GEBZE TECHNICAL UNİVERSITY

COMPUTER ENGINEERING

CSE222-2021

HOMEWORK 03 REPORT

AHMET FURKAN KURBAN

1801042674

1 INTRODUCTİON

1.1 Problem Definition

In this homework, reuse the same scenario of Homework 1 and implement the same system using several implementations of the List abstract data structure.

• You should complete the ArrayList and LinkedList implementations in the textbook.

• You should implement a HybridList class that keeps a LinkedList as a component and the elements stored in the LinkedList are ArrayLists. The number of elements in each ArrayList should be less than MAX_NUMBER. When the number of elements in an ArrayList exceeds MAX_NUMBER a new ArrayList should be generated in the LinkedList. When there is no element in an ArrayList it should be removed from the LinkedList.

System Requirements

I added only new containers such as Arraylist Doublinkedlist and Hybridlist so i added only these codes because Automation code is same before only some functions name is changed.

-Arraylist-

public boolean add(E element)

Adding element to the end

public void add(int index,E element)

Adding element to the given index

public E get(int index)

return element of given index

public E set(int index , E newValue)

change element of given index and return old element

public E remove(int index)

remove element of given index

public boolean equals(Object o)

Find the correct element of given object

public void removeElement(Object element)

Remove element from array of given object

public int contains(Object o)

Check element in the array

Double Linkedlist

public void addFirst(E element)

Add element to the first

public void add(E element)

Add element to the last

public E get(int index)

Return element of given index

public void add(int index,E element)

add element of given index

public int indexOf(Object o)

Search element of given index

public E remove(int index)

remove element from end

public Node<E> getNode (int index)

return node of given index

Hybrid List

HybridList class that keeps a LinkedList as a component and the elements stored in the LinkedList are ArrayLists. The number of elements in each ArrayList is less than MAX_NUMBER. When the number of elements in an ArrayList exceeds MAX_NUMBER a new ArrayList generated in the LinkedList. When there is no element in an ArrayList it removed from the LinkedList.

public void add(E element)

Add element to the end if size more than maxnumber after added new node is created and added to the next node to before node .

public void add(E element,int index)

Add eleement to the given index if size more than maxnumber after added new node is created and added to the next node to before node if after added element the node has this element exceed the maxnumber i remove last element of this node and i add deleted element to the head of next node i keep this to the reach last node so every node has maximum maxnumber elements in it.

public E get(int index)-> E element=linkedlist.get(number1).get(number2);

number1=index/maxnumber; number2=index%maxnumber; so i return number1.node and number2.element from arraylist.

public E set(int index , E newValue)->E element=linkedlist.get(number1).set(number2);

number1=index/maxnumber; number2=index%maxnumber

change the element of given index and return old element.

public E remove (int index)

Remove element from last if after removing there is no element that has node i removed this node.

public E remove ()

Remove element from last if after removing there is no element that has node i removed this node.When i remove element from given index if next node has element i remove first element of nex node and i add before node so every node has maxnumber except last node

public void Print()

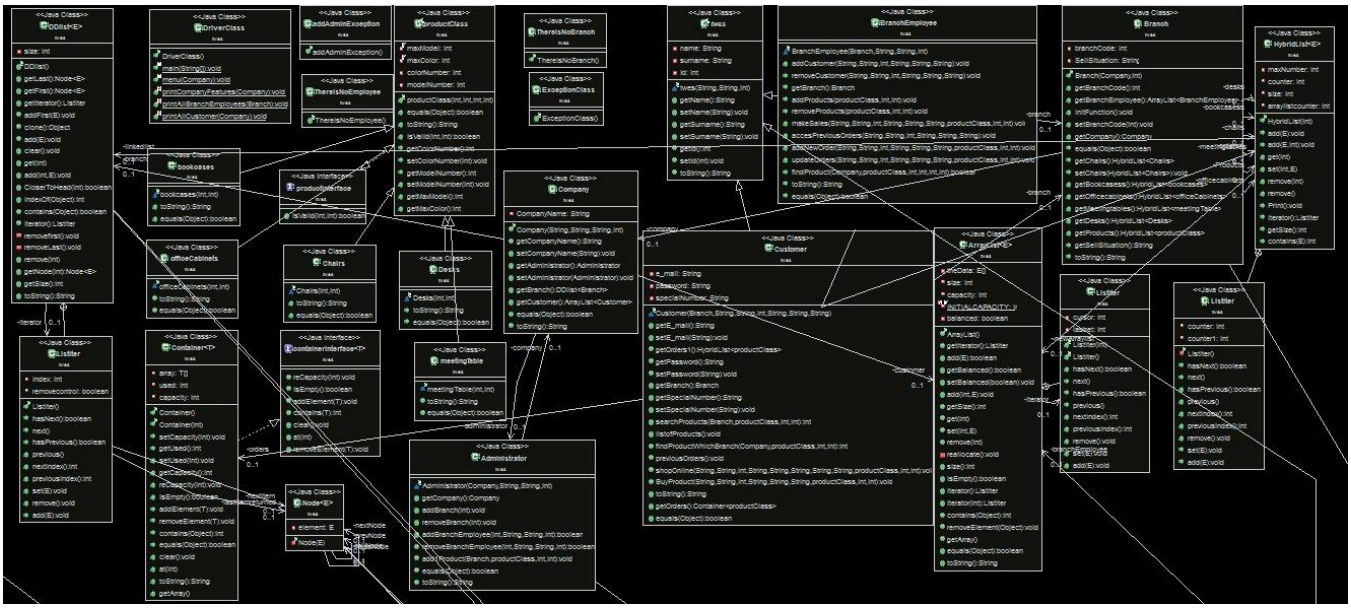
Print the elements.

public int contains(E element)

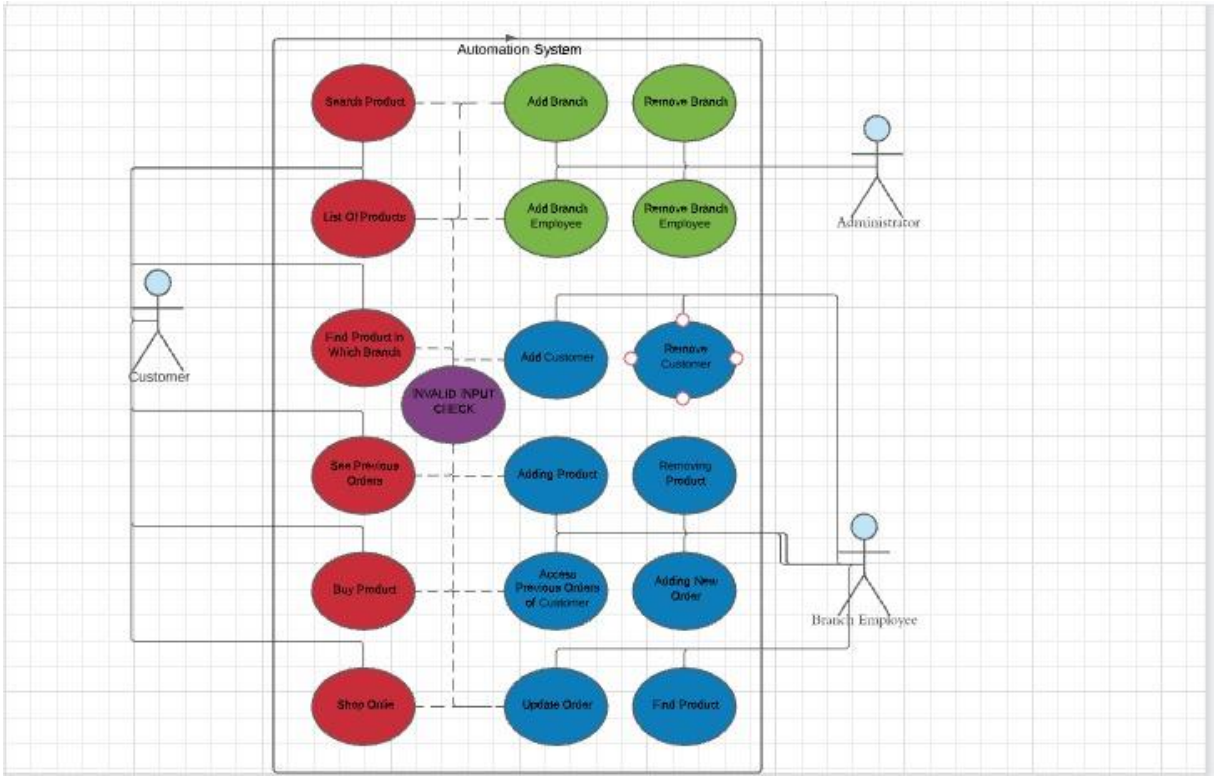
Check the elements all node and all arraylist element.

2.1 Diagrams

Class Diagram -UML-



Use Case Diagram



2.2 Problem Solution Approach

I only add new container such as arraylist hybridlist and doublelinkedlist and test them.rest is same with hw1 only some functions name changed such as at->get.

3 RESULT

3.1 Test Cases

Hybrid list

Here i want to show hybridlist works ,at automation system it is working but i want to show this too.code is availeble at hybridlist class.

[0 1 2 12 3 4 5 6 7 8][9 10 11 12 13 14 15 16 17 18][19 20 21 22 23 24 25 26 27 28][29 30 31 32 33 34 35 36 37 38][39 40 41 42 43 44 45 46 47 48][49]

919293949[]

203040114111[]

11

[0 1 2 3 4 5 6 7 8 9][10 12 12 13 14 15 17 18 19 20][21 22 23 24 25 16 26 27 28 29][30 31 32 33 34 35 36 38 39 40][41 42 43 44 45 46 47 48 11 11][11]

Create a company with administrator-

```
Company company=new Company("Kurban Holding","Adem","Catal",10);
```

Kurban Holding

Administrator : Adem Catal

Administrator:

-Adding Branch:

Branch is added to the this company by Administrator

This branch has already exist at company (Already Exist Branch added one more)

Branch is deleted from this company by Administrator

There is no branch like that at company (Already Exist Branch removed one more)

Branches:

Kurban Holding --> 1. Branch

Kurban Holding --> 2. Branch

-Adding Branch Employee

Adding branch employee is successful.

Company Name: Kurban Holding

Branch Code: 2

Name: Mehmet

Surname: kurban

ID: 1234

-Removing Branch Employee

Removing Branch Employee

Removing branch employee is successful.

Company Name: Kurban Holding

Branch Code: 2

Name: Ali

Surname: kurban

ID: 12345

Branch Employee:

-Adding a Customer

Adding customer operation is successful.

Customer : alibaba@gmail.com21323512321

Name: Ali

Surname: Baba

ID: 123457

-Removing a Customer

Removing customer operation is successful.

Customer : alibaba@gmail.com21323512321

Name: Ali

Surname: Baba

ID: 123457

-Adding Products

Employee is adding the product

DesksproductClass [maxModel=5, maxColor=4, colorNumber=5, modelNumber=3]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=5, modelNumber=4]

-Removing Products

Employee is deleting the product

DesksproductClass [maxModel=5, maxColor=4, colorNumber=5, modelNumber=3]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=3, modelNumber=3]

Here firstly i add then i delete desk 3(model),3(color) then i added desk 3,3 again i want to show you what happens after adding and removing.

-Access Previous Orders of Customer

Employee is looking previous orders of customer

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=1]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=2]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=3]

-Adding New Order

Orders succesfully Added by customer

OfficeChairproductClass [maxModel=7, maxColor=5, colorNumber=1, modelNumber=1]

OfficeChairproductClass [maxModel=7, maxColor=5, colorNumber=2, modelNumber=2]

OfficeChairproductClass [maxModel=7, maxColor=5, colorNumber=3, modelNumber=3]

-Update Order After makesale-

After Make Sales (makesales function have updatedorders function)

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=3]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=2]

Orders succesfully Updated after make sales

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=2]

-FindProduct

```
employee.findProduct(company,desks,1,3,1,1);
```

Element is not founded at that branch so we said admin then admin have added product the branch.(if element is not in branch)

Customer:

-Search Product

```
customer.searchProducts(branch,desk,3,3);
```

Customer is searching the product...

Products is founded index :10

-list Of Products

```
customer.listofProducts();
```

There is whole products but i added 4 of them to show (you can see the terminal whole products)

meetingTableproductClass [maxModel=10, maxColor=4, colorNumber=10, modelNumber=3]

meetingTableproductClass [maxModel=10, maxColor=4, colorNumber=10, modelNumber=4]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=1]

DesksproductClass [maxModel=5, maxColor=4, colorNumber=1, modelNumber=2]

-find product in which branch

```
customer.findProductWhichBranch(company, desk,3, 3);
```

Customer is searching the product is which Branch...

Products is founded at that branch index :1

-see previous orders

```
customer.previousOrders();
```

Customer is looking the his/her previous orders...

OfficeChairproductClass [maxModel=7, maxColor=5, colorNumber=1, modelNumber=1]

OfficeChairproductClass [maxModel=7, maxColor=5, colorNumber=1, modelNumber=2]

OfficeChairproductClass [maxModel=7, maxColor=5, colorNumber=1, modelNumber=3]

-Buy Product

Welcome you are new subscribes we add you!

DesksproductClass [maxModel=5, maxColor=4, colorNumber=3, modelNumber=4]Customer : ElonMusk@gmail.com231212312

Name: Elon

Surname: Musk

ID: 1234567

Customer : AlisQurtua@gmail.com13235123212

Name: Alis

Surname: Qurtua

ID: 123456

Customer : burakKoca@gmail.com1323512

Name: Burak

Surname: Koca

ID: 12331

```
customer.BuyProduct("Elon","Musk",1234567,"ElonMusk@gmail.com","231212312","1",desk,4,4);

employee.addNewOrder(name,surname, id, e_mail, password, specialNumber, product, number, number1);

employee.accesPreviousOrders(name, surname, id, email, password, SpecialNumber);

employee.makeSales(name,surname, id, e_mail, password, specialNumber, product, number, number1);

removeProducts(product,number,number1);

updateOrders(name,surname,id,email,password,SpecialNumber,product,number,number1);
```

Customer is buying product...

Here if customer want to buy product first function is working then employee add new order to customer orders and with previous orders we can see the customers orders then employee make sale to customer , bought product is removing from that branch lastly with update orders function employee remove the orders that customer needed from customer orders but previous orders is still has customer so we can see if we want anytime.

Asymptotic Notations

```
public void setBranchCode(int branchCode) {
    this.branchCode = branchCode;
}
public Company getCompany() {
    return company;
}
@Override
public boolean equals(Object o)
{
    if(o == null)
        return false;
    if(!(o instanceof Branch))
        return false;

    return getCompany().equals(((Branch)o).getCompany()) && getBranchCode() == ((Branch)o).getBranchCode();
}

public HybridList<Chairs> getChairs() {
    return chairs;
}
public void setChairs(HybridList<Chairs> chairs) {
    this.chairs = chairs;
}
public HybridList<bookcases> getBookcasess() {
    return bookcasess;
}
public HybridList<officeCabinets> getOfficecabinets() {
    return officecabinets;
}
public HybridList<meetingTable> getMeetingtables() {
    return meetingtables;
}
public HybridList<Desks> getDesks() {
    return desks;
}
public HybridList<productClass> getProducts() {
    return Products;
}
public String getSellSituation() {
```

Branch Codes

setBranchCode() = O(1)
getCompany() = O(1)
equals() = O(1) + O(1) + O(1) = O(1)
getChairs() = O(1)
setChairs() = O(1)
getBookcases() = O(1)
getOfficecabinets() = O(1)
getBranchEmployee = O(1)

getmeetingtables = O(1)
getDesks() = O(1)
getProducts = O(1)
getSellsituation = O(1)

```java
public int getBranchCode() {
    return branchCode;
}
public ArrayList<BranchEmployee> getBranchEmployee() {
    return branchEmployee;
}
public void initFunction() {
    int number=0,number1=0;
    Products.add(new Chairs(1,1));
    Products.add(new bookcases(1,1));
    Products.add(new officeCabinets(1,1));
    Products.add(new meetingTable(1,1));
    Products.add(new Desks(1,1));
    for(int i=0;i<5;i++) {
        number=Products.get(i).getMaxModel();
        number1=Products.get(i).getMaxColor();
        for(int j=0;j<number;j++) {
            if(i==1) {
                bookcasess.add(new bookcases(j+1,1));
            }
            if(i==2) {
                officecabinets.add(new officeCabinets(j+1,1));
            }
            for(int k=0;k<number1;k++) {
                if(i==0) {
                    chairs.add(new Chairs(j+1,k+1));
                }
                if(i==3) {
                    meetingtables.add(new meetingTable(j+1,k+1));
                }
                if(i==4) {
                    desks.add(new Desks(j+1,k+1));
                }
            }
        }
    }
}
```

initFunction() = $T_1(n)_{productsAdd(1)} + T_2(n)_{productsget()} + (T_3 productsAdd(n) \& T_4(n))$ (for)

$T_1 productsAdd(n) = \underbrace{T_1 DOIlistget(n)}_{O(n)} + \underbrace{T_2 DOIlistadd(n)}_{O(1)} + \underbrace{T_3 DOIlistget(n)}_{O(n)} + \underbrace{T_4 AAlistadd(n)}_{O(1)}$ → Last add

$T_1 productAdd(n) = O(n)$ //

$T_2 Productsget(n) = \underbrace{T_1 DOIlistget(n)}_{O(n)} + \underbrace{T_2 AAlistget(n)}_{O(1)}$

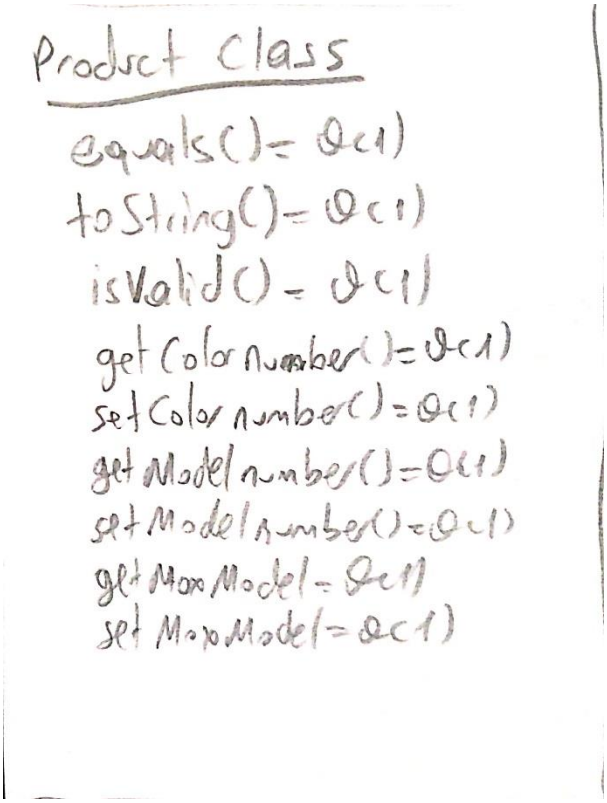
$O(n)$

$T_2 productsget(n) = O(n)$

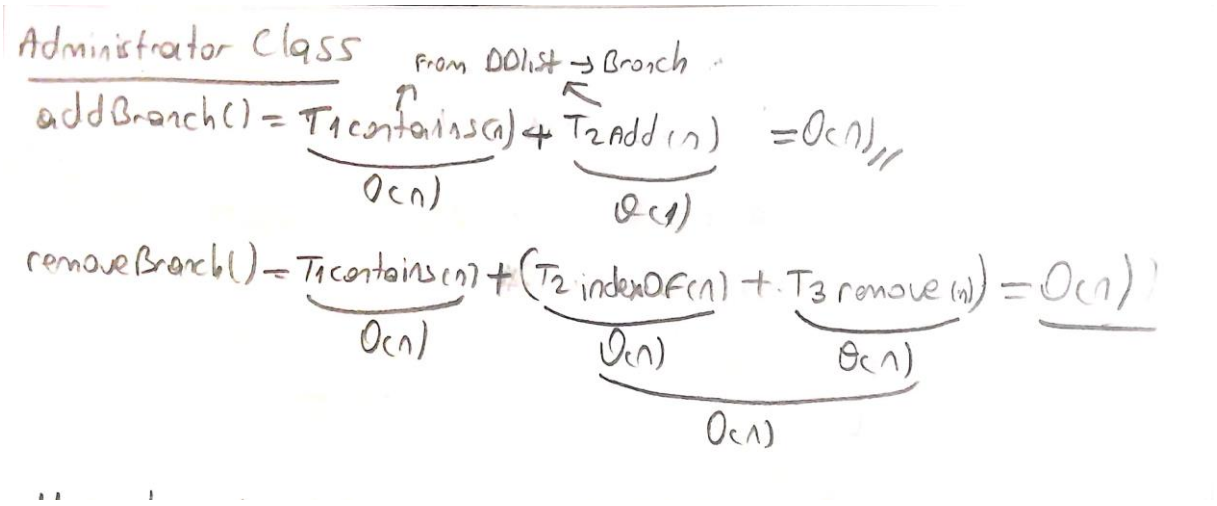
$\underbrace{T_3 ProductsAdd(n)}_{O(n) \text{ from upper}} \& \underbrace{T_4(n)}_{O(n)} = O(n^2)$ (For)

initFunction(n) = $O(n) + O(n) + O(n^2)$
= $O(n^2)$ //

```java
public int getColorNumber() {
    return colorNumber;
}


public void setColorNumber(int colorNumber) {
    this.colorNumber = colorNumber;
}


public int getModelNumber() {
    return modelNumber;
}


public void setModelNumber(int modelNumber) {
    this.modelNumber = modelNumber;
}


public int getMaxModel() {
    return maxModel;
}


public int getMaxColor() {
    return maxColor;
}
```



```
     * @param branchCode is the unique code for the branch.
     */
    public void addBranch(int branchCode) {//we add branch by using branch code
        Branch branch=new Branch(getCompany(),branchCode);//make a new branch for control
        if(getCompany().getBranch().contains(branch)==false) {//check branch is contains at that company ----
            getCompany().getBranch().add(branch);//add branch of that company
            System.out.println("Branch is added to the this company by Administrator");
        }
        else {
            System.out.println("This branch has already exist at company");
        }
    }
    /**
     * Remove the existing branch.
     * @param branchCode is the target branch to remove.
     */
    public void removeBranch(int branchCode) {
        Branch branch=new Branch(getCompany(),branchCode);
        if(getCompany().getBranch().contains(branch)!=false) { //----------
            int a=getCompany().getBranch().indexOf(branch);
            getCompany().getBranch().remove(a);//remove branch from that company
            System.out.println("Branch is deleted from this company by Administrator");
        }
        else {
            System.out.println("There is no branch like that at company");
        }
    }
    /**
```



```java
public boolean addBranchEmployee(int branchCode,String name,String surname,int id) throws ThereIsNoBranch {
    int index=-1;
    Branch branch=new Branch(getCompany(),branchCode);
    index=getCompany().getBranch().indexOf(branch);//----------
    if(index==-1) {
        throw new ThereIsNoBranch();
    }
    if(index!=-1) {//check that branch is contains of that company
        BranchEmployee branchemployee =new BranchEmployee(getCompany().getBranch().get(index),name,surname,id);//ma
        if(getCompany().getBranch().get(index).getBranchEmployee().contains(branchemployee)==-1) {
            getCompany().getBranch().get(index).getBranchEmployee().add(branchemployee);
            return true;
        }
        else {
            System.out.println("This employee already exist in that branch!");
        }
    }
    return false;
}
 /**
 * Remove the existing branch employee from a branch.
 * @param branchCode the target branch to remove the current branch employee.
 * @return true if the branch employee is removed. Otherwise returns false.
 */
public boolean removeBranchEmployee(int branchCode,String name,String surname,int id)throws ThereIsNoBranch {
    int index=-1;
    Branch branch=new Branch(getCompany(),branchCode);
    index=getCompany().getBranch().indexOf(branch);//---------
    if(index==-1) {
        throw new ThereIsNoBranch();
    }
    if(index!=-1) {
        BranchEmployee branchemployee =new BranchEmployee(getCompany().getBranch().get(index),name,surname,id);
        getCompany().getBranch().get(index).getBranchEmployee().removeElement(branchemployee);
        return true;
    }
    return false;
}
```

addBranchemployee = T_1 indexof(n) + T_2 get(n) + T_3 contains(n) + T_4 get(n) + T_5 add(n)

indexof() → O(n) ; get(), contains() → O(n) ; get().add() → O(1)

$T_1 = O(n)$ $T_4() = O(n)$

$T_2 = O(n)$ $T_5() = O(1)$ $T_{addBranchEmployee} = O(n)$

$T_3 = O(n)$

removeBranchEmployee = T_1 indexof(n) + T_2 get(n) + T_3 get(n) + T_4 removeElement

$T_1 = O(n)$ $T_4() = O(n^2)$ → for + get()

$T_2 = O(n)$

$T_3 = O(n)$ $T_{removeBranchEmployee} = O(n^2)$

get(), remove().

```java
public void add1Product(Branch branch,productClass product,int number,int number1) {//this functior
    if(product instanceof Chairs) {
        branch.getChairs().add(new Chairs(number,number1));
    }
    if(product instanceof bookcases) {
        branch.getBookcasess().add(new bookcases(number,number1));
    }
    if(product instanceof officeCabinets) {
        branch.getOfficecabinets().add(new officeCabinets(number,number1));
    }
    if(product instanceof meetingTable) {
        branch.getMeetingtables().add(new meetingTable(number,number1));
    }
    if(product instanceof Desks) {
        branch.getDesks().add(new Desks(number,number1));
    }
}
@Override
public boolean equals(Object o)
{
    if(o == null)
        return false;

    return  getCompany().equals(((Administrator)o).getCompany()) && super.equals(o);
}

@Override
public String toString()
{
    return "Company : " + getCompany().getCompanyName() + "\n"+ super.toString();
}
```

add1product() = $T_1 add(n) \rightarrow T_2 get(n) + T_3 add(n) + T_4 get(n) + T_5 add()$

$T_2 get(n)$: $O(n)$ — $T_3 add(n)$: $O(1)$ — $T_4 get(n) + T_5 add()$: $O(1)$ (ArrayList add) — get(), add()

$T_{add1product} = O(n)$

$O(n)$

equals() = $O(1)$

toString() = $O(1)$

```java
public void addCustomer(String name,String surname,int id,String email,String password,String SpecialNumber) {
    Customer customer=new Customer(branch,name,surname,id,email,password,SpecialNumber);//we add customer to branch
    if(getBranch().getCompany().getCustomer().contains(customer) == -1)//if branch has not this customer
        getBranch().getCompany().getCustomer().add(customer);
    else {
        System.out.println("This customer is added before so you can not add again same person to the branch!");
    }
}
/**
 * Remove the existing user from the company.
 * @param name of the user.
 * @param surname of the user.
 * @param id of the user.
 */
public void removeCustomer(String name,String surname,int id,String email,String password,String SpecialNumber) {
    Customer customer=new Customer(branch,name,surname,id,email,password,SpecialNumber);
    if(getBranch().getCompany().getCustomer().contains(customer) != -1){
        getBranch().getCompany().getCustomer().removeElement(customer);
    }
    else {
        System.out.println("This customer is not in that branch so you can not remove it!");
    }
}
public Branch getBranch(){
    return branch;
}
```

Branch Employee class

addCustomer() = T_1 contains(n) + T_2 add(1) = $O(n^2)$ //

contains(n): $O(n^2)$

For * equals

For: $O(n)$, equals: $O(n)$

For + $O(1)$

$O(n)$

add: $O(1)$

removeCustomer() = T_1 contains(n) + T_2 removeElement(n) = $O(n^2)$ //

$O(n^2)$ +

For * get

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

$O(n^2)$

```java
public void addProducts(productClass product,int number,int number1) throws ArrayIndexOutOfBoundsException{// add product at that branch
    if(product instanceof Chairs) {
        if(number1 < 0 || number1-1 >= getBranch().getChairs().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getChairs().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        getBranch().getChairs().add(new Chairs(number,number1));
    }
    if(product instanceof bookcases) {
        if(number1 < 0 || number1-1 >= getBranch().getBookcasess().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getBookcasess().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        getBranch().getBookcasess().add(new bookcases(number,number1));
    }
    if(product instanceof officeCabinets) {
        if(number1 < 0 || number1-1 >= getBranch().getOfficecabinets().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getOfficecabinets().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        getBranch().getOfficecabinets().add(new officeCabinets(number,number1));
    }
    if(product instanceof meetingTable) {
        if(number1 < 0 || number1-1 >= getBranch().getMeetingtables().get(1).getMaxColor() || number-1 < 0 || number >= getBranch().getMeetingtables().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        getBranch().getMeetingtables().add(new meetingTable(number,number1));
    }
    if(product instanceof Desks) {
        if(number1 < 0 || number1-1 >= getBranch().getDesks().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getDesks().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        getBranch().getDesks().add(new Desks(number,number1));
    }
}
```

addProducts() = T1 get() + T2 add(n) = O(n) + O(n) = O(n)

get() → HybridList → get() → DLL O(n)

get() → ALL O(n)

add() → HybridList → add() → DLL O(1)

add() → ALL O(n)

get() → DLL O(n)

```java
public void removeProducts(productClass product,int number,int number1)throws ArrayIndexOutOfBoundsException {
    if(product instanceof Chairs) {
        if(number1 < 0 || number1-1 >= getBranch().getChairs().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getChairs().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        int a=getBranch().getChairs().contains((Chairs) product);
        getBranch().getChairs().remove(a);
    }
    if(product instanceof bookcases) {
        if(number1 < 0 || number1-1 >= getBranch().getBookcasess().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getBookcasess().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        int a=getBranch().getBookcasess().contains((bookcases) product);
        getBranch().getBookcasess().remove(a);
    }
    if(product instanceof officeCabinets) {
        if(number1 < 0 || number1-1 >= getBranch().getOfficecabinets().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getOfficecabinets().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        int a=getBranch().getOfficecabinets().contains((officeCabinets) product);
        getBranch().getOfficecabinets().remove(a);
    }
    if(product instanceof meetingTable) {
        if(number1 < 0 || number1-1 >= getBranch().getMeetingtables().get(1).getMaxColor() || number-1 < 0 || number >= getBranch().getMeetingtables().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        int a=getBranch().getMeetingtables().contains((meetingTable) product);
        getBranch().getMeetingtables().remove(a);
    }
    if(product instanceof Desks) {
        if(number1 < 0 || number1-1 >= getBranch().getDesks().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getDesks().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
        int a=getBranch().getDesks().contains((Desks) product);
        getBranch().getDesks().remove(a);
    }
}
```

removeProducts() = T1 get(n) + T2 contains(n) + T3 remove(n)

O(n) O(n²) O(n²)

Hybrid list → while + O(n) remove get add

get() → O(n)

remove() → DLL → O(n)

O(n) + O(n) + O(n²)

T removeProducts(n) = O(n²)

```java
public void makeSales(String name,String surname,int id,String email,String password,String SpecialNumber,productClass product,int number,int number1) {
    int index=-1;
    Customer customer=new Customer(getBranch(),name,surname,id,email,password,SpecialNumber);
    if((index = getBranch().getCompany().getCustomer().contains(customer)) == -1) {//if first shopping of customer we add customer to the that branch
        System.out.println("Welcome you are new subscribes we add you!");
        getBranch().getBranchEmployee().get(1).addCustomer(name,surname,id,email,password,SpecialNumber); //random employee dosent matter
    }
    removeProducts(product,number,number1);//remove product from branch
    updateOrders(name,surname,id,email,password,SpecialNumber,product,number,number1);//remove product from order list of customer
}
public void accesPreviousOrders(String name,String surname,int id,String email,String password,String SpecialNumber) {
    int index=0;
    Customer customer=new Customer(branch,name,surname,id,email,password,SpecialNumber);
    if((index = getBranch().getCompany().getCustomer().contains(customer)) != -1) {
        for(int i=0;i<getBranch().getCompany().getCustomer().get(index).getOrders().getUsed();i++) {
            System.out.println(getBranch().getCompany().getCustomer().get(index).getOrders().at(i));
        }
    }
    else {
        System.out.println("This Customer is not in that branch so you can not access previous orders of customer");
    }
}
```

$$MakeSales() = T_1 contains(n) + T_2 addCustomer(n) + T_3 RemoveProducts(n) + T_4 updateOrders(n)$$

$T_1 contains(n) = O(n^2)$
$T_2 addCustomer(n) = O(n^2)$
$T_3 removeProducts(n) = O(n^2)$
$T_4 updateOrders(n) = O(n^2)$

$\Rightarrow T_{MakeSales}(n) = O(n^2)$

$$accesPreviousOrders() = \underbrace{T_1 contains(n)}_{O(n^2)} + \underbrace{T_2 for(n)}_{O(n)} \underbrace{T_3 get(n)}_{O(n)}$$

$$= O(n^2)$$

```
public void addNewOrder(String name,String surname,int id,String email,String password,String SpecialNumber,productClass product,int number,int number1) throws Array
{

    int index=0;
    Customer customer=new Customer(branch,name,surname,id,email,password,SpecialNumber);
    if((index = getBranch().getCompany().getCustomer().contains(customer)) != -1) {
        if(product instanceof Chairs) {
            if(number1 < 0 || number1-1 >= getBranch().getChairs().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getChairs().get(1).getMaxModel())
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().addElement(new Chairs(number,number1));
            getBranch().getCompany().getCustomer().get(index).getOrders1().addElement(new Chairs(number,number1));
        }
        if(product instanceof bookcases) {
            if(number1 < 0 || number1-1 >= getBranch().getBookcasess().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getBookcasess().get(1).getMaxMod
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().addElement(new bookcases(number,number1));
            getBranch().getCompany().getCustomer().get(index).getOrders1().addElement(new Chairs(number,number1));
        }
        if(product instanceof officeCabinets) {
            if(number1 < 0 || number1-1 >= getBranch().getOfficecabinets().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getOfficecabinets().get(1).g
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().addElement(new officeCabinets(number,number1));
            getBranch().getCompany().getCustomer().get(index).getOrders1().addElement(new Chairs(number,number1));
        }
        if(product instanceof meetingTable) {
            if(number1 < 0 || number1-1 >= getBranch().getMeetingtables().get(1).getMaxColor() || number-1 < 0 || number >= getBranch().getMeetingtables().get(1).get
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().addElement(new meetingTable(number,number1));
            getBranch().getCompany().getCustomer().get(index).getOrders1().addElement(new Chairs(number,number1));
        }
        if(product instanceof Desks) {
            if(number1 < 0 || number1-1 >= getBranch().getDesks().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getDesks().get(1).getMaxModel())
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().addElement(new Desks(number,number1));
            getBranch().getCompany().getCustomer().get(index).getOrders1().addElement(new Chairs(number,number1));
        }
    }
    else {
        System.out.println("This Customer is not in that branch so you can not add new orders to customer");
```

addNewOrder() = T_1 contains(n) + T_2 get(n) + T_3 getOrders + T_4 get(n) + T_5 add(n)

$T_1(n) = O(n^2)$
$T_2(n) = O(n)$
$T_3(n) = O(1)$
$T_4(n) = O(1)$
$T_5(n) = O(1)$

$O(n^2)$

```
public void updateOrders(String name,String surname,int id,String email,String password,String SpecialNumber,productClass product,int number,int number1)throws Array
    int index=0;
    Customer customer=new Customer(branch,name,surname,id,email,password,SpecialNumber);
    if((index = getBranch().getCompany().getCustomer().contains(customer)) != -1) {
        if(product instanceof Chairs) {
            if(number1 < 0 || number1-1 >= getBranch().getChairs().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getChairs().get(1).getMaxModel())
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().removeElement(new Chairs(number,number1));
        }
        if(product instanceof bookcases) {
            if(number1 < 0 || number1-1 >= getBranch().getBookcasess().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getBookcasess().get(1).getMaxMod
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().removeElement(new bookcases(number,number1));
        }
        if(product instanceof officeCabinets) {
            if(number1 < 0 || number1-1 >= getBranch().getOfficecabinets().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getOfficecabinets().get(1).g
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().removeElement(new officeCabinets(number,number1));
        }
        if(product instanceof meetingTable) {
            if(number1 < 0 || number1-1 >= getBranch().getMeetingtables().get(1).getMaxColor() || number-1 < 0 || number >= getBranch().getMeetingtables().get(1).get
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().removeElement(new meetingTable(number,number1));
        }
        if(product instanceof Desks) {
            if(number1 < 0 || number1-1 >= getBranch().getDesks().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getDesks().get(1).getMaxModel())
                throw new ArrayIndexOutOfBoundsException("Invalid index!");
            getBranch().getCompany().getCustomer().get(index).getOrders().removeElement(new Desks(number,number1));
        }
    }
    else {
        System.out.println("This Customer is not in that branch so you can not update new orders to customer");
    }
}
```

updateOrders() = T_1 contains + T_2 get(n) + T_3 getOrders(n) + T_4 get(n) + T_5 remove(n)

$T_1 = O(n^2)$
$T_2 = O(n)$
$T_3 = O(1)$
$T_4 = O(1)$
$T_5 = O(n^2)$

$O(n^2)$

```
public boolean findProduct(Company company,productClass product,int number1,int number2,int number3,int number4) {//check neede
    int index=-1;
    index=company.getCustomer().get(number3).searchProducts(company.getBranch().get(number4), product, number1, number2);

    if(index==-1) {
        branch.getCompany().getAdministrator().add1Product(branch,product,number1,number2);//if product is not in branch it cal
        System.out.println("Element is not founded at that branch so we said admin then admin have added product the branch");
        return true;
    }
    else {
        return true;
    }

}
public String toString()
{
    return  "Company Name: " + getBranch().getCompany().getCompanyName() + "\n" +
            "Branch Code: " + getBranch().getBranchCode() + "\n" +
            super.toString();
}
@Override
public boolean equals(Object o)
{
    if(o == null)
        return false;
    if(!(o instanceof BranchEmployee))
        return false;

    return getBranch().equals(((BranchEmployee)o).getBranch()) && super.equals(o);
}
```

findProduct() = T_1get(n) + T_2searchProducts(n) + T_3add1product(n)

$O(1)$ $O(n^2)$ $O(n)$

$O(n^2)$

```
public int searchProducts(Branch branch,productClass product,int number,int number1)throws ArrayIndexOutOfBoundsException {//search product is in that branch or not
    int index=-1;
    if(product instanceof Chairs) {
        if(number1 < 0 || number1-1 >= getBranch().getChairs().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getChairs().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
            if((index=branch.getChairs().contains((Chairs) product)) !=-1) {
                return index;
            }

    }
    if(product instanceof bookcases) {
        if(number1 < 0 || number1-1 >= getBranch().getBookcasess().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getBookcasess().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
            if((index=branch.getBookcasess().contains((bookcases)product)) !=-1) {
                return index;
            }

    }
    if(product instanceof officeCabinets) {
        if(number1 < 0 || number1-1 >= getBranch().getOfficecabinets().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getOfficecabinets().get(1).getMaxModel
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
            if((index=branch.getOfficecabinets().contains((officeCabinets)product)) !=-1) {
                return index;
            }
    }
    if(product instanceof meetingTable) {
        if(number1 < 0 || number1-1 >= getBranch().getMeetingtables().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getMeetingtables().get(1).getMaxModel()
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
            if((index=branch.getMeetingtables().contains((meetingTable)product)) !=-1) {
                return index;
            }

    }
    if(product instanceof Desks) {
        if(number1 < 0 || number1-1 >= getBranch().getDesks().get(1).getMaxColor() || number < 0 || number-1 >= getBranch().getDesks().get(1).getMaxModel())
            throw new ArrayIndexOutOfBoundsException("Invalid index!");
            if((index=branch.getDesks().contains((Desks)product)) !=-1) {
```

searchProducts() = T_1get(n) + T_2contains(n) $= O(n^2)$

$O(n)$ $O(n^2)$

```
public void listofProducts() {
    for(int i=0;i<getBranch().getChairs().getSize();i++){
        System.out.println(getBranch().getChairs().get(i));
        System.out.println("----------------------------------------");
    }

    for(int i=0;i<getBranch().getBookcasess().getSize();i++) {
        System.out.println(getBranch().getBookcasess().get(i));
        System.out.println("----------------------------------------");
    }

    for(int i=0;i<getBranch().getOfficecabinets().getSize();i++) {
        System.out.println(getBranch().getOfficecabinets().get(i));
        System.out.println("----------------------------------------");
    }

    for(int i=0;i<getBranch().getMeetingtables().getSize();i++) {
        System.out.println(getBranch().getMeetingtables().get(i));
        System.out.println("----------------------------------------");
    }

    for(int i=0;i<getBranch().getDesks().getSize();i++) {
        System.out.println(getBranch().getDesks().get(i));
        System.out.println("----------------------------------------");
    }
}
```

listofproducts() = $T_1$for(n) * T_2get(n) = O(n²)

O(n) O(n)

```
public int findProductWhichBranch(Company company,productClass product,int number,int number1) {//return branch index that have that p
    int index=-1;
    if(product instanceof Chairs) {
        for(int j=0;j<company.getBranch().getSize();j++) {
                if((index=company.getBranch().get(j).getChairs().contains((Chairs)product)) !=-1) {
                    index=j;
                    return index;
                }

        }
    }
    if(product instanceof bookcases) {
        for(int j=0;j<company.getBranch().getSize();j++) {
                if((index=company.getBranch().get(j).getBookcasess().contains((bookcases)product)) !=-1) {
                    index=j;
                    return index;
                }

        }
    }
    if(product instanceof officeCabinets) {
        for(int j=0;j<company.getBranch().getSize();j++) {
                if((index=company.getBranch().get(j).getOfficecabinets().contains((officeCabinets)product)) !=-1) {
                    index=j;
                    return index;
                }

        }
    }
    if(product instanceof meetingTable) {
        for(int j=0;j<company.getBranch().getSize();j++) {
                if((index=company.getBranch().get(j).getMeetingtables().contains((meetingTable)product)) !=-1) {
                    index=j;
                    return index;
                }
```

FindProductWhichBranch() = $T_1$for(n) * T_2contains(n) = O(n³)

O(n) O(n²)

```java
public void previousOrders() {//previous orders of customer
    for(int i=0;i<getOrders1().getUsed();i++) {
        System.out.println(getOrders1().at(i));
    }
}
public void shopOnline(String name,String surname,int id,String email,String password,String SpecialNumber,String adress , String Telno,productCl
    System.out.println("Online Shopping ");
    BuyProduct(name,surname,id,email,password,SpecialNumber,product,number,number1);
}
public void BuyProduct(String name,String surname,int id,String email,String password,String SpecialNumber,productClass product,int number,int nu
    int index=-1;
    index=findProductWhichBranch(getBranch().getCompany(),product,number,number1);
    if(index!=-1) {
     System.out.println("Products is founded at that branch index :" +(index+1));
    }
    else {
        System.out.println("Products is not founded at any branch");
    }
    BranchEmployee employee=getBranch().getCompany().getBranch().get(index).getBranchEmployee().get(1);
    if(employee==null) {
        throw new ThereIsNoEmployee();
    }
    else {
        employee.addNewOrder(name,surname, id, e_mail, password, specialNumber, product, number, number1);//add element of customer order
        employee.accesPreviousOrders(name, surname, id, email, password, SpecialNumber);//to see orders
        employee.makeSales(name,surname, id, e_mail, password, specialNumber, product, number, number1);//and sale
    }
}
```