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


**Mustafa Karakaş**
**1801042627**


- **SYSTEM REQUIREMENTS**

**Constructers:**

First of all, the company constructer should be called.

```
Company c = new Company();
```

This constructer creates 4 empty branches as default and company holds them.

To be able to add employee and new branches we need a Administrator.

```
public Administrators(String _name,String _surname,Company comp)
```

To be able to create a new employee :

```
public BranchEmployee(String _name,String _surname)
```

To be able to create a new branch:

```
public Branch(Company company)
```

To be able to create a new customer:

```
public Customer(String _name,String _surname,String _e_mail,String pw)
```

## Administrator's methods:

To be able to add Branch Employee we should call Administator's method:

```
/**
 * adds employee to branch which has given index
 * @param emp branchEmployee which is wanted to be added to the branch
 * @param branchIndex index of branch at company
 */
public void addEmployee(BranchEmployee emp,int branchIndex)throws IndexOutOfBoundsException
```

As we said, as default there is 4 branches at company. So if administrator have not added any branch yet, then we can not add employee to the 5. branch (index can not be equal to 4).

To be able to remove Branch Employee from the branch ,we should call administrator's method:

```
public boolean removeEmployee(BranchEmployee emp,int branchIndex)throws IndexOutOfBoundsException
```

To be able to add new branch to the company we should call the Administrator's method:

```
public void addBranch()
```

To remove the branch:

```
public boolean removeBranch(int branchIndex)throws IndexOutOfBoundsException
```

When we remove a branch , it's employees are unemployed. So if we call their method then it throws BranchEmployeeDoesNotHaveAuthority exception.

They can also query whether there are any products that need to be supplied.

```
public void askForProductNeed(Product p,int branchIndex)throws IndexOutOfBoundsException
```

If the given product does not exist in the branch which has the given index, administrator supplies it.

Administrator can show branch employees and product of the branch which has the given index:

```
public void showBranchsProducts(int branchIndex)throws IndexOutOfBoundsException
```

```
public void showBranchEmployees(int branchIndex)throws IndexOutOfBoundsException
```

Administrator can also add/remove product:

```
public void addProduct(Product p,int branchIndex) throws IndexOutOfBoundsException
```

```
public boolean removeProduct(Product p,int branchIndex)throws IndexOutOfBoundsException
```

Administrator has a method that adds all the products available to the branch that has the given index:

```
public void addAllProducts(int BranchIndex)throws IndexOutOfBoundsException
```

## Branch Employee's Methods:

Branch employee can add remove products to the branch which he/she works.

```
public void addProduct(Product p)throws BranchEmployeeDoesNotHaveAuthority
```

```
public boolean removeProduct(Product p) throws BranchEmployeeDoesNotHaveAuthority
```

Branch employees can inquire about the products in stock :

```
public void showBranchProducts()
```

Inform the manager that the product should be purchased when any product is less than the requested amount :

```
public void informManager(Company c,Product p,int branchIndex)
```

Branch Employee can make a sale:

```
public void makeSale(Customer c,Product p)throws BranchEmployeeDoesNotHaveAuthority
```

Branch Employee can access the information of the previous orders of a customer by using the customer number and add new order to this section

```
public vector<Product> getPreviousOrders(Company company,int customerNumber)
```

customerNumber can be accessed via getCustomerNumber method of customer class:

```
public int getCustomerNumber()
```

To be able to reach branch of employee , we should hold it. And only Administrator is allowed to do it. So I added inner class named Authority to administrator class and this method can be invoked by only Administrator.

```
public void setBranch(Administrators.Authority I_am_an_Administrator ,Branch branch)
```

If someone tries to send null to this function it throws an error.

Checks if the given product is available in the branch

```
public boolean isProductAvailable(Product p)throws BranchEmployeeDoesNotHaveAuthority
```

Sells product. If the given product is not available , branch employee informs admin. And admin supplies to the product which branch employee needed. So customer can buy it.

```
public void makeSale(Customer c,Product p)throws BranchEmployeeDoesNotHaveAuthority{
```

```
public void informManager(Company c,Product p,int branchIndex)
```

Branch employee can have the customer's previous orders.

```
public vector<Product> getPreviousOrders(Company company,int customerNumber)
```

## Customer

```
public void seeTheListOfProducts()
```

Shows the product which registered company has

```
public int inWhichStoreIsProduct(Product p)
```

Returns the index of the branch which has the given product in the company

```
public boolean searchForProduct(Product p,Company c)
```

Searchs for the given product in the given company, if it is available returns true, else returns false.

```
public void addOrder(BranchEmployee.Authority authority,Product p)
```

This method is called by only branch employee. It adds the product to the previous orders of the customer.

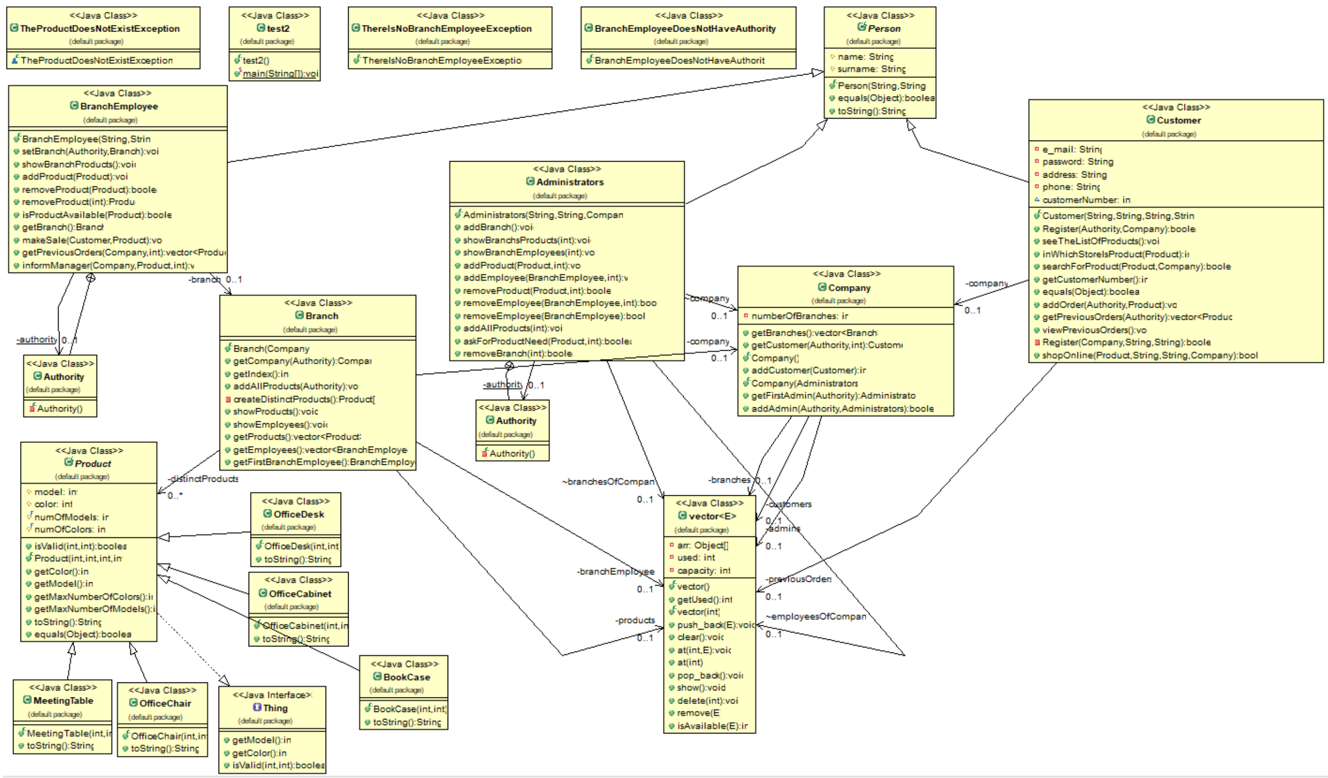
```
public boolean shopOnline(Product p,String Address,String phone,Company comp)
                                   throws ThereIsNoBranchEmployeeException{
```

Buys product remotely. If the company has no employees throws error. If the customer has registration to the another company returns false. If the customer do not register to any company or he/she registered to the given company then ,customer can buy the product and the function returns true. If customer have not registered yet, registers it to the given company.If the given product is not available then admin supplies it , so that customer can buy it.

```
public boolean Register(BranchEmployee.Authority authority,Company comp){
```

When customer buys something for the first time BranchEmpoyee completes his/her registration.

- **USE CASE AND CLASS DIAGRAMS**




Company is the class holds all of the sistem. And it consists of branches. Branches have products and employees. Products derive from Product class. And Product class implements Thing interface. Company also holds Administrators. Administrators ,BranchEmployees and Customers are derived from Person class.

- **PROBLEM SOLUTION APPROACH**

To be able to use arrays easily , I created a container vector class. The biggest class in the project is Company.It holds branches and administrators.Branches holds products and employees. Administrator have an access to all branches. So that, Administrators can add/remove branches , employees and products.Employees can make a sell and register customers. Customers can buy products online.If they enter their addresses and phone numbers. Branch employees can also add and remove product.

Administrators are very capable players. They can change almost everything in the system. And others do not have all these powers. So, I had to create something which tells other classes that, the caller of your method is an Administrator. So , he/she can change some data fields of them and only administrators can do that. The solution was creating a inner class which is private and final. It's name is Authority. When other classes' methods see that their parameter is a authority class of Administrator ,they let it change their data fields.Of course they should check that if the parameter is null or not. The same idea worked for the branch employee class too. It has also authority class and it makes branch employee class more powerful but not as much as administrator of course. What was I trying to build was a 'friend class' but special to the some methods. So , I could make the methods ,which I want only administrators or branch employees can call, public and all classes could see them but only administrators or branch employees could call them.

- **TEST CASES**

```
Company c = new Company();
Administrators admin = new Administrators("Mustafa","Karakas",c);
BranchEmployee ElonMusk = new BranchEmployee("Elon","Musk");
BranchEmployee JeffBezos = new BranchEmployee("Jeff","Bezos");
BranchEmployee Mark = new BranchEmployee("Mark","Zuckerberg");
```

Company and administrator and 3 employee are created.

```
admin.addBranch();
admin.addEmployee(ElonMusk,4);
admin.addEmployee(JeffBezos,4);
admin.addEmployee(Mark,4);
ElonMusk.addProduct(new OfficeChair(1,1));
ElonMusk.addProduct(new OfficeChair(2,1));
ElonMusk.addProduct(new OfficeChair(3,1));
```

admin added a new branch now Company has 5 branches.

admin added umemployed employeers to the branch which has 4 as a index.

ElonMusk added 3 new products to the branch which he is working in.

```
Customer mike = new Customer("mike","tyson",null,null);

ElonMusk.makeSale(mike, new OfficeChair(1,1));
ElonMusk.makeSale(mike, new OfficeChair(2,1));
```

New customer has come. Elon selled him 2 products which are already in the branch.

Because of that was mike's first order, Mike has been registered to the company by Elon and he has a customer number from now on. Because of that he is the first customer in the company it's number is 0 , it will increment by 1 for every customer who registered to the company :

```
This is mike tyson's first purchasing and he/she is registered to the company
mike tyson's customer number :0
```

```
// office chair 4,1 is not available employee will inform admin
//and the admin will supply the product
ElonMusk.makeSale(mike, new OfficeChair(4,1));
```

Elon tried to sell something which is not available in the branch, so he should inform the admin so that admin can supply the product and Elon can make a sale.

```
Elon Musk informed manager to add Office Chair Model :4 Color Type :1
Office Chair Model :4 Color Type :1 added to 5. branch
```

So far, mike bought 3 different products:

```
System.out.println("The products "+mike+" has bought");
ElonMusk.getPreviousOrders(c,mike.getCustomerNumber()).show();
```

```
The products mike tyson has bought
Office Chair Model :1 Color Type :1
Office Chair Model :2 Color Type :1
Office Chair Model :4 Color Type :1
```

mike can see all the products available in the company , before that I want to add more products:

```java
for(int i=0;i<5;i++){
    admin.addProduct(new MeetingTable(i+1,1),i);
    admin.addProduct(new OfficeDesk(i+1,1),i);

}
```

```java
System.out.println("List of products:");
mike.seeTheListOfProducts();
```

```
List of products:
List of products at the 1 . branch
Meeting Table Model :1 Color Type :1
Office Desk Model :1 Color Type :1

List of products at the 2 . branch
Meeting Table Model :2 Color Type :1
Office Desk Model :2 Color Type :1

List of products at the 3 . branch
Meeting Table Model :3 Color Type :1
Office Desk Model :3 Color Type :1

List of products at the 4 . branch
Meeting Table Model :4 Color Type :1
Office Desk Model :4 Color Type :1

List of products at the 5 . branch
Office Chair Model :3 Color Type :1
Meeting Table Model :5 Color Type :1
Office Desk Model :5 Color Type :1
```

We can check if the given product is available at the branch:

```java
System.out.println("check product availability\n");
if(JeffBezos.isProductAvailable(p=new OfficeChair(1,1))){
    System.out.println("We have the product you want : "+p);
}else{
    System.out.println("We do not have the product you want : "+p);
}
if(JeffBezos.isProductAvailable(p=new OfficeChair(3,1))){
    System.out.println("We have the product you want : "+p);
}else{
    System.out.println("We do not have the product you want : "+p);
}
System.out.println("\ncheck product availability end\n");
```

```
check product availability

We do not have the product you want : Office Chair Model :1 Color Type :1
We have the product you want : Office Chair Model :3 Color Type :1

check product availability end
```

Administrators can remove employees:

```java
System.out.println("\nRemove employee\n");
if(admin.removeEmployee(ElonMusk,4)){
    System.out.println(ElonMusk+" is fired by "+admin);
}
else{
    System.out.println("Given employee does not exist in the given branch");
}
```

```
Remove employee

Elon Musk is fired by Mustafa Karakas
```

(I was struggling to find outputs. I copied output to the editor so that I can search for words . Thats why output's background has changed. )

So Elon Musk do not have a job anymore. If he tries to make a sale again (before admin hire him again) the function will throw an exception.

```java
        System.out.println("This will throw an exception because "+ElonMusk+" has been fired\n");
        ElonMusk.makeSale(mike, new OfficeChair(5,1));
    }
    catch(BranchEmployeeDoesNotHaveAuthority e){
        System.out.println(e);
    }
```

```
This will throw an exception because Elon Musk has been fired

mustafa.karakas.hw1.BranchEmployeeDoesNotHaveAuthority: BranchEmployee does not work for any branch
```

Check if the product exists in the branch

```java
Product p;
if(Mark.removeProduct(p=new MeetingTable(1,1)))
{
    System.out.println("Given product is deleted "+p );
}
else{
    System.out.println("Given product does not exist in the branch "+Mark+ " is working");
}
```

```
Given product does not exist in the branch Mark Zuckerberg is working
```

Every product has finite number of models and colors. So we should check that if the model and color is valid.

```java
    System.out.println("\nProduct is not valid exception test\n");
    p = new MeetingTable(3,10);

}catch(TheProductDoesNotExistException e){
    System.out.println(e);
}catch(BranchEmployeeDoesNotHaveAuthority e){
```

```
Product is not valid exception test

mustafa.karakas.hw1.TheProductDoesNotExistException: We do not have the product you wanted to order
```

when we remove a branch , the branch's employees are fired.

```java
try{
    Customer Ahmet= new Customer("Ahmet","Yurt",null,null);// he can not buy online
    JeffBezos.makeSale(Ahmet,new MeetingTable(1,1));
    System.out.println(Ahmet+"'s customer number:"+Ahmet.getCustomerNumber());
    admin.removeBranch(4); // employees are now unemployed

    System.out.println("This will throw an exception because "+JeffBezos+" 's branch is removed so he is unemployed:");

    JeffBezos.makeSale(Ahmet,new MeetingTable(1,2));
}
```

```
This is Ahmet Yurt's first purchasing and he/she is registered to the company
Jeff Bezos informed manager to add Meeting Table Model :1 Color Type :1
Meeting Table Model :1 Color Type :1 added to 5. branch
Ahmet Yurt's customer number:1
the branch at the 4. index is removed. It's branch employees are now unemployed
This will throw an exception because Jeff Bezos 's branch is removed so he is unemployed:Ahmet's customer Number:
mustafa.karakas.hw1.BranchEmployeeDoesNotHaveAuthority: BranchEmployee does not work for any branch
```

We added JeffBezos again. And additionally, there is a method of Administrator which adds all the valid products to the given branch. I can not show it here because the is is very long. If you make the condition true you can see the list in the terminal.

```java
finally{
    admin.addEmployee(JeffBezos,3);
}
boolean seeAllProductsAreAdded = false; // this ruins the terminal appearance but it works

if(seeAllProductsAreAdded){
    try{
        admin.addAllProducts(3);
        JeffBezos.showBranchProducts();
    }
    catch(IndexOutOfBoundsException e){
        System.out.println(e);
    }
    catch(BranchEmployeeDoesNotHaveAuthority e){
        System.out.println(e);
    }
}
```

The customers can search for product before registration:

```
if(Veli.searchForProduct(p = new OfficeCabinet(4,1),c)){
    System.out.println(p+" is found at company");
}
else{
    System.out.println(p+" is not found at company");
}
for(int i=1;i<5;i++)
    JeffBezos.addProduct(new OfficeCabinet(i,1));
if(Veli.searchForProduct(p = new OfficeCabinet(4,1),c)){
    System.out.println(p+" is found at company");
}
else{
    System.out.println(p+" is not found at company");
}
```

```
Office CabinetModel :4 Color Type :1 is not found at company
Office CabinetModel :4 Color Type :1 is found at company
```

Veli did not register to the any company yet. So , that case has some negative effects:

```
Veli.inWhichStoreIsProduct(p);
```

```
You should register to company before searching something
```

Before registration , customer's customer numbers are -1.

```
p = new OfficeCabinet(4,1);
JeffBezos.makeSale(Veli, new OfficeCabinet(1,1));
System.out.println(p+" is in the "+ (Veli.inWhichStoreIsProduct(p)+1)+". branch");
System.out.println("Veli's customer number :"+ Veli.getCustomerNumber());
System.out.println("\nBranch products:");
JeffBezos.showBranchProducts();
Veli.viewPreviousOrders();
```

Veli has registered to the company.

```
This is Veli Turt's first purchasing and he/she is registered to the company
Office CabinetModel :4 Color Type :1 is in the 4. branch
Veli's customer number :2

Branch products:
Meeting Table Model :4 Color Type :1
Office Desk Model :4 Color Type :1
Office CabinetModel :2 Color Type :1
Office CabinetModel :3 Color Type :1
Office CabinetModel :4 Color Type :1

Veli Turt's previous orders:
Office CabinetModel :1 Color Type :1
```

Admin can ask for product need. If the branch does not have the product than adds it.

```
System.out.println("\nBranch products before addition:");
JeffBezos.showBranchProducts();

System.out.println("Admin asks for product need");
admin.askForProductNeed(new MeetingTable(2,1), 3);
admin.askForProductNeed(new MeetingTable(2,1), 3); // this does not add

System.out.println("\nBranch products after addition:");
JeffBezos.showBranchProducts();
JeffBezos.makeSale(Veli,new MeetingTable(2,1));
System.out.println("\nBranch products after selling the item:");
JeffBezos.showBranchProducts();
```

```
Branch products before addition:
Meeting Table Model :4 Color Type :1
Office Desk Model :4 Color Type :1
Office CabinetModel :2 Color Type :1
Office CabinetModel :3 Color Type :1
Office CabinetModel :4 Color Type :1

Admin asks for product need
Meeting Table Model :2 Color Type :1 added to 4. branch
Meeting Table Model :2 Color Type :1 has not added to 4. branch because there is already

Branch products after addition:
Meeting Table Model :4 Color Type :1
Office Desk Model :4 Color Type :1
Office CabinetModel :2 Color Type :1
Office CabinetModel :3 Color Type :1
Office CabinetModel :4 Color Type :1
Meeting Table Model :2 Color Type :1


Branch products after selling the item:
Meeting Table Model :4 Color Type :1
Office Desk Model :4 Color Type :1
Office CabinetModel :2 Color Type :1
Office CabinetModel :3 Color Type :1
Office CabinetModel :4 Color Type :1
```

Online shopping:

```java
System.out.println("\nOnline shopping");
Customer Temel = new Customer("Temel","Turan","asdf@gmail.com","1234");
Product p;
if(Temel.shopOnline(p=new OfficeCabinet(2,1),"Gebze/Kocaeli","0543 543 5443",c)){
    System.out.println(p+" is selled"+ " Temel's customer number :"
                        +Temel.getCustomerNumber());
}
else{
    System.out.println("You are registered to the another company");
}
System.out.println("\nBranch products after selling Online:");
JeffBezos.showBranchProducts();
// if the customer has registered to the another company shopOnline returns false

Company c2 = new Company();
Customer cust2 = new Customer("Unlucky","Customer","asdf@gmail.com","1234");
if(Temel.shopOnline(p=new OfficeCabinet(3,1),"Gebze/Kocaeli","0543 543 5443",c2)){
    System.out.println(p+" is selled");
}
else{
    System.out.println(Temel+" registered to the another company");
}
```

```
Online shopping
Office CabinetModel :2 Color Type :1 is selled Temel's customer number :3

Branch products after selling Online:
Meeting Table Model :4 Color Type :1
Office Desk Model :4 Color Type :1
Office CabinetModel :3 Color Type :1
Office CabinetModel :4 Color Type :1

Temel Turan registered to the another company
```

c2 company has no employee, so that this statements throws an exception:

```java
    cust2.shopOnline(p, "Gebze/Kocaeli","0543 543 5443",c2);
}
catch(ThereIsNoBranchEmployeeException e){
    System.out.println(e);
}
```

```
mustafa.karakas.hw1.ThereIsNoBranchEmployeeException: There is no branchEmployee to do that job
```

- **RUNNING AND RESULTS**

I have added almost all of my test code results to here. I hope checking it is easier now.