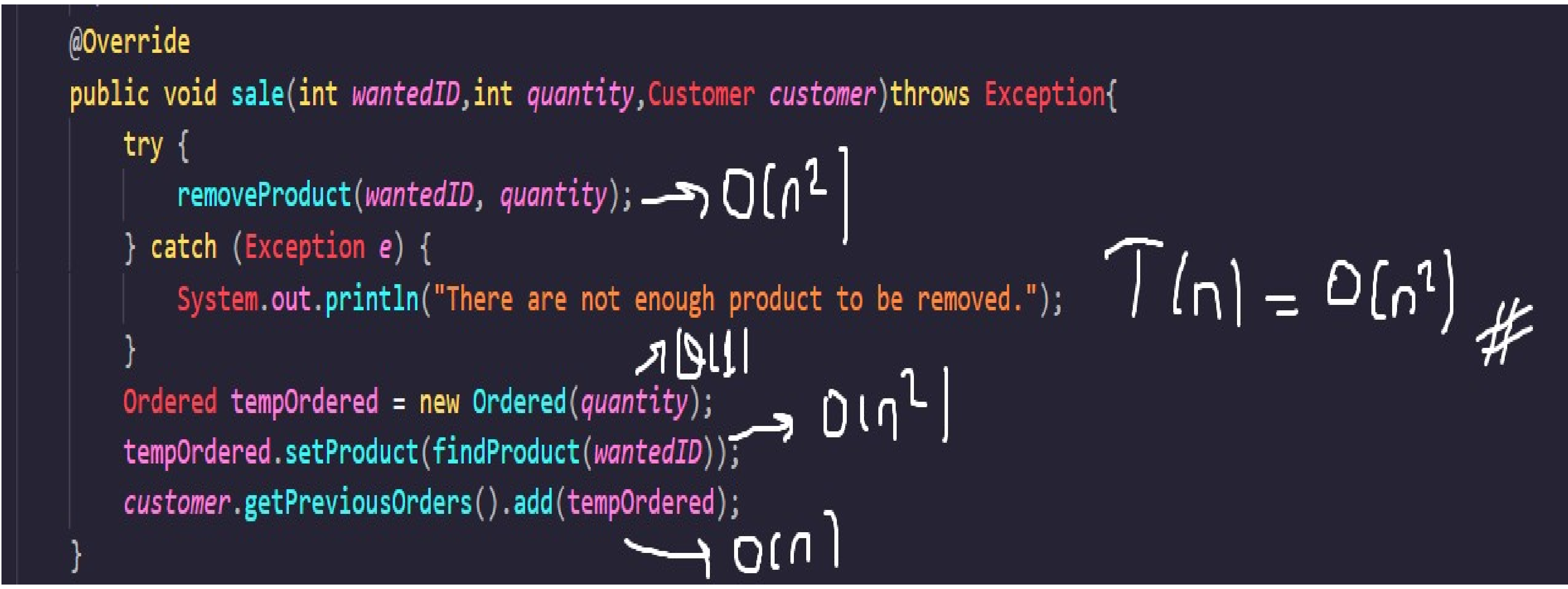
$T(n) = O(n^2)$ #



```
@Override
public boolean removeProduct(int wantedID,int quantity)throws Exception {
    for (int j = 0; j < branch.getProducts().size(); j++) {
        if (branch.getProducts().get(j).getId()==wantedID) {
            if (branch.getProducts().get(j).getStock()>=quantity) {
                branch.getProducts().get(j).decreaseStock(quantity);
                return true;
            }
        }
    }
    throw new Exception();
}
```

$O(m)$ $O(n)$

$T(n) = O(n^2)$



```
@Override
public void sale(int wantedID,int quantity,Customer customer)throws Exception{
    try {
        removeProduct(wantedID, quantity);
    } catch (Exception e) {
        System.out.println("There are not enough product to be removed.");
    }
    Ordered tempOrdered = new Ordered(quantity);
    tempOrdered.setProduct(findProduct(wantedID));
    customer.getPreviousOrders().add(tempOrdered);
}
```

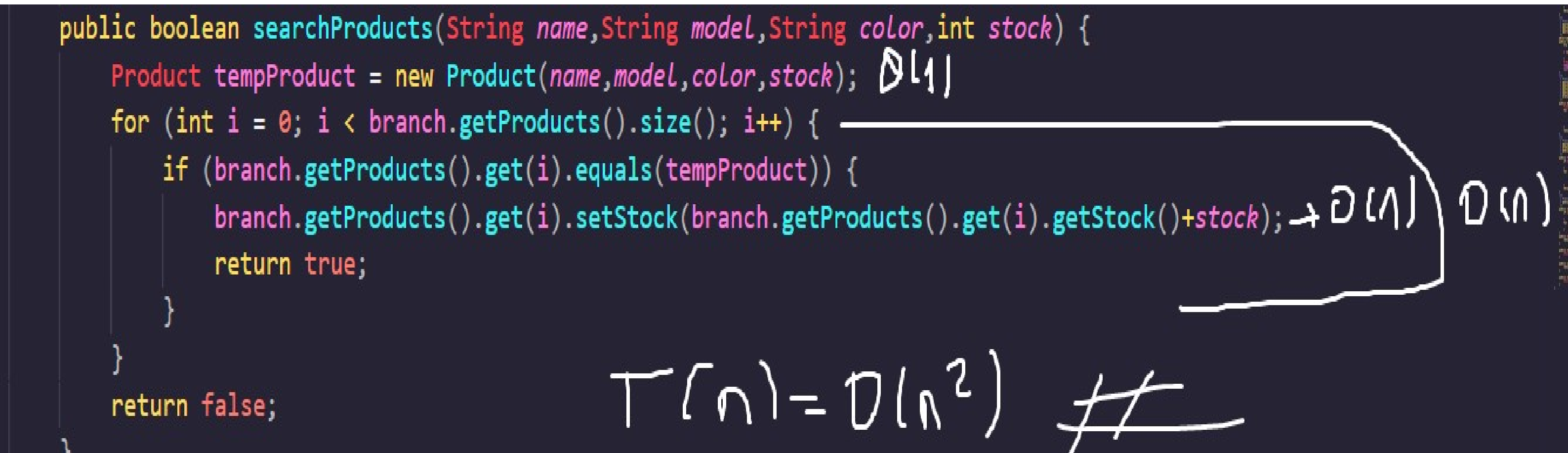
$O(n^2)$

$O(1)$

$O(n^2)$

$O(n)$

$T(n) = O(n^2)$ #



```
public boolean searchProducts(String name,String model,String color,int stock) {
    Product tempProduct = new Product(name,model,color,stock);
    for (int i = 0; i < branch.getProducts().size(); i++) {
        if (branch.getProducts().get(i).equals(tempProduct)) {
            branch.getProducts().get(i).setStock(branch.getProducts().get(i).getStock()+stock);
            return true;
        }
    }
    return false;
}
```

$O(1)$

$O(n)$ $O(n)$

$T(n) = O(n^2)$ #