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

Sena Nur Ulukaya
1901042622

1. SYSTEM REQUIREMENTS

We need at least Company, Branch, Administrator, Employee and Customer class to represent our company. We will also need some additional classes to properly implement these classes like Order and Person. We need data structures to keep the data. Linked list used for Branches, array list used for users and Hybrid List used for products.

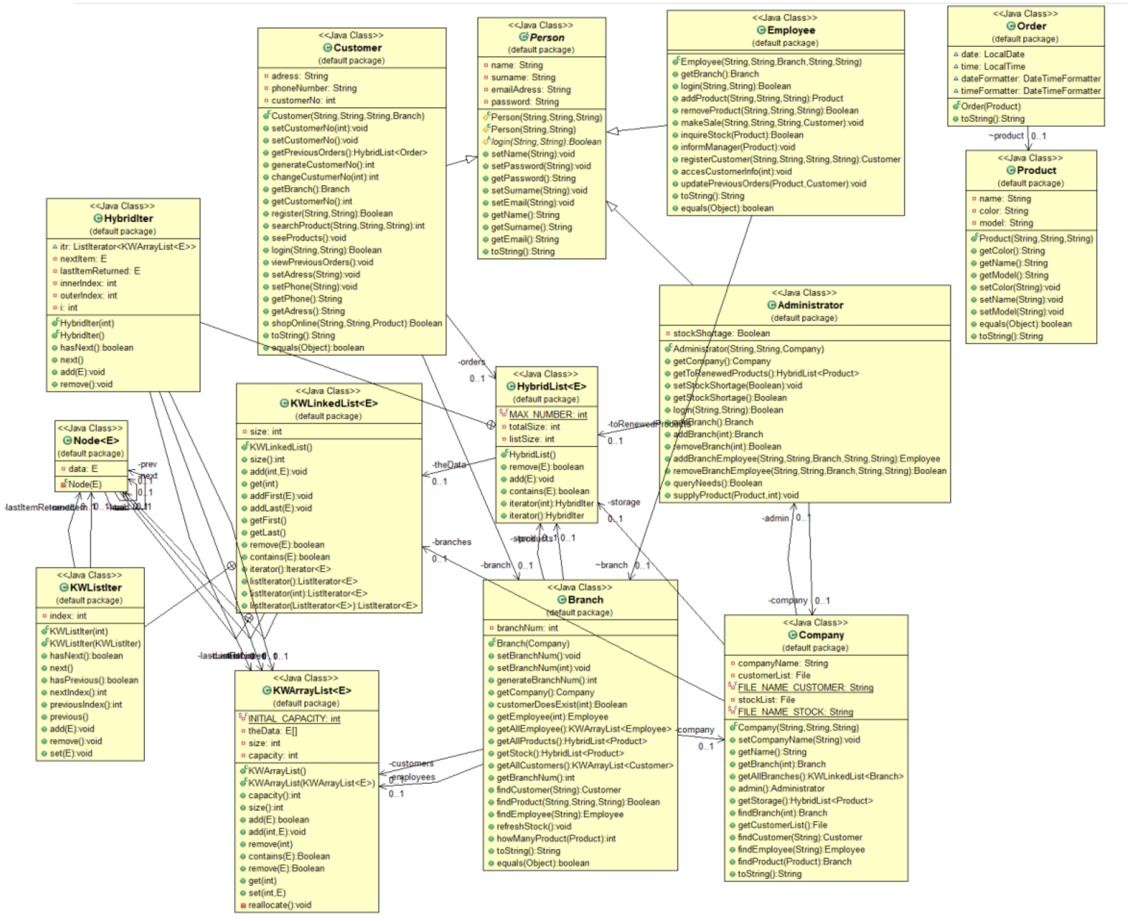
Furniture company has some features to find specific employees and customers. You can create a company with company name and admin information. All users have to login the system and customers must register if they don't have an account. Administrator is assigned by the system.

Administrators can delete branch/employee, add branch/employee, and query the needs and supply if something is out of the stock.

Customers can register, search for products, see the list of products, see which store a product is in, shop online by entering address and phone information, and view their previous orders.

Employees can inquire about the products in stock, inform the manager that the product should be purchased when any product is less than the requested amount, add / remove products, make sales, access the information of the previous orders of a customer by using the customer number and add new order to this section.

2. USE CASE AND CLASS DIAGRAMS



<<Java Class>> Customer (default package)
adress: String
phoneNumber: String
customerNo: int
Customer(String,String,String,Branch)
setCustomerNo(int):void
setCustomerNo():void
getPreviousOrders():HybridList<Order>
generateCustomerNo():int
changeCustumerNo(int):int
getBranch():Branch
getCustomerNo():int
register(String,String):Boolean
searchProduct(String,String,String):int
seeProducts():void
login(String,String):Boolean
viewPreviousOrders():void
setAdress(String):void
setPhone(String):void
getPhone():String
getAdress():String
shopOnline(String,String,Product):Boolean
toString():String
equals(Object):boolean
<<Java Class>> Person (default package)
name: String
surname: String
emailAdress: String
password: String
Person(String,String,String)
Person(String,String)
login(String,String):Boolean
setName(String):void
setPassword(String):void
getPassword():String
setSurname(String):void
setEmail(String):void
getName():String
getSurname():String
getEmail():String
toString():String
<<Java Class>> Employee (default package)
Employee(String,String,Branch,String,String)
getBranch():Branch
login(String,String):Boolean
addProduct(String,String,String):Product
removeProduct(String,String,String):Boolean
makeSale(String,String,String,Customer):void
inquireStock(Product):Boolean
informManager(Product):void
registerCustomer(String,String,String,String):Customer
accesCustomerInfo(int):void
updatePreviousOrders(Product,Customer):void
toString():String
equals(Object):boolean
<<Java Class>> Order (default package)
date: LocalDate
time: LocalTime
dateFormatter: DateTimeFormatter
timeFormatter: DateTimeFormatter
Order(Product)
toString():String
~product 0..1
<<Java Class>> Product (default package)
name: String
color: String
model: String
Product(String,String,String)
getColor():String
getName():String
getModel():String
setColor(String):void
setName(String):void
setModel(String):void
equals(Object):boolean
toString():String
<<Java Class>> HybridIter (default package)
itr: ListIterator<KWArrayList<E>>
nextItem: E
lastItemReturned: E
innerIndex: int
outerIndex: int
i: int
HybridIter(int)
HybridIter()
hasNext():boolean
next()
add(E):void
remove():void
<<Java Class>> Administrator (default package)
stockShortage: Boolean
Administrator(String,String,Company)
getCompany():Company
getToRenewedProducts():HybridList<Product>
setStockShortage(Boolean):void
getStockShortage():Boolean
login(String,String):Boolean
addBranch():Branch
addBranch(int):Branch
removeBranch(int):Boolean
addBranchEmployee(String,String,Branch,String,String):Employee
removeBranchEmployee(String,String,Branch,String,String):Boolean
queryNeeds():Boolean
supplyProduct(Product,int):void
<<Java Class>> Node<E> (default package)
data: E
Node(E)
-prev
-next
0..1
<<Java Class>> KWLinkedList<E> (default package)
size: int
KWLinkedList()
size():int
add(int,E):void
get(int)
addFirst(E):void
addLast(E):void
getFirst()
getLast()
remove(E):boolean
contains(E):boolean
iterator():Iterator<E>
listIterator():ListIterator<E>
listIterator(int):ListIterator<E>
listIterator(ListIterator<E>):ListIterator<E>
<<Java Class>> HybridList<E> (default package)
MAX_NUMBER: int
totalSize: int
listSize: int
HybridList()
remove(E):boolean
add(E):void
contains(E):boolean
iterator(int):HybridIter
iterator():HybridIter
-orders 0..1
-theData 0..1
-toRenewedProducts 0..1
-storage 0..1
-branches 0..1
-products 0..1
-branch 0..1
~branch 0..1
-admin 0..1
-company 0..1
<<Java Class>> KWListIter (default package)
index: int
KWListIter(int)
KWListIter(KWListIter)
hasNext():boolean
next()
hasPrevious():boolean
nextIndex():int
previousIndex():int
previous()
add(E):void
remove():void
set(E):void
-lastItemReturned
<<Java Class>> Branch (default package)
branchNum: int
Branch(Company)
setBranchNum():void
setBranchNum(int):void
generateBranchNum():int
getCompany():Company
customerDoesExist(int):Boolean
getEmployee(int):Employee
getAllEmployee():KWArrayList<Employee>
getAllProducts():HybridList<Product>
getStock():HybridList<Product>
getAllCustomers():KWArrayList<Customer>
getBranchNum():int
findCustomer(String):Customer
findProduct(String,String,String):Boolean
findEmployee(String):Employee
refreshStock():void
howManyProduct(Product):int
toString():String
equals(Object):boolean
-company 0..1
<<Java Class>> KWArrayList<E> (default package)
INITIAL_CAPACITY: int
theData: E[]
size: int
capacity: int
KWArrayList()
KWArrayList(KWArrayList<E>)
capacity():int
size():int
add(E):boolean
add(int,E):void
remove(int)
contains(E):Boolean
remove(E):Boolean
get(int)
set(int,E)
reallocate():void
-customers
-employees
0..1
<<Java Class>> Company (default package)
companyName: String
customerList: File
FILE_NAME_CUSTOMER: String
stockList: File
FILE_NAME_STOCK: String
Company(String,String,String)
setCompanyName(String):void
getName():String
getBranch(int):Branch
getAllBranches():KWLinkedList<Branch>
admin():Administrator
getStorage():HybridList<Product>
findBranch(int):Branch
getCustomerList():File
findCustomer(String):Customer
findEmployee(String):Employee
findProduct(Product):Branch
toString():String

3. PROBLEM SOLUTION APPROACH

I have used generic Linked List for branches. Linked list has an iterator to make it run more efficiently. It has add, remove, contain, get etc. methods to properly represent branches.

I have used ArrayList for users. It also has the same methods like linked list but I didn't implement an iterator because it is not necessary.

I have used Hybrid List for the product. Hybrid List is linked list with an Array List node. It is more efficient for products. Because when a product is sold, I must remove it from the product array. Shifting the whole array takes a lot more time than shifting just some part of it. All Array Lists have MAX_NUMBER element in them and if I remove one of the elements, only MAX_NUMBER of elements is shifting rather than product size.

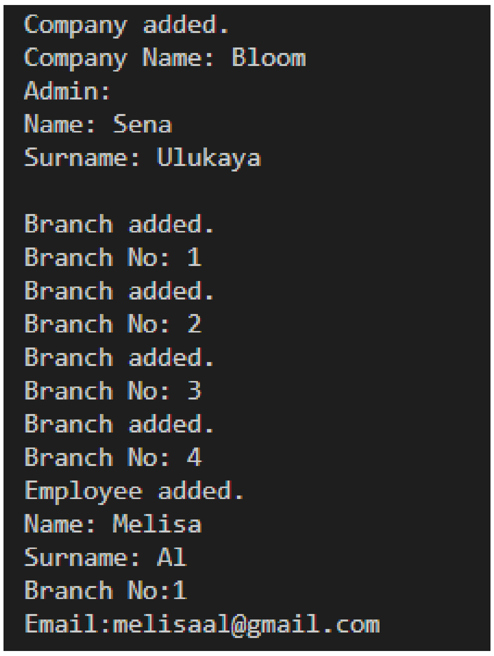
Hybrid List has an iterator and I have implemented all my other methods -like add, remove, contains with the help of the iterator.

Data structures helped me to implement all my classes according to my system requirements.

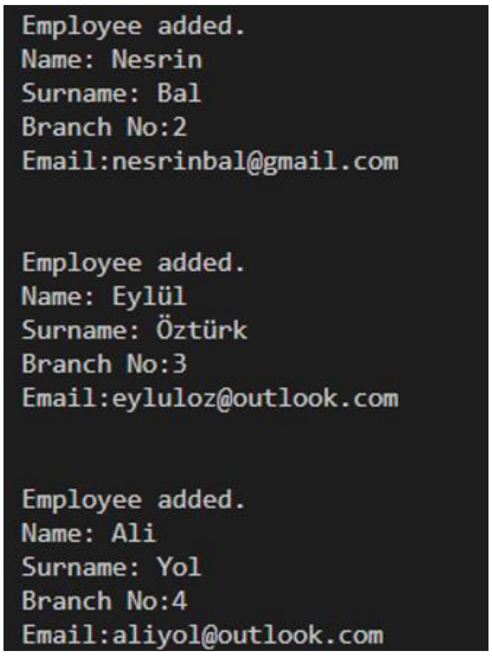
4. TEST CASES

I created a driver code and I have tested all my methods with possible cases there. Also, I have created a menu with a Main Menu, Customer Menu, Employee Menu, Admin Menu. You can test all the methods with every case there. When you run the code, it will ask if you want to try yourself or want me to try it for you. According to that driver or menu will run.

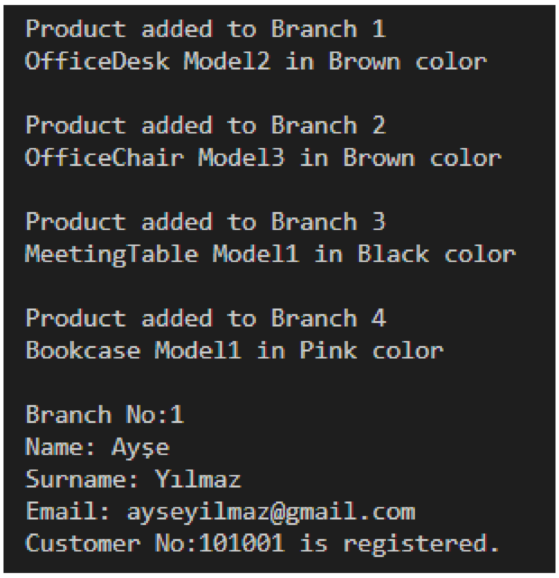
I will add test cases for and running results of menu at part 5. Password is always "12345" for test and I have added a couple of employee, branch, customer, and product to test the menu. You can use these in the menu.



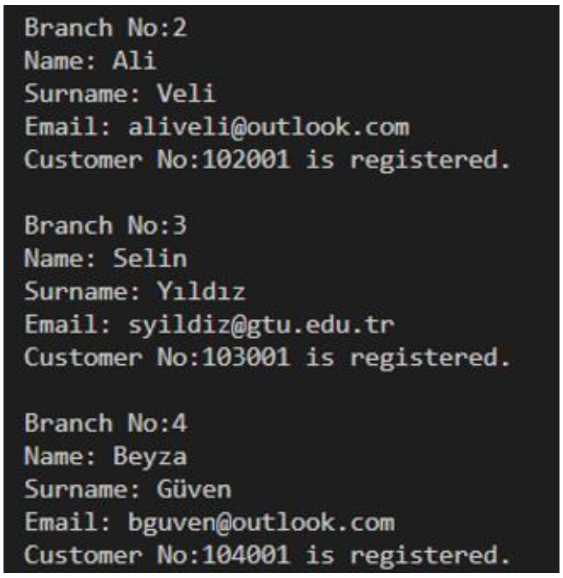
```
Company added.
Company Name: Bloom
Admin:
Name: Sena
Surname: Ulukaya

Branch added.
Branch No: 1
Branch added.
Branch No: 2
Branch added.
Branch No: 3
Branch added.
Branch No: 4
Employee added.
Name: Melisa
Surname: Al
Branch No:1
Email:melisaal@gmail.com
```



```
Employee added.
Name: Nesrin
Surname: Bal
Branch No:2
Email:nesrinbal@gmail.com


Employee added.
Name: Eylül
Surname: Öztürk
Branch No:3
Email:eyluloz@outlook.com


Employee added.
Name: Ali
Surname: Yol
Branch No:4
Email:aliyol@outlook.com
```



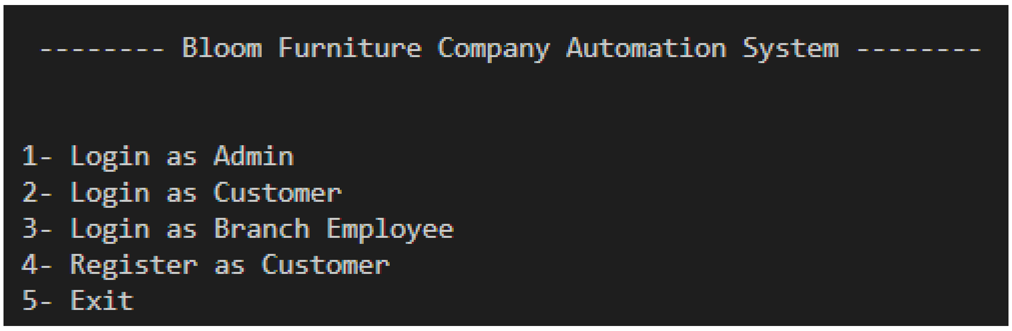
```
Product added to Branch 1
OfficeDesk Model2 in Brown color

Product added to Branch 2
OfficeChair Model3 in Brown color

Product added to Branch 3
MeetingTable Model1 in Black color

Product added to Branch 4
Bookcase Model1 in Pink color

Branch No:1
Name: Ayşe
Surname: Yılmaz
Email: ayseyilmaz@gmail.com
Customer No:101001 is registered.
```



```
Branch No:2
Name: Ali
Surname: Veli
Email: aliveli@outlook.com
Customer No:102001 is registered.

Branch No:3
Name: Selin
Surname: Yıldız
Email: syildiz@gtu.edu.tr
Customer No:103001 is registered.

Branch No:4
Name: Beyza
Surname: Güven
Email: bguven@outlook.com
Customer No:104001 is registered.
```

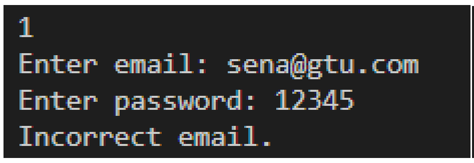
5. RUNNING AND RESULTS

Main Menu:

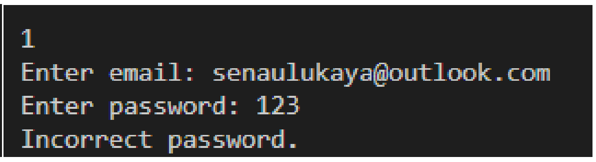


```
 -------- Bloom Furniture Company Automation System --------


1- Login as Admin
2- Login as Customer
3- Login as Branch Employee
4- Register as Customer
5- Exit
```

1- **Admin Menu:** You should login with correct email(senaulukaya@outlook.com) and password (12345). If e-mail or password is wrong:

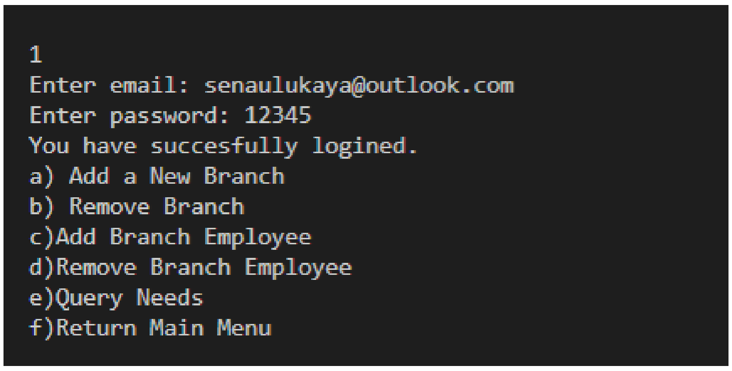


```
1
Enter email: sena@gtu.com
Enter password: 12345
Incorrect email.
```



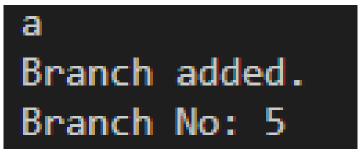
```
1
Enter email: senaulukaya@outlook.com
Enter password: 123
Incorrect password.
```

If you enter the right email and the password it will login and new options will dissapear.



```
1
Enter email: senaulukaya@outlook.com
Enter password: 12345
You have succesfully logined.
a) Add a New Branch
b) Remove Branch
c)Add Branch Employee
d)Remove Branch Employee
e)Query Needs
f)Return Main Menu
```

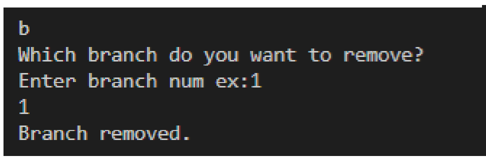
a) **Add a New Branch:** 5th branch is added.



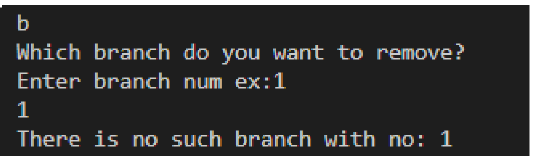
```
a
Branch added.
Branch No: 5
```

b) Remove Branch: Branch will removed if it is exist, else it will display error. (Branches with no 1-2-3-4-5 exist.)

*I tried to remove branch 1 twice, but It didn't let me in the second time.



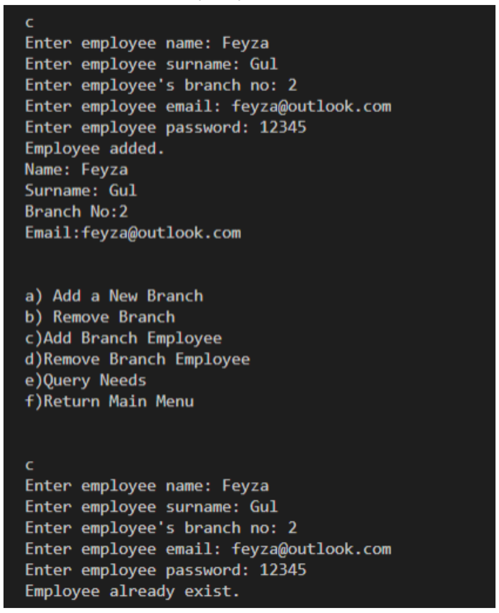
```
b
Which branch do you want to remove?
Enter branch num ex:1
1
Branch removed.
```



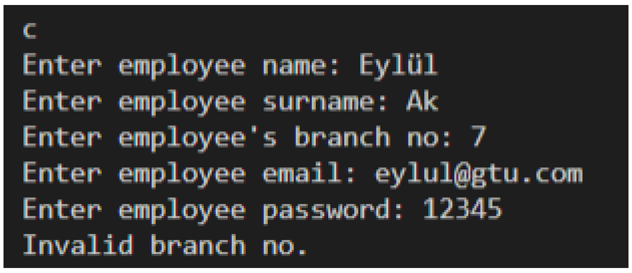
```
b
Which branch do you want to remove?
Enter branch num ex:1
1
There is no such branch with no: 1
```

c) Add Branch Employee: Employee will be added, if e-mail is unique. Else it will display error.

*I tried to add same person twice, first one is succesful but second time is not.
* I tried to add a employee with invalid branch no.



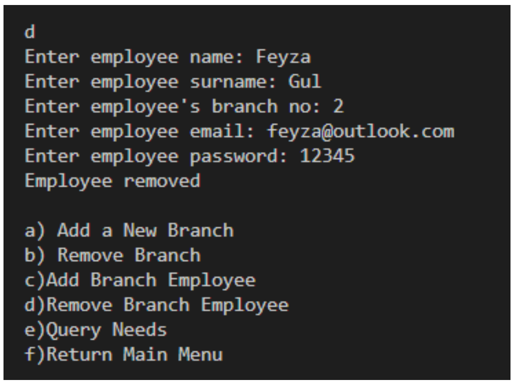
```
c
Enter employee name: Feyza
Enter employee surname: Gul
Enter employee's branch no: 2
Enter employee email: feyza@outlook.com
Enter employee password: 12345
Employee added.
Name: Feyza
Surname: Gul
Branch No:2
Email:feyza@outlook.com


a) Add a New Branch
b) Remove Branch
c)Add Branch Employee
d)Remove Branch Employee
e)Query Needs
f)Return Main Menu


c
Enter employee name: Feyza
Enter employee surname: Gul
Enter employee's branch no: 2
Enter employee email: feyza@outlook.com
Enter employee password: 12345
Employee already exist.
```



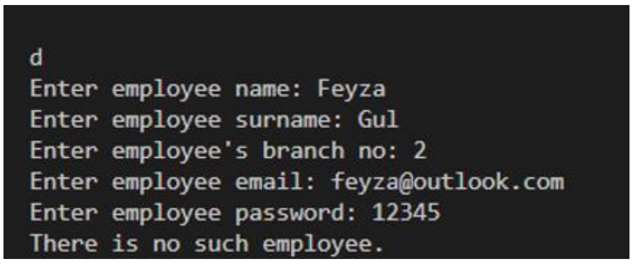
```
c
Enter employee name: Eylül
Enter employee surname: Ak
Enter employee's branch no: 7
Enter employee email: eylul@gtu.com
Enter employee password: 12345
Invalid branch no.
```

d) Remove Branch Employee: Employee will removed if it is exist, else it will display error.

*I tried to remove same employee twice, and second time it gave error.



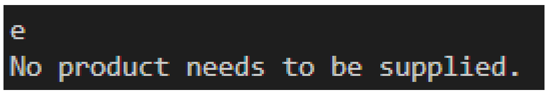
```
d
Enter employee name: Feyza
Enter employee surname: Gul
Enter employee's branch no: 2
Enter employee email: feyza@outlook.com
Enter employee password: 12345
Employee removed

a) Add a New Branch
b) Remove Branch
c)Add Branch Employee
d)Remove Branch Employee
e)Query Needs
f)Return Main Menu
```



```
d
Enter employee name: Feyza
Enter employee surname: Gul
Enter employee's branch no: 2
Enter employee email: feyza@outlook.com
Enter employee password: 12345
There is no such employee.
```

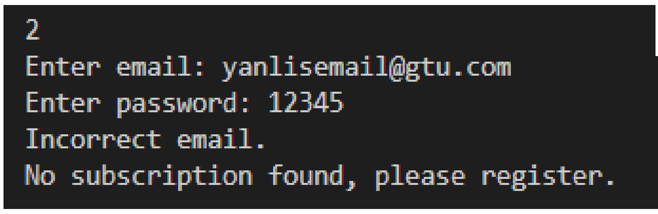
e) **Query Needs:** Display the products that needs to be supplied if any employee informed the admin before.

*Admin is not informed, so no products need to be supplied. I will use this method later, after employee operations.

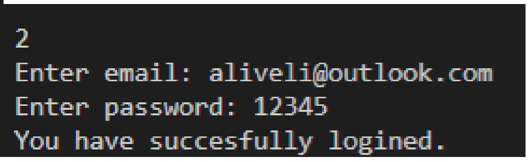


```
e
No product needs to be supplied.
```

2- **Customer Menu:** You should login with already existed subscription. If you won't system will ask you to register.

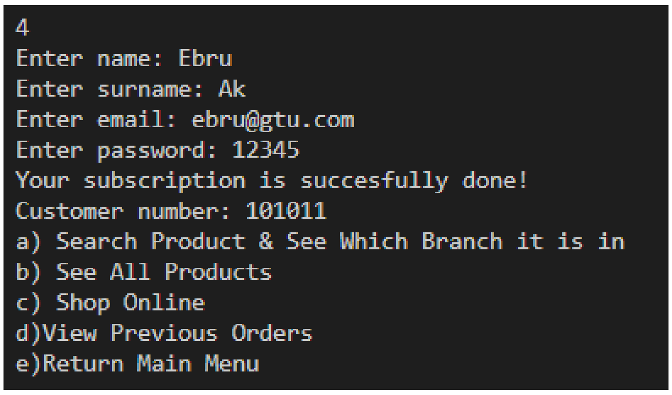


```
2
Enter email: yanlisemail@gtu.com
Enter password: 12345
Incorrect email.
No subscription found, please register.
```



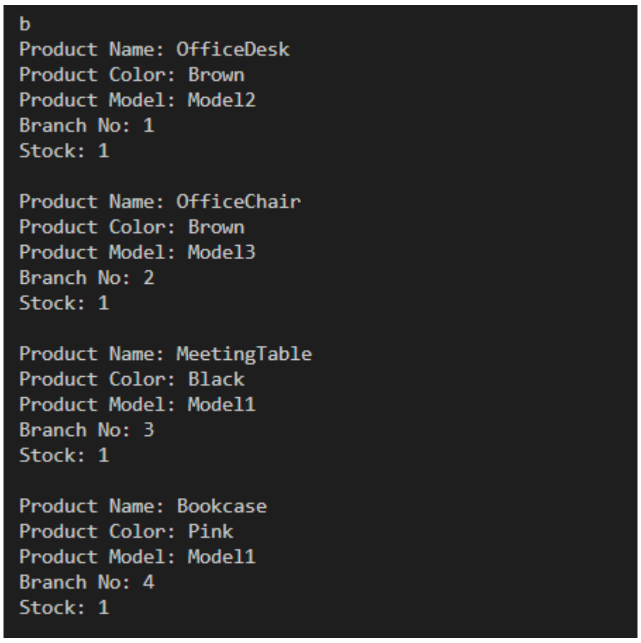
```
2
Enter email: aliveli@outlook.com
Enter password: 12345
You have succesfully logined.
```

*Now I will register a new account. (Option 4 in main menu).



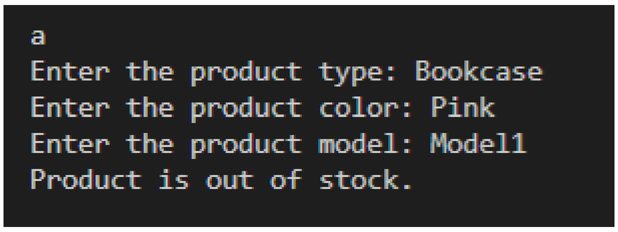
```
4
Enter name: Ebru
Enter surname: Ak
Enter email: ebru@gtu.com
Enter password: 12345
Your subscription is succesfully done!
Customer number: 101011
a) Search Product & See Which Branch it is in
b) See All Products
c) Shop Online
d)View Previous Orders
e)Return Main Menu
```

b) **See All Products:** List of all the available products with their stock number.



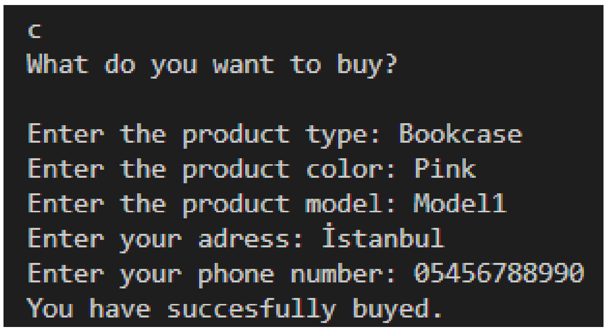
```
b
Product Name: OfficeDesk
Product Color: Brown
Product Model: Model2
Branch No: 1
Stock: 1

Product Name: OfficeChair
Product Color: Brown
Product Model: Model3
Branch No: 2
Stock: 1

Product Name: MeetingTable
Product Color: Black
Product Model: Model1
Branch No: 3
Stock: 1

Product Name: Bookcase
Product Color: Pink
Product Model: Model1
Branch No: 4
Stock: 1
```

a) **Search Product & See Which Branch it is in:** If product is not exist/out of stock it will give error.

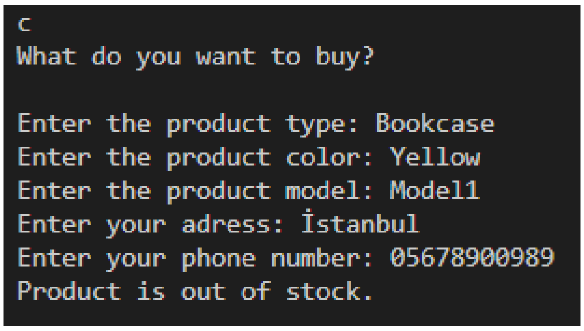


```
a
Enter the product type: Bookcase
Enter the product color: Pink
Enter the product model: Model1
Product is out of stock.
```

c) **Online Shop:** Customer can buy any product from any branch if it is exist. They have to enter their email and phone number. They will get error if product is out of stock.

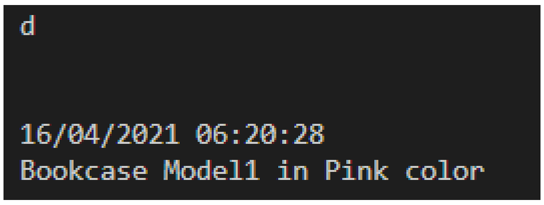


```
c
What do you want to buy?

Enter the product type: Bookcase
Enter the product color: Pink
Enter the product model: Model1
Enter your adress: İstanbul
Enter your phone number: 05456788990
You have succesfully buyed.
```



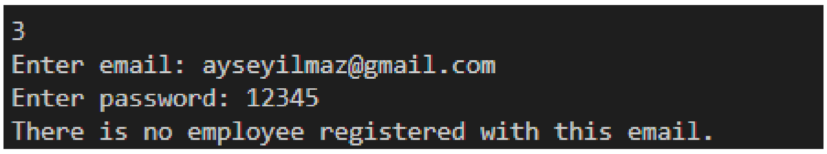
```
c
What do you want to buy?

Enter the product type: Bookcase
Enter the product color: Yellow
Enter the product model: Model1
Enter your adress: İstanbul
Enter your phone number: 05678900989
Product is out of stock.
```

d) **See previous orders:** Customers can display their previous orders with the time and date. The product that I bought is in the list.

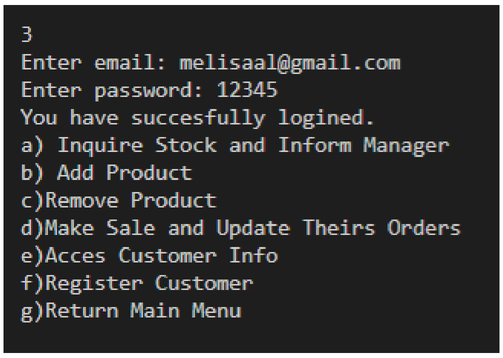


```
d


16/04/2021 06:20:28
Bookcase Model1 in Pink color
```

3- **Employee Menu:** You should login with a correct email and password. If you won't system will give error.

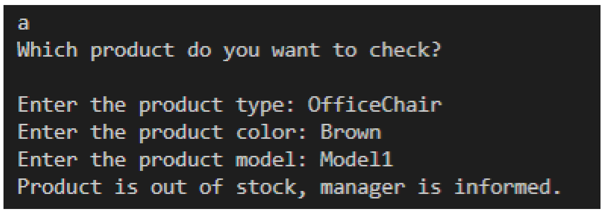


```
3
Enter email: ayseyilmaz@gmail.com
Enter password: 12345
There is no employee registered with this email.
```



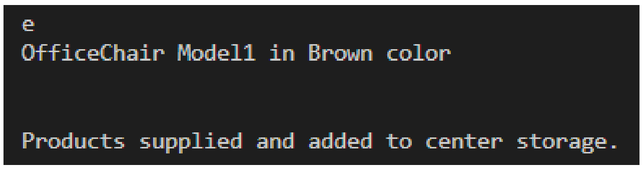
```
3
Enter email: melisaal@gmail.com
Enter password: 12345
You have succesfully logined.
a) Inquire Stock and Inform Manager
b) Add Product
c)Remove Product
d)Make Sale and Update Theirs Orders
e)Acces Customer Info
f)Register Customer
g)Return Main Menu
```

a) Inquire Stock and Inform Manager



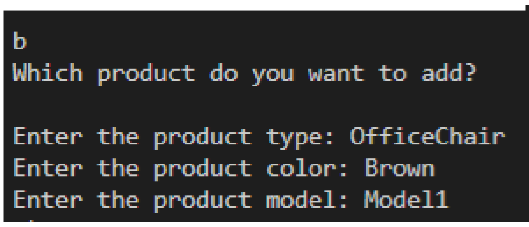
```
a
Which product do you want to check?

Enter the product type: OfficeChair
Enter the product color: Brown
Enter the product model: Model1
Product is out of stock, manager is informed.
```

*I am adding the Query Needs output for admin now. Since admin is informed, it can see OfficeChair is out of stock and supplied it.

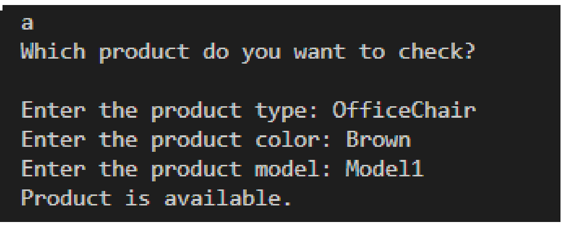


```
e
OfficeChair Model1 in Brown color


Products supplied and added to center storage.
```

b) Add Product



```
b
Which product do you want to add?

Enter the product type: OfficeChair
Enter the product color: Brown
Enter the product model: Model1
```

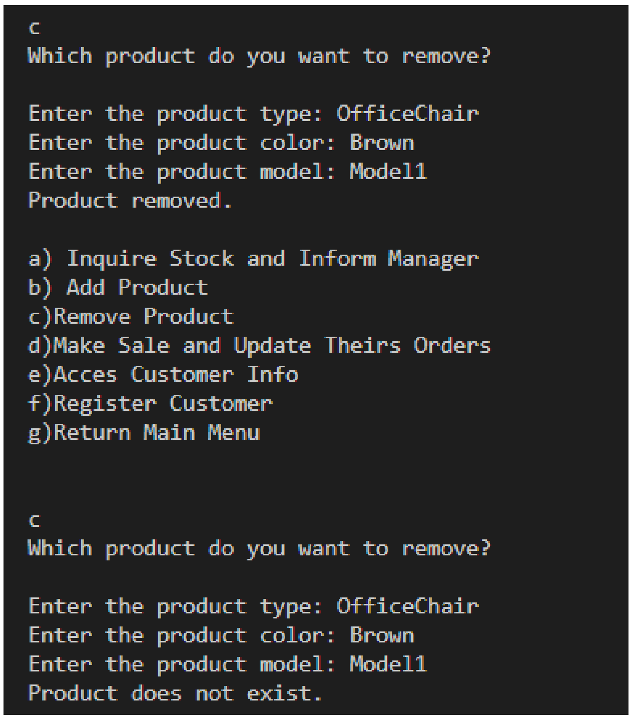


```
a
Which product do you want to check?

Enter the product type: OfficeChair
Enter the product color: Brown
Enter the product model: Model1
Product is available.
```

*I will inquire stock again, after adding the product.

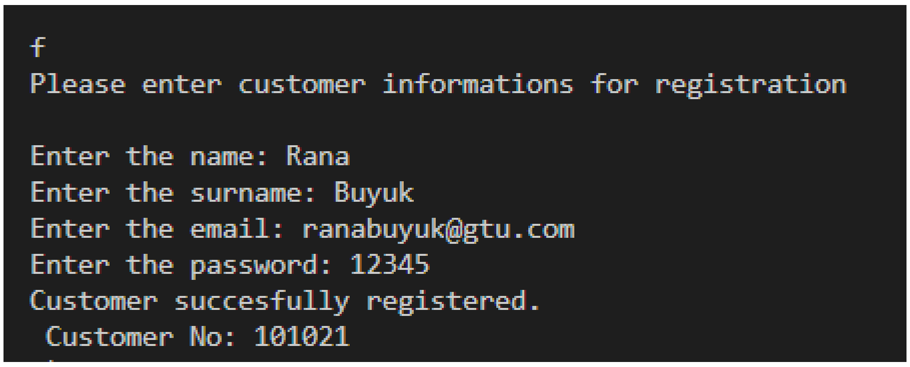
c) Remove Product: If product exist in the branch, it will removed.

*I tried to remove the product that I add twice. It gave error second time**.**



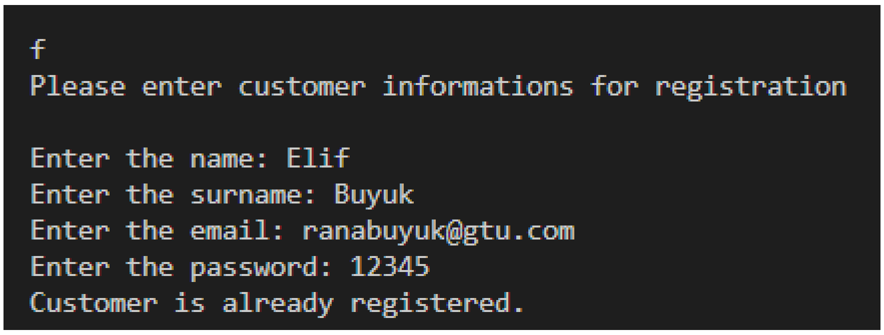
```
c
Which product do you want to remove?

Enter the product type: OfficeChair
Enter the product color: Brown
Enter the product model: Model1
Product removed.

a) Inquire Stock and Inform Manager
b) Add Product
c)Remove Product
d)Make Sale and Update Theirs Orders
e)Acces Customer Info
f)Register Customer
g)Return Main Menu


c
Which product do you want to remove?

Enter the product type: OfficeChair
Enter the product color: Brown
Enter the product model: Model1
Product does not exist.
```

f)Register Customer: Add customer, if it is not already have an account.



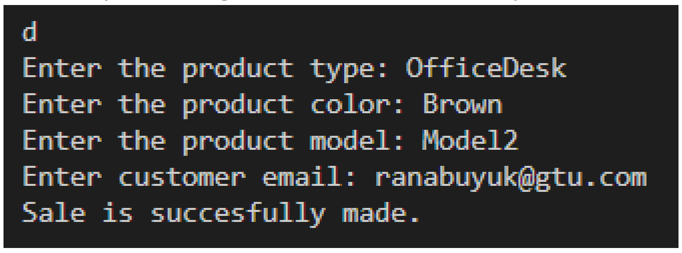
```
f
Please enter customer informations for registration

Enter the name: Rana
Enter the surname: Buyuk
Enter the email: ranabuyuk@gtu.com
Enter the password: 12345
Customer succesfully registered.
 Customer No: 101021
```

*Email has to be unique for each customer.



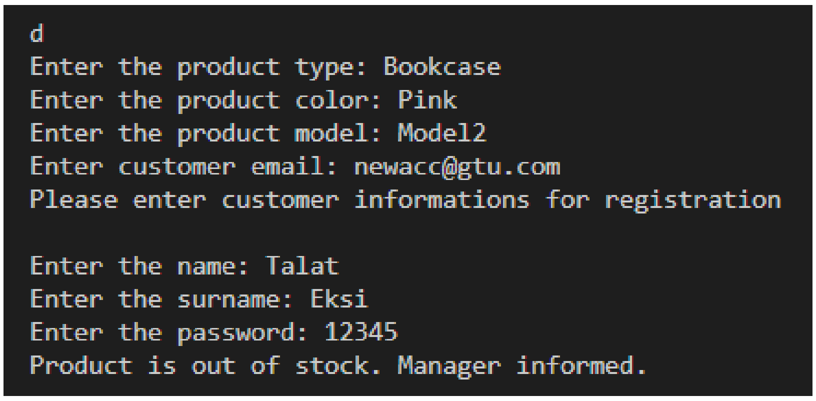
```
f
Please enter customer informations for registration

Enter the name: Elif
Enter the surname: Buyuk
Enter the email: ranabuyuk@gtu.com
Enter the password: 12345
Customer is already registered.
```

d) Make Sale and Update Customer Orders: If customer email is not registered, system will ask you to register the customer. If product is exist, sale is succesful else it will give error.



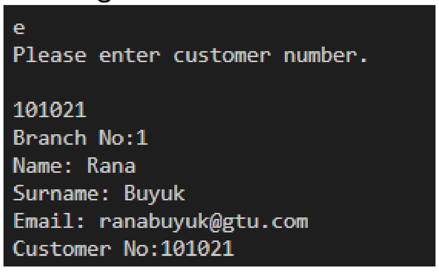
```
d
Enter the product type: OfficeDesk
Enter the product color: Brown
Enter the product model: Model2
Enter customer email: ranabuyuk@gtu.com
Sale is succesfully made.
```

*Both email is not registered and product is out of stock cases.

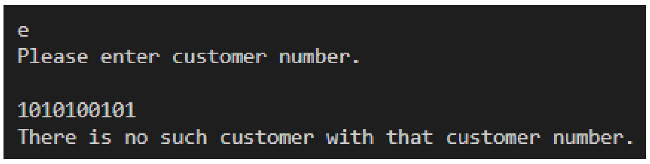


```
d
Enter the product type: Bookcase
Enter the product color: Pink
Enter the product model: Model2
Enter customer email: newacc@gtu.com
Please enter customer informations for registration

Enter the name: Talat
Enter the surname: Eksi
Enter the password: 12345
Product is out of stock. Manager informed.
```

e) Acces Customer Info: If customer no exist, you can Access customer info, if it is not it will give error.



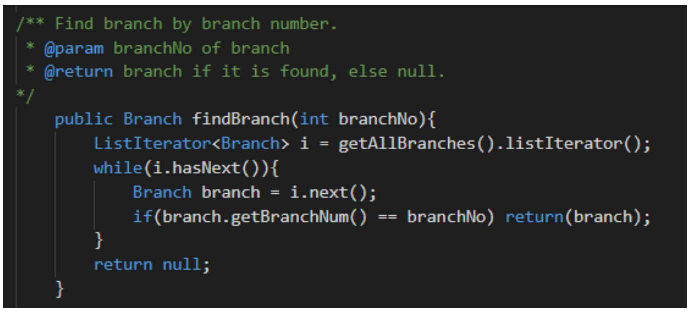
```
e
Please enter customer number.

101021
Branch No:1
Name: Rana
Surname: Buyuk
Email: ranabuyuk@gtu.com
Customer No:101021
```



```
e
Please enter customer number.

1010100101
There is no such customer with that customer number.
```

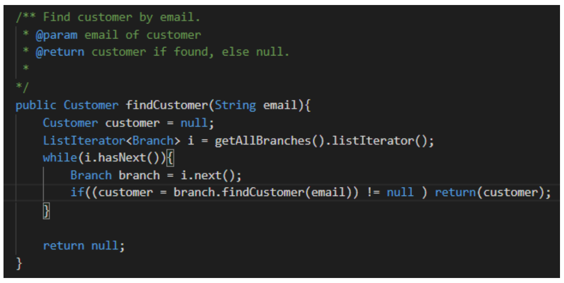
PART 2: Time Complexity Analysis

1- Company Class

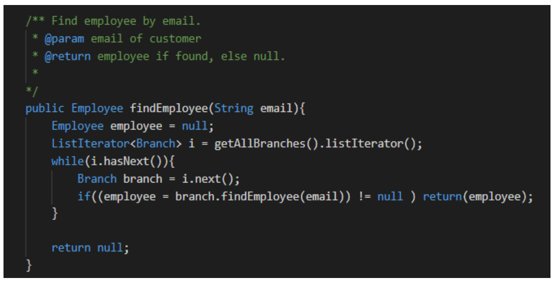


```java
/** Find branch by branch number.
 * @param branchNo of branch
 * @return branch if it is found, else null.
*/
    public Branch findBranch(int branchNo){
        ListIterator<Branch> i = getAllBranches().listIterator();
        while(i.hasNext()){
            Branch branch = i.next();
            if(branch.getBranchNum() == branchNo) return(branch);
        }
        return null;
    }
```

- In best case, branch is in the first node, so loop will break after 1 iteration. Creating iterator from begining index is also constant, like hasNext and next.
  $T_b(n) = \theta(1)$.
- In worst case, branch does not exist, it will run until the last node of the branch linked list.
  $T_w(n) = \theta(n)$.
- In general we can say, **$T(n) = O(n)$.**

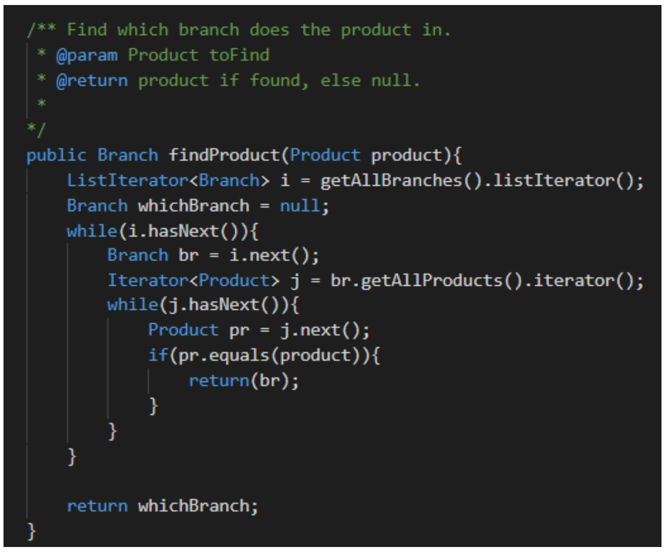


```java
/** Find customer by email.
 * @param email of customer
 * @return customer if found, else null.
 *
*/
public Customer findCustomer(String email){
    Customer customer = null;
    ListIterator<Branch> i = getAllBranches().listIterator();
    while(i.hasNext()){
        Branch branch = i.next();
        if((customer = branch.findCustomer(email)) != null ) return(customer);
    }

    return null;
}
```



```java
/** Find employee by email.
 * @param email of customer
 * @return employee if found, else null.
 *
*/
public Employee findEmployee(String email){
    Employee employee = null;
    ListIterator<Branch> i = getAllBranches().listIterator();
    while(i.hasNext()){
        Branch branch = i.next();
        if((employee = branch.findEmployee(email)) != null ) return(employee);
    }

    return null;
}
```

- In best case, customer/employee is in the first index of the first branch, so loop will break after 1 iteration. In if condition branch is cheking if customer/employee is in it with the same approach. Creating iterator from begining index is also constant, like hasNext and next.
  $T_b(n) = \theta(1)$.
- In worst case, customer/employee does not exist, it will run until the end of the every branch's customer list. If condition is **$\theta(n)$,** and outer loop is also **$\theta(n)$.**
  $T_w(n) = \theta(n*n) = \theta(n^2)$.
- In general we can say, **$T(n) = O(n^2)$.**

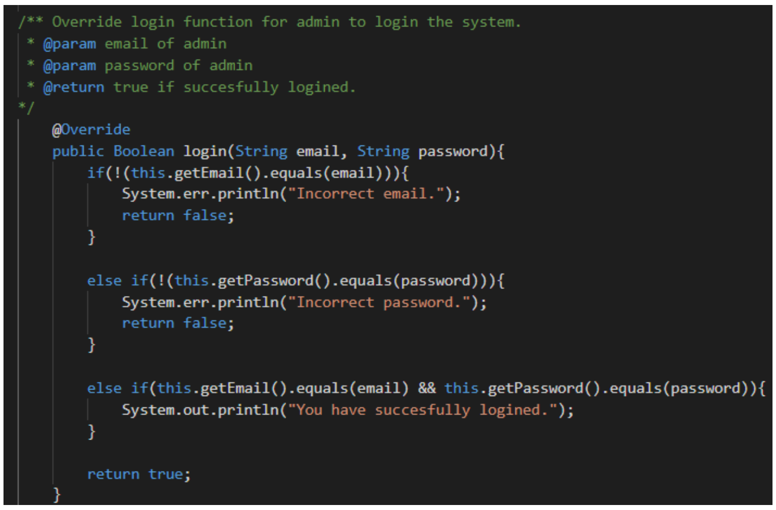


```java
/** Find which branch does the product in.
 * @param Product toFind
 * @return product if found, else null.
 *
*/
public Branch findProduct(Product product){
    ListIterator<Branch> i = getAllBranches().listIterator();
    Branch whichBranch = null;
    while(i.hasNext()){
        Branch br = i.next();
        Iterator<Product> j = br.getAllProducts().iterator();
        while(j.hasNext()){
            Product pr = j.next();
            if(pr.equals(product)){
                return(br);
            }
        }
    }

    return whichBranch;
}
```

n is number of branches and m is product in the branches.

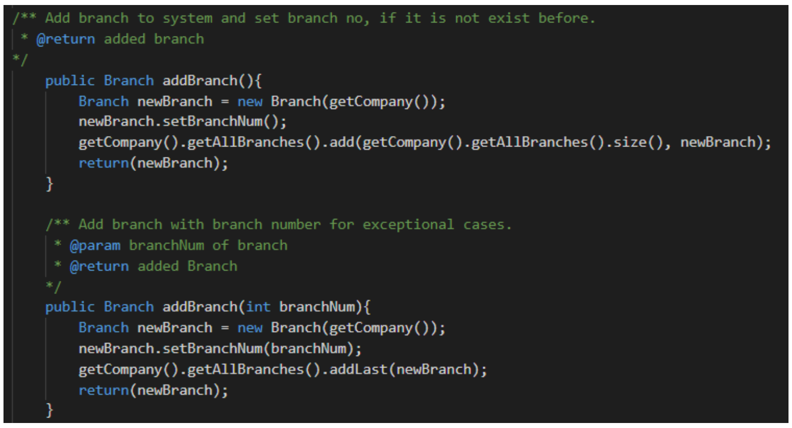
- In best case, product is in the first index of the first branch, so loop will break after 1 iteration. Creating both of iterators from begining index is also constant, like hasNext and next.
  T_b(n,m) = θ(1).
- In worst case, product does not exist, it will run until the end of the every branch's product hybridlist. Inner loop is **θ(m),** and outer loop is **θ(n).**
  T_w(n,m) = θ(n*m) = θ(n*m).
- In general we can say, **T(n) = O(n*m).**

2- Administrator Class

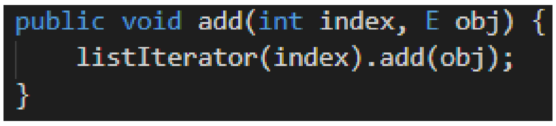


```java
/** Override login function for admin to login the system.
 * @param email of admin
 * @param password of admin
 * @return true if succesfully logined.
*/
    @Override
    public Boolean login(String email, String password){
        if(!(this.getEmail().equals(email))){
            System.err.println("Incorrect email.");
            return false;
        }

        else if(!(this.getPassword().equals(password))){
            System.err.println("Incorrect password.");
            return false;
        }

        else if(this.getEmail().equals(email) && this.getPassword().equals(password)){
            System.out.println("You have succesfully logined.");
        }

        return true;
    }
```

- String equals method is constant only when the strings are the same object, or object is not instance of string, or the string lengths are different. Otherwise, it has to compare two strings until the end of the string lenght if each char is same but the last char.
- In best case, there's a chance that all the if conditions are the cases that I mentioned above.
  T_b(n) = θ(1).
- In worst case, It has to compare and that is **O(n).** (first char can be different best case, or last char worst case).

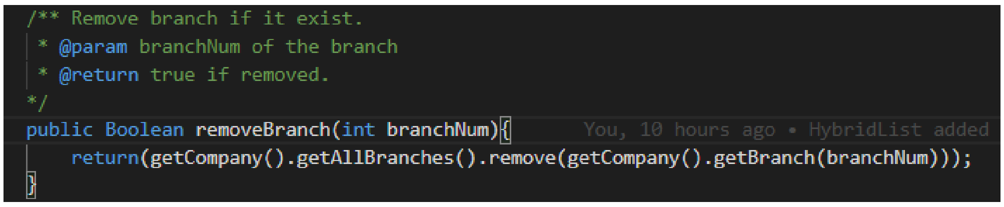


```java
/** Add branch to system and set branch no, if it is not exist before.
 * @return added branch
*/
    public Branch addBranch(){
        Branch newBranch = new Branch(getCompany());
        newBranch.setBranchNum();
        getCompany().getAllBranches().add(getCompany().getAllBranches().size(), newBranch);
        return(newBranch);
    }

    /** Add branch with branch number for exceptional cases.
     * @param branchNum of branch
     * @return added Branch
    */
    public Branch addBranch(int branchNum){
        Branch newBranch = new Branch(getCompany());
        newBranch.setBranchNum(branchNum);
        getCompany().getAllBranches().addLast(newBranch);
        return(newBranch);
    }
```



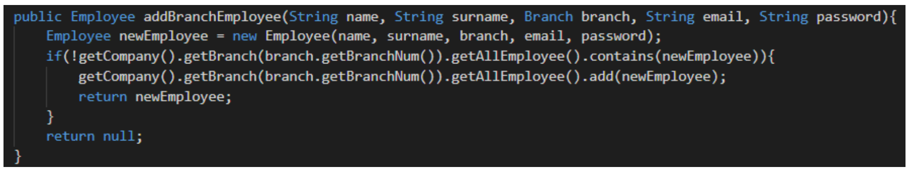
```java
public void add(int index, E obj) {
    listIterator(index).add(obj);
}
```

- setBranchNum is O(n), because it's generating branch num. Adding at the end of the linked list with listIterator is constant time, exceptional case for the end of the linked list. Adding with the branch num is contanst time because it is not generating num and adding to the last.
  $T_1(n)$ = O(n).
  $T_2(n)$ = θ(1).

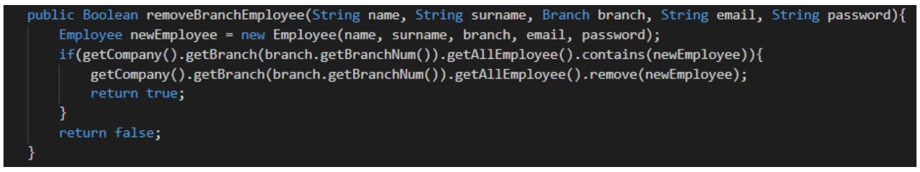


```java
/** Remove branch if it exist.
 * @param branchNum of the branch
 * @return true if removed.
*/
public Boolean removeBranch(int branchNum){        You, 10 hours ago • HybridList added
    return(getCompany().getAllBranches().remove(getCompany().getBranch(branchNum)));
}
```

- If branch is the first index of the linked list removing is constant.
  $T_b(n)$ = θ(1).
- If branch does not exist,it have to loop until the end.
  $T_w(n)$ = θ(n).
- **T(n) = O(n).**

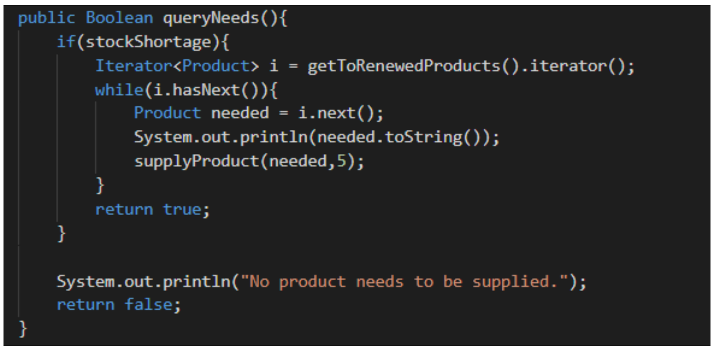


```java
public Employee addBranchEmployee(String name, String surname, Branch branch, String email, String password){
    Employee newEmployee = new Employee(name, surname, branch, email, password);
    if(!getCompany().getBranch(branch.getBranchNum()).getAllEmployee().contains(newEmployee)){
        getCompany().getBranch(branch.getBranchNum()).getAllEmployee().add(newEmployee);
        return newEmployee;
    }
    return null;
}
```



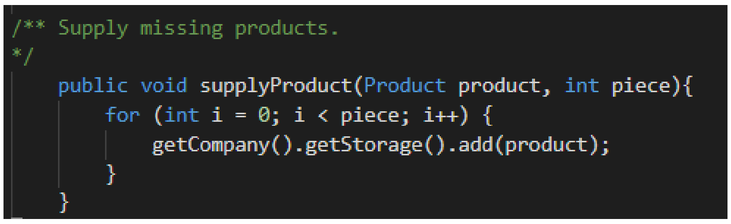
```java
public Boolean removeBranchEmployee(String name, String surname, Branch branch, String email, String password){
    Employee newEmployee = new Employee(name, surname, branch, email, password);
    if(getCompany().getBranch(branch.getBranchNum()).getAllEmployee().contains(newEmployee)){
        getCompany().getBranch(branch.getBranchNum()).getAllEmployee().remove(newEmployee);
        return true;
    }
    return false;
}
```

- If condition is checking if element contains in the array list. In best case it is constant time and worst case it does not contain. Inside of the body, adding at the end is amortized constant time. Removing can be constant if it is end of the list, but in other cases it requires shifting.
  Add: $T_b(n)$ = θ(1), $T_w(n)$ = θ(n). T(n) = O(n).
  Remove: $T_b(n)$ = θ(1), $T_w(n)$ = θ(n). T(n) = O(n).



```
public Boolean queryNeeds(){
    if(stockShortage){
        Iterator<Product> i = getToRenewedProducts().iterator();
        while(i.hasNext()){
            Product needed = i.next();
            System.out.println(needed.toString());
            supplyProduct(needed,5);
        }
        return true;
    }

    System.out.println("No product needs to be supplied.");
    return false;
}
```

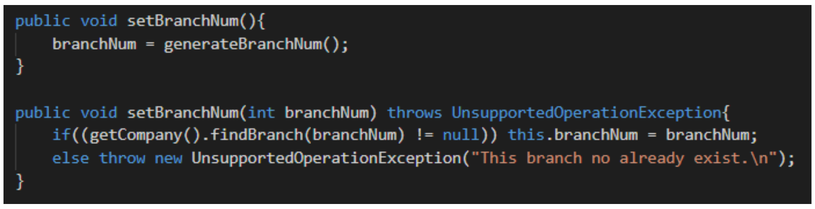
- Iterator will run until the end of the hybrid list of the out-of-stock-products. Supplying product is constant because 5 time is constant.
  T(n) = θ(n).



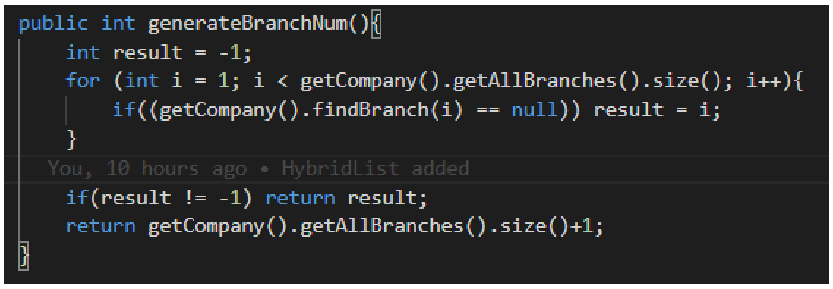
```
/** Supply missing products.
*/
    public void supplyProduct(Product product, int piece){
        for (int i = 0; i < piece; i++) {
            getCompany().getStorage().add(product);
        }
    }
```

- Adding at the end of the hybridlist is constant time. Loop will run piece times no matter what.
  T(n) = θ(n).

3- Branch Class

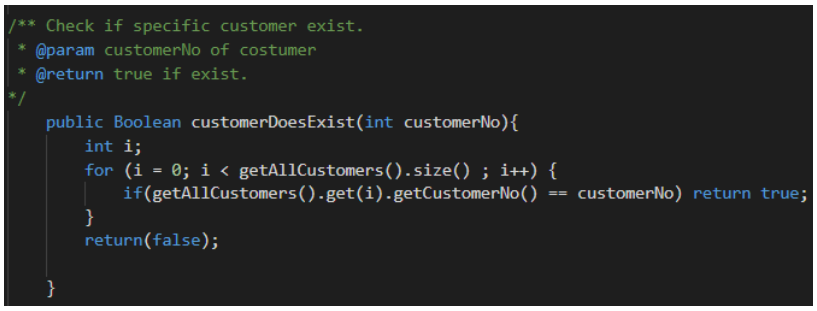


```
public void setBranchNum(){
    branchNum = generateBranchNum();
}

public void setBranchNum(int branchNum) throws UnsupportedOperationException{
    if((getCompany().findBranch(branchNum) != null)) this.branchNum = branchNum;
    else throw new UnsupportedOperationException("This branch no already exist.\n");
}
```



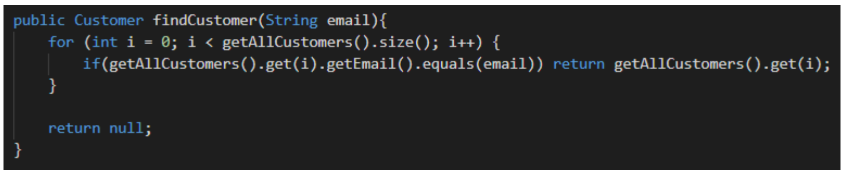
```
public int generateBranchNum(){
    int result = -1;
    for (int i = 1; i < getCompany().getAllBranches().size(); i++){
        if((getCompany().findBranch(i) == null)) result = i;
    }
  You, 10 hours ago • HybridList added
    if(result != -1) return result;
    return getCompany().getAllBranches().size()+1;
}
```

- Generatebranchnum’s if condition is generally O(n) as I analyzed above. Also, it is running n times.
  T(n) = O(n^2).
- Because of that, setBranchNum() is also **T(n) = O(n^2).,**
- setBranchNum(int) is **T(n) = O(n).**

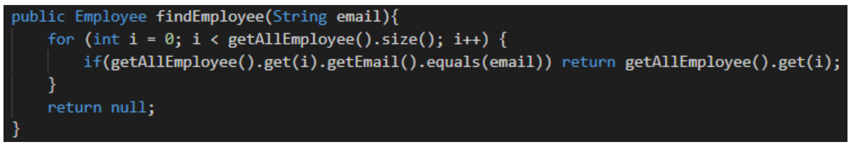


```java
/** Check if specific customer exist.
 * @param customerNo of costumer
 * @return true if exist.
*/
    public Boolean customerDoesExist(int customerNo){
        int i;
        for (i = 0; i < getAllCustomers().size() ; i++) {
            if(getAllCustomers().get(i).getCustomerNo() == customerNo) return true;
        }
        return(false);

    }
```

- In best case it is constant time. In worst case customer does not exist.
  $T_b(n) = \theta(1)$, $T_w(n) = \theta(n)$. $T(n) = O(n)$.

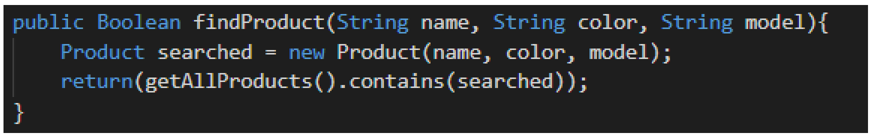


```java
public Customer findCustomer(String email){
    for (int i = 0; i < getAllCustomers().size(); i++) {
        if(getAllCustomers().get(i).getEmail().equals(email)) return getAllCustomers().get(i);
    }

    return null;
}
```



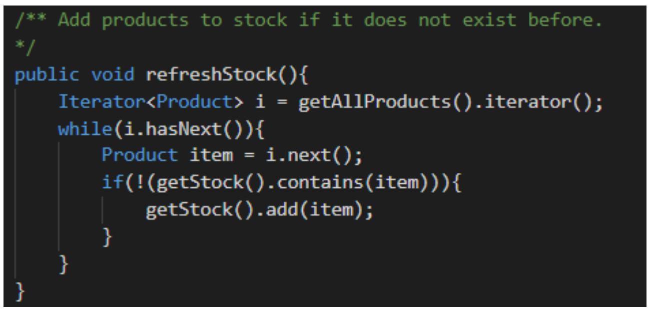
```java
public Employee findEmployee(String email){
    for (int i = 0; i < getAllEmployee().size(); i++) {
        if(getAllEmployee().get(i).getEmail().equals(email)) return getAllEmployee().get(i);
    }
    return null;
}
```

- In best case it is linear time because either strings will be compared by equals or loop will run until the end. Worst case loop will run n times and string is compared. N is size, m is string lenght.
  $T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta(n*m)$. $T(n,m) = O(n*m) = \Omega(n)$.



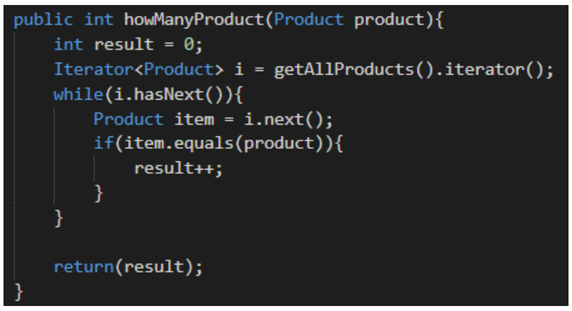
```java
public Boolean findProduct(String name, String color, String model){
    Product searched = new Product(name, color, model);
    return(getAllProducts().contains(searched));
}
```

- Contain is constant best case, linear worst case. So,
  $T_b(n) = \theta(1)$, $T_w(n) = \theta(n)$. $T(n) = O(n)$.



```java
/** Add products to stock if it does not exist before.
*/
public void refreshStock(){
    Iterator<Product> i = getAllProducts().iterator();
    while(i.hasNext()){
        Product item = i.next();
        if(!(getStock().contains(item))){
            getStock().add(item);
        }
    }
}
```

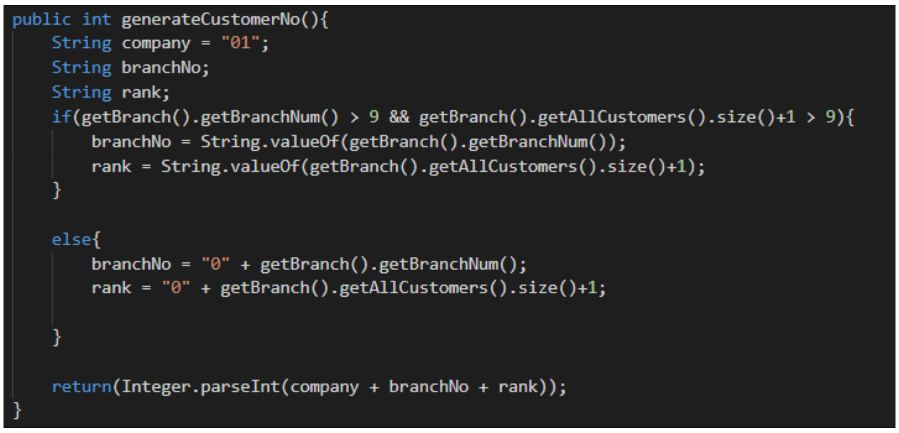
- Outer loop will run n times each, but contains can be constant time in the best case, linear in worst case. Adding at the end of the hybridlist is constant time.
  $T_b(n) = \theta(n)$, $T_w(n) = \theta(n^2)$. $T(n) = O(n^2) = \Omega(n)$.



```java
public int howManyProduct(Product product){
    int result = 0;
    Iterator<Product> i = getAllProducts().iterator();
    while(i.hasNext()){
        Product item = i.next();
        if(item.equals(product)){
            result++;
        }
    }

    return(result);
}
```

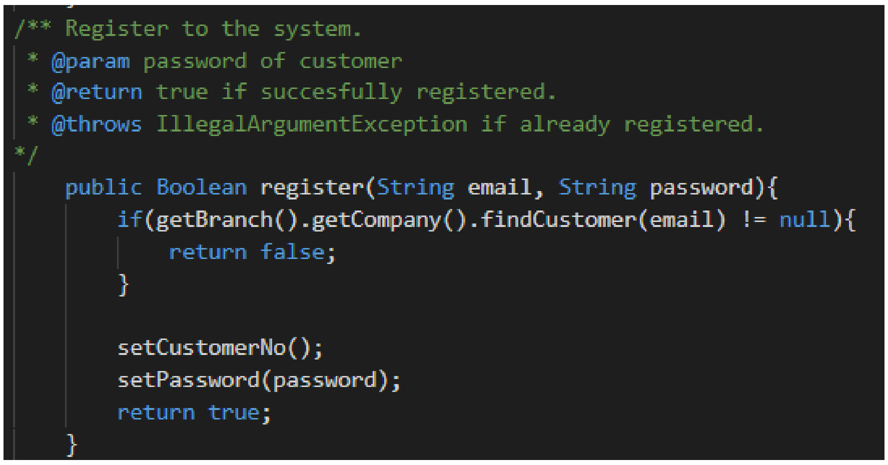
- Loop will run n times always. But equals can be constant in best case and linear in worst case because it is comparing strings.
  $T_b(n) = \theta(n)$, $T_w(n) = \theta(n^2)$. $T(n) = O(n^2) = \Omega(n)$.

4- Customer Class



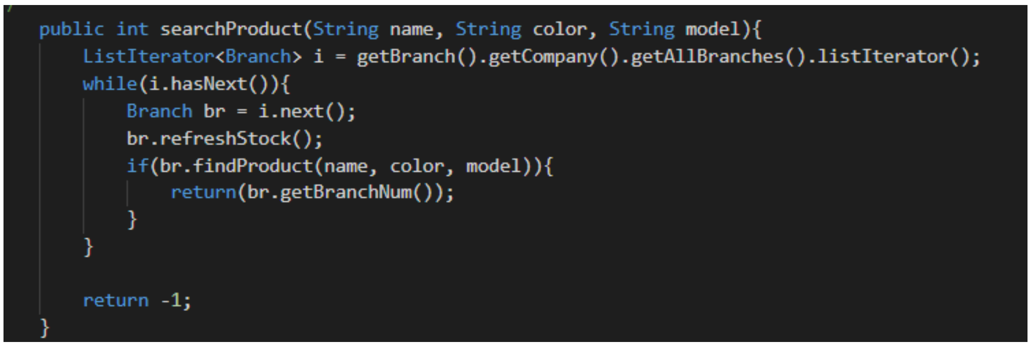
```java
public int generateCustomerNo(){
    String company = "01";
    String branchNo;
    String rank;
    if(getBranch().getBranchNum() > 9 && getBranch().getAllCustomers().size()+1 > 9){
        branchNo = String.valueOf(getBranch().getBranchNum());
        rank = String.valueOf(getBranch().getAllCustomers().size()+1);
    }

    else{
        branchNo = "0" + getBranch().getBranchNum();
        rank = "0" + getBranch().getAllCustomers().size()+1;

    }

    return(Integer.parseInt(company + branchNo + rank));
}
```

- **$T(n) = \theta(1)$.**



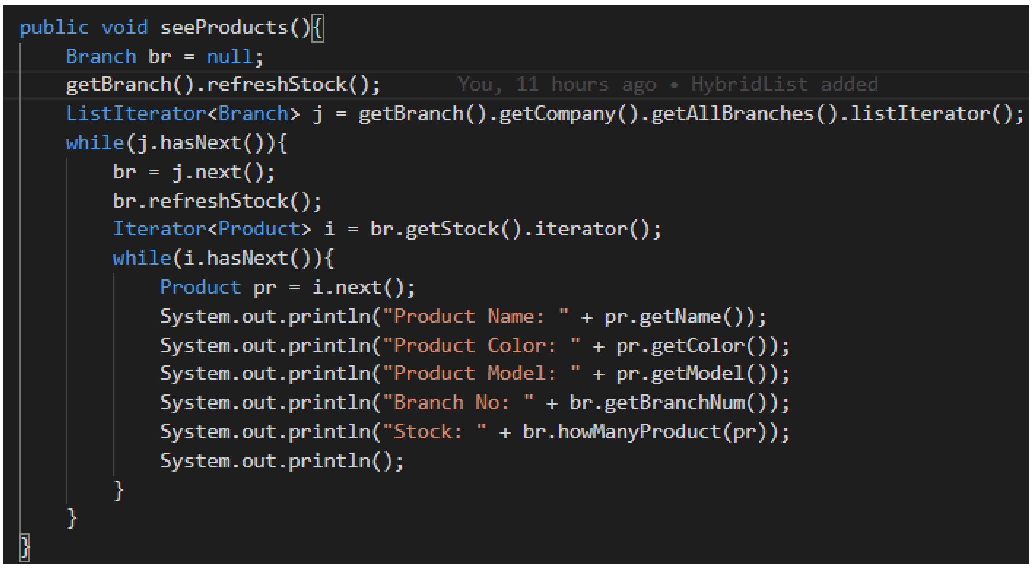
```java
/** Register to the system.
 * @param password of customer
 * @return true if succesfully registered.
 * @throws IllegalArgumentException if already registered.
*/
    public Boolean register(String email, String password){
        if(getBranch().getCompany().findCustomer(email) != null){
            return false;
        }

        setCustomerNo();
        setPassword(password);
        return true;
    }
```

- Find customer is constant best case, $\theta(n^2)$ worst case. So, it is too.
  $T_b(n) = \theta(1)$, $T_w(n) = \theta(n^2)$. $T(n) = O(n^2)$.



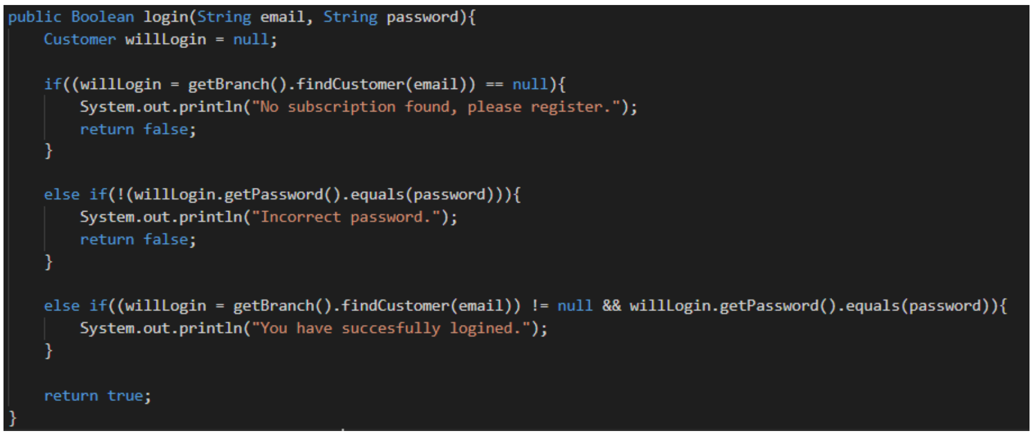
```java
public int searchProduct(String name, String color, String model){
    ListIterator<Branch> i = getBranch().getCompany().getAllBranches().listIterator();
    while(i.hasNext()){
        Branch br = i.next();
        br.refreshStock();
        if(br.findProduct(name, color, model)){
            return(br.getBranchNum());
        }
    }

    return -1;
}
```

- Loop will run n times and find product will run m times worst case, constant best case. So, **$T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta(n*m)$. $T(n,m) = O(n*m)$.**



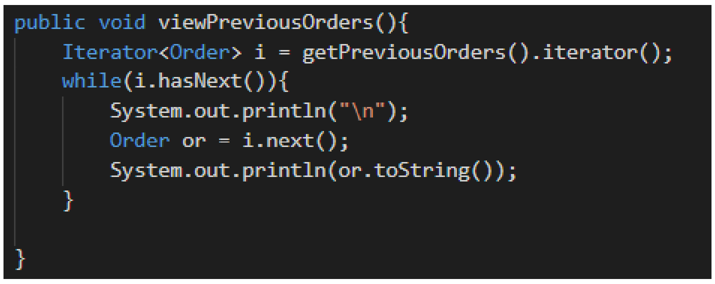
```java
public void seeProducts(){
    Branch br = null;
    getBranch().refreshStock();        You, 11 hours ago • HybridList added
    ListIterator<Branch> j = getBranch().getCompany().getAllBranches().listIterator();
    while(j.hasNext()){
        br = j.next();
        br.refreshStock();
        Iterator<Product> i = br.getStock().iterator();
        while(i.hasNext()){
            Product pr = i.next();
            System.out.println("Product Name: " + pr.getName());
            System.out.println("Product Color: " + pr.getColor());
            System.out.println("Product Model: " + pr.getModel());
            System.out.println("Branch No: " + br.getBranchNum());
            System.out.println("Stock: " + br.howManyProduct(pr));
            System.out.println();
        }
    }
}
```

- Refresh stock is $T_b(n) = \theta(n)$, $T_w(n) = \theta(n^2)$. $T(n) = O(n^2) = \Omega(n)$.
- Outer loop will run n times and inner loop will run m times and refresh in the each branch. **$T_b(n,m) = \theta(n*(n+m))$, $T_w(n,m) = \theta(n*(n^2 + m))$. $T(n,m) = O\ n*(n^2 + m)) = \Omega(n*(n+m))$.**



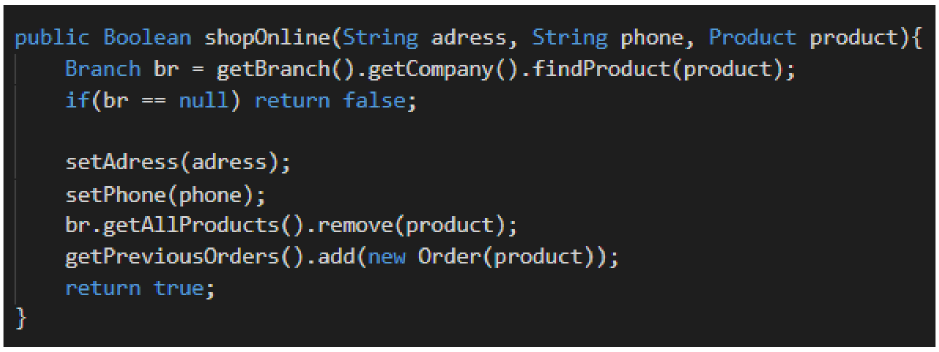
```java
public Boolean login(String email, String password){
    Customer willLogin = null;

    if((willLogin = getBranch().findCustomer(email)) == null){
        System.out.println("No subscription found, please register.");
        return false;
    }

    else if(!(willLogin.getPassword().equals(password))){
        System.out.println("Incorrect password.");
        return false;
    }

    else if((willLogin = getBranch().findCustomer(email)) != null && willLogin.getPassword().equals(password)){
        System.out.println("You have succesfully logined.");
    }

    return true;
}
```

- Find customer is $T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta(n*m)$. $T(n,m) = O(n*m) = \Omega(n)$ and equals is constant best case and linear worst case. **$T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta((n*m)+ m)$. $T(n,m)$.**



```java
public void viewPreviousOrders(){
    Iterator<Order> i = getPreviousOrders().iterator();
    while(i.hasNext()){
        System.out.println("\n");
        Order or = i.next();
        System.out.println(or.toString());
    }

}
```

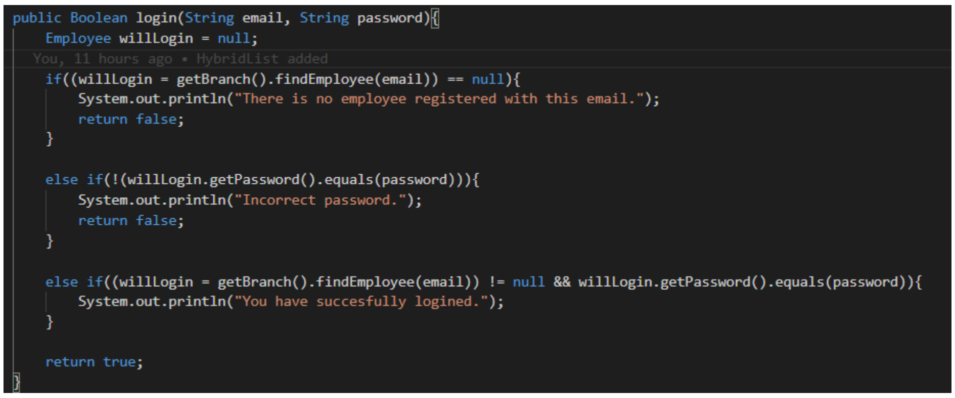
- It will run n times.
  T(n) = θ(n).



```java
public Boolean shopOnline(String adress, String phone, Product product){
    Branch br = getBranch().getCompany().findProduct(product);
    if(br == null) return false;

    setAdress(adress);
    setPhone(phone);
    br.getAllProducts().remove(product);
    getPreviousOrders().add(new Order(product));
    return true;
}
```

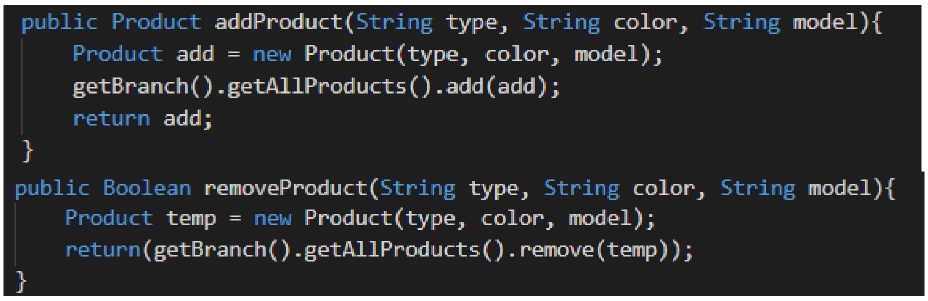
- Find product is $T_b(n) = \theta(1)$, $T_w(n) = \theta(n)$. $T(n) = O(n)$.
- Removing can be constant if it is at the end but, it will just require “MAX_NUMBER” shifting and we know the max number so it is constant.
- Adding at the end of the hybrid list is constant. So,
  $T_b(n) = \theta(1)$, $T_w(n) = \theta(n)$. $T(n) = O(n)$.

5- Employee Class



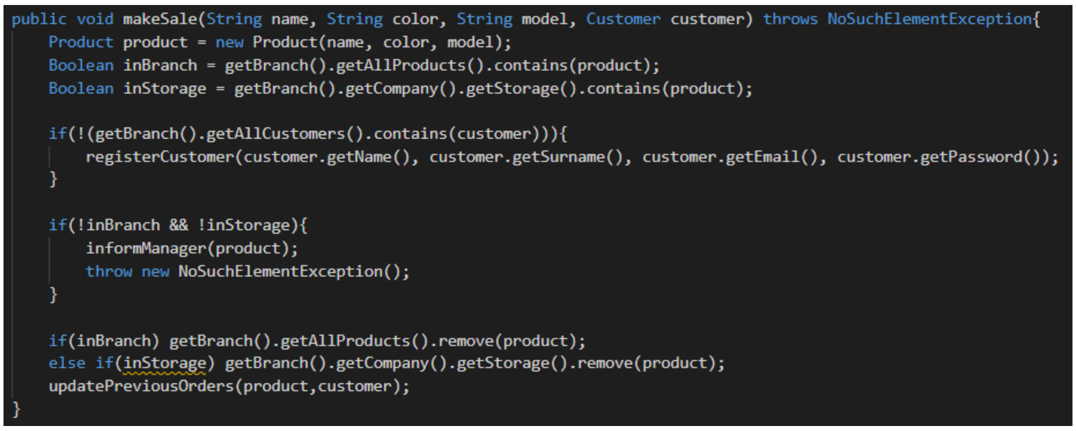
```java
public Boolean login(String email, String password){
    Employee willLogin = null;
  You, 11 hours ago • HybridList added
    if((willLogin = getBranch().findEmployee(email)) == null){
        System.out.println("There is no employee registered with this email.");
        return false;
    }

    else if(!(willLogin.getPassword().equals(password))){
        System.out.println("Incorrect password.");
        return false;
    }

    else if((willLogin = getBranch().findEmployee(email)) != null && willLogin.getPassword().equals(password)){
        System.out.println("You have succesfully logined.");
    }

    return true;
}
```

- Find employee is $T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta(n*m)$. $T(n,m) = O(n*m) = \Omega(n)$ and equals is constant best case and linear worst case.
  $T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta((n*m)+ m)$. $T(n,m)$.



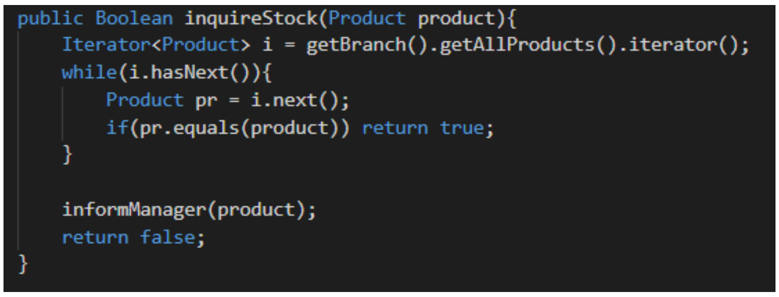
```java
public Product addProduct(String type, String color, String model){
    Product add = new Product(type, color, model);
    getBranch().getAllProducts().add(add);
    return add;
}
public Boolean removeProduct(String type, String color, String model){
    Product temp = new Product(type, color, model);
    return(getBranch().getAllProducts().remove(temp));
}
```

- Adding at the end of the hybridlist and removing is constant. Because we know how many shifting it does need.
  T(n) = θ(1).



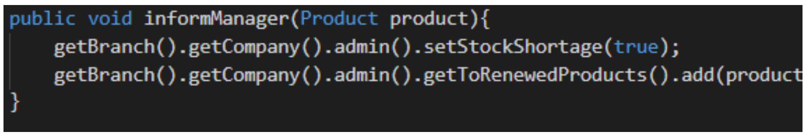
```java
public void makeSale(String name, String color, String model, Customer customer) throws NoSuchElementException{
    Product product = new Product(name, color, model);
    Boolean inBranch = getBranch().getAllProducts().contains(product);
    Boolean inStorage = getBranch().getCompany().getStorage().contains(product);

    if(!(getBranch().getAllCustomers().contains(customer))){
        registerCustomer(customer.getName(), customer.getSurname(), customer.getEmail(), customer.getPassword());
    }

    if(!inBranch && !inStorage){
        informManager(product);
        throw new NoSuchElementException();
    }

    if(inBranch) getBranch().getAllProducts().remove(product);
    else if(inStorage) getBranch().getCompany().getStorage().remove(product);
    updatePreviousOrders(product,customer);
}
```

- Contains is constant best, linear worst case.
- informing manager is constant.
- Adding and removing is constant.
- Updating orders is constant. So,
  $T_b(n)$ = θ(1), $T_w(n)$ = θ(n). T(n) = O(n).



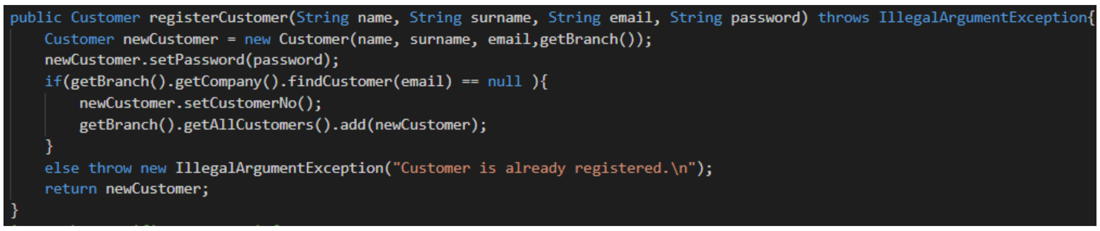
```java
public Boolean inquireStock(Product product){
    Iterator<Product> i = getBranch().getAllProducts().iterator();
    while(i.hasNext()){
        Product pr = i.next();
        if(pr.equals(product)) return true;
    }

    informManager(product);
    return false;
}
```

- It will run only 1 times in best case but equals is linear in these case. In worst case it will run n times.
  $T_b(n,m)$ = θ(m), $T_w(n,m)$ = θ(n*m). T(n) = O(n*m) = Ω(m).



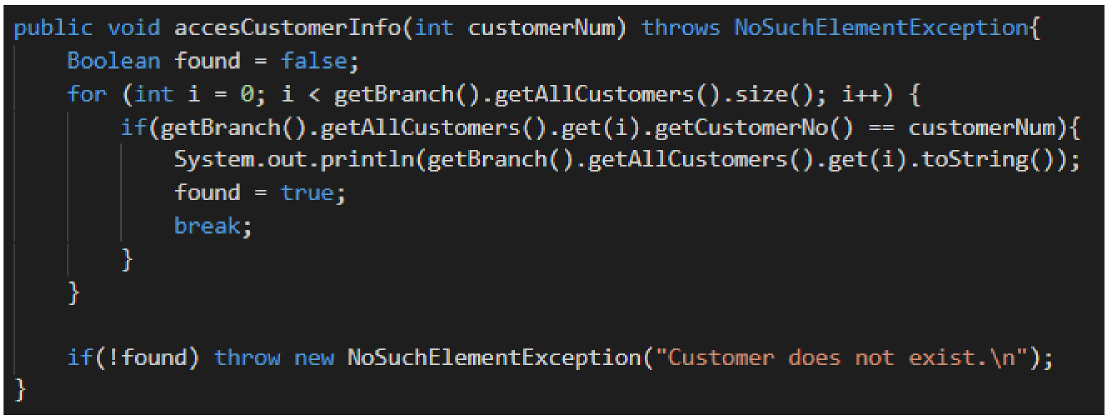
```java
public void informManager(Product product){
    getBranch().getCompany().admin().setStockShortage(true);
    getBranch().getCompany().admin().getToRenewedProducts().add(product
}
```

- Adding is constant in hybrid list.
  T(n) = θ(1).



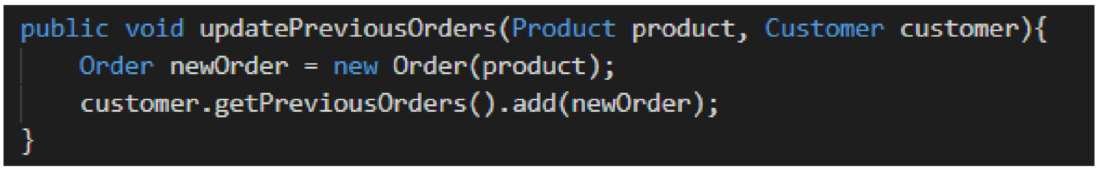
```java
public Customer registerCustomer(String name, String surname, String email, String password) throws IllegalArgumentException{
    Customer newCustomer = new Customer(name, surname, email,getBranch());
    newCustomer.setPassword(password);
    if(getBranch().getCompany().findCustomer(email) == null ){
        newCustomer.setCustomerNo();
        getBranch().getAllCustomers().add(newCustomer);
    }
    else throw new IllegalArgumentException("Customer is already registered.\n");
    return newCustomer;
}
```

- Find customer is is $T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta(n*m)$. $T(n,m) = O(n*m) = \Omega(n)$ and adding is constant. So,
  $T_b(n,m) = \theta(n)$, $T_w(n,m) = \theta(n*m)$. $T(n,m) = O(n*m) = \Omega(n)$.



```java
public void accesCustomerInfo(int customerNum) throws NoSuchElementException{
    Boolean found = false;
    for (int i = 0; i < getBranch().getAllCustomers().size(); i++) {
        if(getBranch().getAllCustomers().get(i).getCustomerNo() == customerNum){
            System.out.println(getBranch().getAllCustomers().get(i).toString());
            found = true;
            break;
        }
    }

    if(!found) throw new NoSuchElementException("Customer does not exist.\n");
}
```

- Loop will run 1 time in best case, it wont found in worst case.
  $T_b(n) = \theta(1)$, $T_w(n) = \theta(n)$. $T(n) = O(n)$.



```java
public void updatePreviousOrders(Product product, Customer customer){
    Order newOrder = new Order(product);
    customer.getPreviousOrders().add(newOrder);
}
```

- Adding is constant.
  $T(n) = \theta(1)$.