GTU Department of Computer Engineering

CSE 222/505 - Spring 2021
Homework #03 Report

Muhammed Oğuz
1801042634

PART 2

Running Times

1- Administrator Class

```
@Override // O(n)
public void addBranch(Branch Branch) throws Exception {
    if(company.getBranches().contains(Branch)) // O(n)
        throw new Exception("Branch adding failed. Same Branch already exist");
    company.getBranches().addLast(Branch); // O(1)
    return;
}
```

First of all, my addBranch function looks for containing already. It takes O(n) time. And whole function takes O(n)

```
@Override // O(n)
public void removeBranch(Branch Branch) throws Exception {
    if (!company.getBranches().contains(Branch)) // O(n)
        throw new Exception("Branch removing failed. There is no Branch with this id");

    Branch temp = company.getBranches().get(Branch); // O(1)
    company.getBranches().remove(Branch);
    // remove all employees from removed Branch.
    if (company.getEmployees().removeAll(temp.getEmployees())) // O(n)
        System.out.println("Success removeAll");
    else
        System.out.println("RemoveAll failed");
```

removeBranc() method takes O(n) time.

```
@Override // O(1)
public void addBranchEmployee(Employee employee) throws Exception {
    // add to Branch employees and all employees.
    if (employee == null)
        throw new Exception("Employee adding failed");
    if (company.getBranches().get(employee.getBranch()) == null)
        throw new Exception("Employee adding failed. Same employee already exist or wrong Branch id");
    if(!(company.getEmployees().add(employee) && company.getBranches().get(employee.getBranch()).getEmployees().add(employee)))
        throw new Exception("Employee adding failed. Same employee already exist or wrong Branch id");
    return;
}
```

İn ArrayList, add method takes O(1) time. This just use add method. So takes O(1) time

```
@Override // O(n)
public void removeBranchEmployee(Employee employee) throws Exception {
    Employee temp;
    if ((temp = company.getEmployees().get(employee)) != null)
    {
        company.getEmployees().remove(temp);
        company.getBranches().get(temp.getBranch()).getEmployees().remove(temp);
        return;
    }
```

Takes O(n) time. Because removve method shifts all items.

2- Branch Class

```
}

public int getId() {
    return id;
}

public HybridList<Product> getProducts() {
    return products;
}
public KWArrayList<Employee> getEmployees() {
    return employees;
}
public void setId(int id) {
    this.id = id;
}
```

Getter and setters. All takes O(1)

3- Company Class

```java
public boolean addAdmin(Administrator admin){
    if(admins.add(admin))
        return true;
    return false;
}

public String getCompanyName() {
    return companyName;
}
public KWArrayList<Administrator> getAdmins() {
    return admins;
}
public KWArrayList<Employee> getEmployees() {
    return employees;
}
public IKWArrayList<Customer> getCustomers() {
    return customers;
}
public KWLinkedList<Branch> getBranches() {
    return Branches;
}
```

All takes O(1) time. Add method uses ArrayList, its add method takes O(1), getters takes O(1) time either.

```java
@Override
public Administrator loginAdmin(Administrator admin) {
    if (admins.contains(admin))
    {
        //System.out.println("Admin contains.");
        if(admins.get(admin).getPassword().equals(admin.getPassword()))
        {
            return admins.get(admin);
        }
    }
    return null;
}

@Override
public Employee loginEmployee(Employee employee) {
    if (employees.contains(employee))
    {
        //System.out.println("Employee contains.");
        if(employees.get(employee).getPassword().equals(employee.getPassword()))
        {
            return employees.get(employee);
        }
    }
    return null;
}

@Override
public Customer loginCustomer(Customer customer) {
    if (customers.contains(customer))
    {
        //System.out.println("customers contains");
        if (customers.get(customer).getPassword().equals(customer.getPassword()) &&
            customers.get(customer).getMail().equals(customer.getMail()))
        {
            return customers.get(customer);
        }
    }
    return null;
}
```

Login methods takes O(1) times. Because all of them just uses getters.

```
    @Override
    public String toString() {
        String r =  "\nInformation of Company -> " + getCompanyName() + "\n" +
                    "Admins     \n----------\n" + admins.toString()     + "\n" +
                    "Branches and Products   \n----------\n" + Branches.toString()   + "\n" +
                    "Employees  \n----------\n" + employees.toString()  + "\n" +
                    "Customers  \n----------\n" + customers.toString();
        return r;
    }

}
```

toString() takes O(1) time. Because, all toString methods for values takes O(n) time for their toString methods.

4- Customer Class

```
    public void setMail(String mail) {
        this.mail = mail;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public void setId(int id) {
        this.id = id;
    }

    public int getId() {
        return id;
    }

    public String getMail() {
        return mail;
    }

    public String getAddress() {
        return address;
    }

    public String getPhone() {
        return phone;
    }
```

Setters and getters. Takes O(1) time.

```java
@Override
public boolean buyOffline(Company company, Branch Branch, Product product) {

    if (company.getBranches().get(Branch).getProducts().contains(product)) {
        if (company.getBranches().get(Branch).getEmployees().size() == 0) {
            System.out.println("This Branch has no employee");
            return false;
        }

        try {
            company.getBranches().get(Branch).getEmployees().get(0).removeProduct(product);
        } catch (Exception e) {
            e.printStackTrace();
            return false;
        }

        this.add(product);
        return true;
    }

    return false;
```

Takes O(n) time. Because, ne of the used methods is contains() method, it takes O(n) and neither other functions not takes more time than this. So takes O(n)

```java
// Search all Branches till found desired product.
@Override
public boolean buyOnline(Company company, Product product) {
    for (int i = 0; i < company.getBranches().size(); i++) {
        // buyOffline function looks a Branch. For loop looks for all Branches
        if (buyOffline(company, company.getBranches().get(i), product))
            return true;
    }
    return false;
}
```

Calls buyOffline methods n times. So takes O(n^2) time.

```java
public void add(Product product){
    if (products.contains(product))
    {
        Product temp = products.get(product);
        temp.setStock(temp.getStock() + product.getStock());
    }
    else
        products.add(product);
}
```

Takes O(n^2) time. First check if this product is already exist. İt takes O(n) but not considered. After than, uses add method of HyberList class. This add method takes O(n^2).

```java
@Override
public String toString() {
    if (address == null && phone == null)
        return new String("Customer:" + getName() + " " + getSurname() +"\t" + " ID: " + getId() + " Mail:" + getMail());
    else
        return new String("Customer:" + getName() + " " + getSurname() +"\t" + " ID: " + getId() + " Mail:" + getMail() +
                " Address: " + getAddress() + " Phone: " + getPhone());
}
```

Takes O(1). Because just uses getters.

5- Employee Class

```java
public void setBranch(Branch Branch) {
    this.Branch = Branch;
}

public Branch getBranch() {
    return Branch;
}
```

Setteer and getter. Takes O(1)

```
    @Override
    public void addCustomer(Customer customer) throws Exception {
        int temp = company.getUniqueCustomerId();
        customer.setId(temp);
        company.setUniqueCustomerId(++temp);
        if(!company.getCustomers().add(customer))
            throw new Exception("Adding Customer Failed. Same Customer already exist");
    }
```

Uses ArrayList class as a data struct. So add method takes O(1) time.

```
    @Override
    public void removeCustomer(Customer customer) throws Exception {
        if (!company.getCustomers().remove(customer))
            throw new Exception("Remove Customer Failed, There is no match");
        return;
    }
```

Uses ArrayList classs as a data struct. So remove methods takes O(n) because of shifting.

```
    @Override
    public void addProduct(Product product) throws Exception {
        if (product.getStock() ≤ 0)
        {
            System.out.println("Product stock can not be 0 or lower");
            throw new Exception("Product adding failed.");
        }
        Product temp = getBranch().getProducts().get(product);
        if (temp ≠ null)
        {
            temp.setStock(product.getStock() + temp.getStock());
        }
        else
        {
            getBranch().getProducts().add(product);
        }
    }
```

Uses HyberList as a data struct. Add method takes O(n^2) time.

```java
@Override
public void removeProduct(Product product) throws Exception {
    Product temp = getBranch().getProducts().get(product);

    if (temp != null)
    {
        int check = temp.getStock() - product.getStock();
        if (check < 0)
        {
            company.getAdmins().get(0).informedProducts(getBranch(), temp);
            throw new Exception("Removing Product stock Failed");
        }
        else
        {
            temp.setStock(temp.getStock() - product.getStock());
            if (check == 0)
                company.getAdmins().get(0).informedProducts(getBranch(), temp);
            return;
        }
    }
    throw new Exception("Removing Product stock Failed");
}

/**
```

Uses hyberList as a data struct. Remove method takes O(n^2) time

```java
@Override
public HybridList<Product> customerProducts(int id) throws Exception {
    for (int i = 0; i < company.getCustomers().size(); i++)
    {
        if (company.getCustomers().get(i).getId() == id)
            return company.getCustomers().get(i).getProducts();
    }
    throw new Exception("There is no customer with this id");
}
```

For loop takes O(n) and get customers via ArrayList data struct takes O(1). So takes O(n) time.

```java
@Override
public String toString() {
    return new String("Employee:" + getName() + " " +  getSurname()+ " Branch Id: " + getBranch().getId());
}
```

Just getters. Takes O(1) time.

PART 2

1. SYSTEM REQUIREMENTS

First, this is an automation system of a company. So, for a working system. There should be a company.

```
System.out.println("Load Example Company");
company = useExampleCompany();
```

As example, loads an example company.

When loading a company, This functions does the following. Create

an admin

```
CompanyAdministrator admin = new CompanyAdministrator(company, "Erdogan", "Hoca", "123");
```

Create 4 Branches

```
CompanyBranch branch1 = new CompanyBranch(1);
CompanyBranch branch2 = new CompanyBranch(2);
CompanyBranch branch3 = new CompanyBranch(3);
CompanyBranch branch4 = new CompanyBranch(4);
```

Create 4 Employees

```
CompanyEmployee employee1 = new CompanyEmployee(company, "Burak", "Hoca", "123", branch1);
CompanyEmployee employee2 = new CompanyEmployee(company, "Basak", "Hoca", "123", branch2);
CompanyEmployee employee3 = new CompanyEmployee(company, "Ilhan", "Hoca", "123", branch3);
CompanyEmployee employee4 = new CompanyEmployee(company, "Muhammed", "Student", "123", branch4);
```

Create 2 Customers

```
Customer customer1 = new Customer("Siftah", "Para", "123", "123");
Customer customer2 = new Customer("Foo", "Bar", "123", "123");
```

And add all of them to company respectively,

```
company.addAdmin(admin);
admin.addBranch(branch1); admin.addBranch(branch2);
admin.addBranch(branch3); admin.addBranch(branch4);
admin.addBranchEmployee(employee1); admin.addBranchEmployee(employee2);
admin.addBranchEmployee(employee3); admin.addBranchEmployee(employee4);
employee1.addCustomer(customer1); employee1.addCustomer(customer2);
```

Notice that, company adds admin, admin adds employee and branch and employee adds customer.

And finally, add products to sell

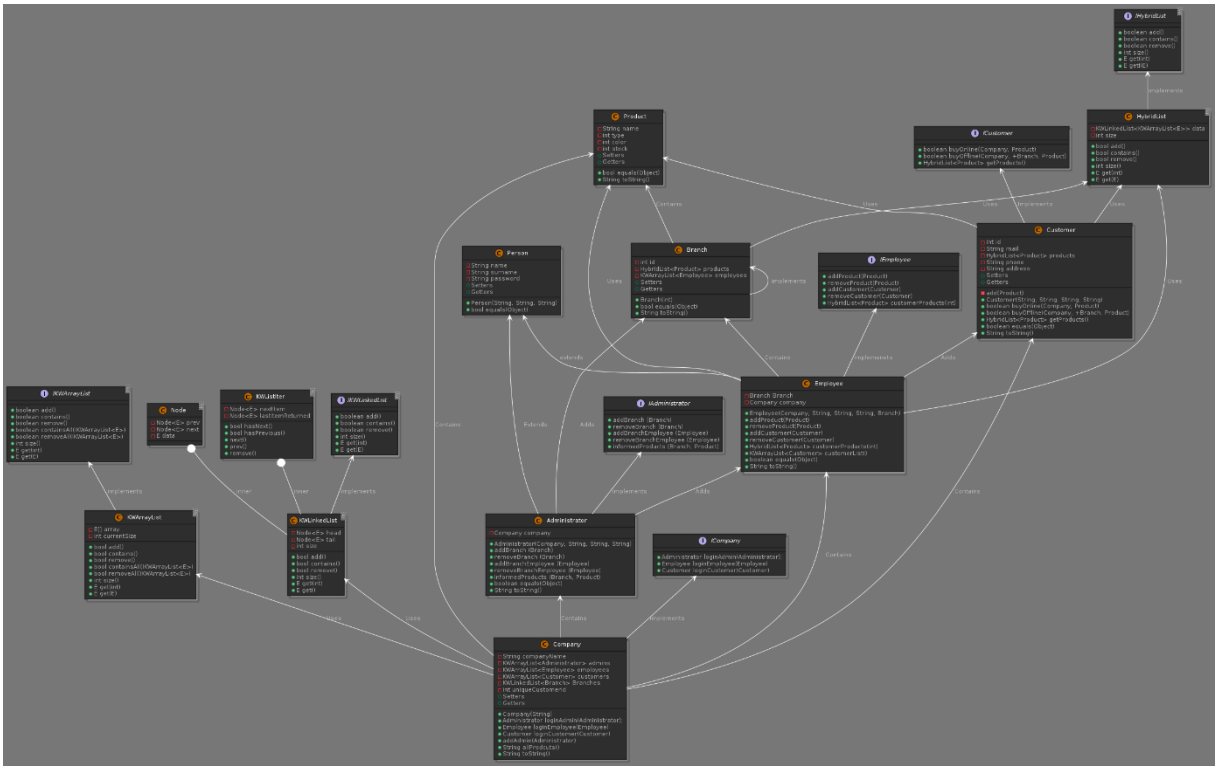
```
employee1.addProduct(new Product("Chair", 1, 1, 5));
employee1.addProduct(new Product("Desk", 3, 4, 5));
employee1.addProduct(new Product("Table", 2, 5, 5));
employee1.addProduct(new Product("Bookcase", 1, 4, 5));
employee1.addProduct(new Product("Cabinet", 4, 0, 5));
```

After this process, your automation system for this company is ready to use.

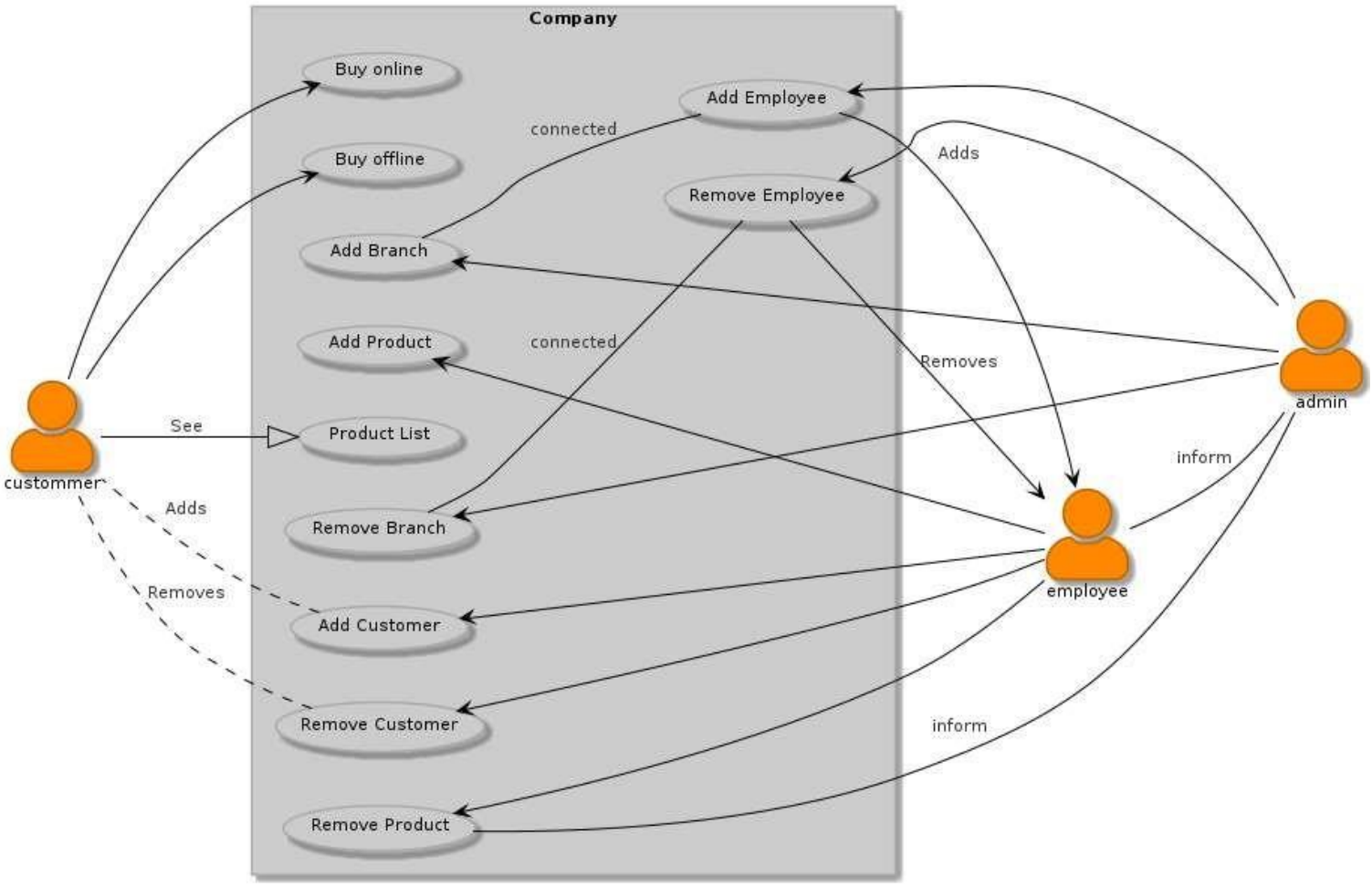
2. USE CASE AND CLASS DIAGRAMS

2.1 Class Diagram UML

(HD Version in diagram folder)



2.2 Use Case Diagram



3. PROBLEM SOLUTION APPROACH

First of all, I have to implement a representation of LinkedList and ArrayList classes. I took codes from book, online java doc and implement with my knowledge. I try to connect Java OOP principles and implemented those classes for my usage. In my first homework it was just a ArrayContainer that behaves like ArrayList. So KWArrayList class is easier than KWLinkedList.

KWLinkedList class was a bit harder because, implementing Node and Iterator class was new to me. Altough I implemented successfully.

As a solution. I draw a plan on a paper to, which class will represent which work or which class will extends or contains which plan. This was like Use Case and Class UML diagram. After those thinking process. I start to implement my interfaces and source codes separately to see what I should do and how I should.

There was another problem when I implement the code. This was, accessing a parent class from a child class. I found a solution for this like, adding parent class to derived class as a field and assign via a constructor solved my problem. Also, I use Node and Iterator class as inner classes on my KWLinkedList class.

4. TEST CASES

Create a company

```
Company company = new Company("GTU Office Solutions");
```

Create and add a admin

```
CompanyAdministrator admin = new CompanyAdministrator(company, "Erdogan", "Hoca", "123");
company.addAdmin(admin);
```

Login as a admin

```
admin = company.loginAdmin(
        new CompanyAdministrator(company, getStr("Name:"), getStr("Surname:"), getStr("Password:")));
```

Add Branch

```
((CompanyAdministrator) admin).addBranch(new CompanyBranch(getInt("Enter Branch Id:")));
```

Add Employee

```
((CompanyAdministrator) admin)
        .addBranchEmployee(new CompanyEmployee(company, getStr("Name:"), getStr("Surname:"),
                getStr("Password:"), new CompanyBranch(getInt("Enter Branch Id:"))));
```

Remove Branch

```
((CompanyAdministrator) admin).removeBranch(new CompanyBranch(getInt("Enter Branch Id:")));
```

Remove Employee

```
((CompanyAdministrator) admin).removeBranchEmployee(
        new CompanyEmployee(company, getStr("Name:"), getStr("Surname:"), null, null));
System.out.println("Remove Employee Successful");
```

See Branch List

```
System.out.println("Branch List\n" + company.getBranches());
```

See Employee List

```
System.out.println("Employee List\n" + company.getEmployees());
```

Login as a employee

```
employee = company.loginEmployee(
        new CompanyEmployee(company, getStr("Name:"), getStr("Surname:"), getStr("Password:"), null));
```

Add Customer

```
employee.addCustomer(getCustomer());
```

Add Product

```
employee.addProduct(getProduct());
```

Remove Customer

```
employee.removeCustomer(new Customer(getStr("Name:"), getStr("Surname:"), null, null));
```

Remove Product

```
employee.removeProduct(getProduct());
```

See a customer's products

```
employee.customerProducts(temp);
```

See Customer list

```
System.out.println("Customers List\n" + company.getCustomers());
```

Login as a customer

```
customer = company.loginCustomer(getCustomer());
```

Buy online (If this is the first time, than ask for address and phone number)

```
se 1:
 if (customer.getPhone() == null && customer.getAddress() == null) {
     System.out.println("First Time Online Buyers Should Provide Address and Phone");
     customer.setAddress(getStr("Address:"));
     customer.setPhone(getStr("Phone:"));
 }
 System.out.println("Enter Desired Product Information");
 if (customer.buyOnline(company, getProduct()))
```

Buy Offline

```
if (customer.buyOffline(company, new CompanyBranch(getInt("Branch Id:")), getProduct()))
```

See old purchases

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

See all products of company

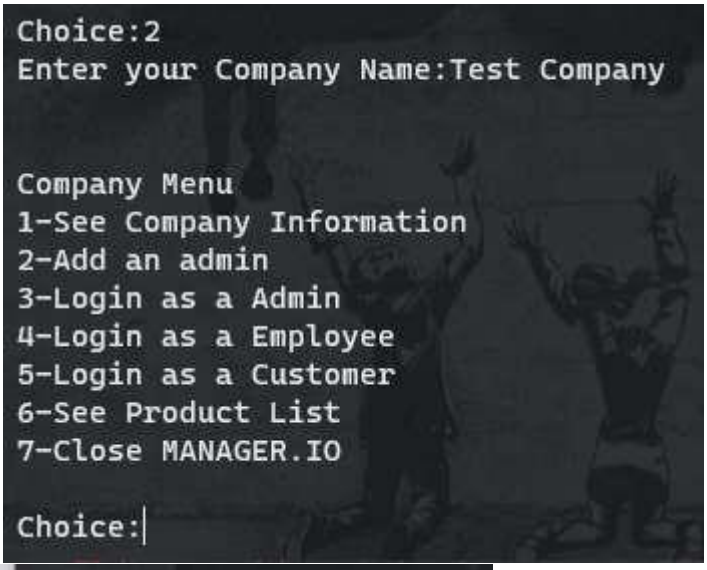
```
System.out.println(company.allProducts());
```

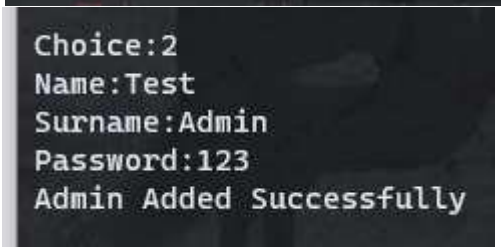
5. RUNNING AND RESULTS

There are several options for result. In this report, menu option was shown.

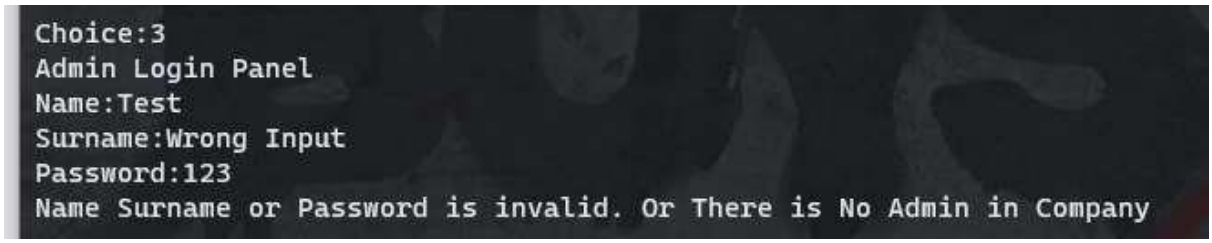
Menu when run the program.

Test Company Created





Login as admin (not success)



Successful login as admin

```
Choice:3
Admin Login Panel
Name:Test
Surname:Admin
Password:123
Login Success
```

ADMIN PANEL

Add Branch

Add Employee

```
Choice:1
Enter Branch Id:3
Branch successfully added.
```

```
Choice:1
Enter Branch Id:3
Branch adding failed. Same branch already exist
```

```
Choice:2
Name:Test
Surname:Employee
Password:123
Enter Branch Id:3
Employee Successfully Added.
```

```
Choice:2
Name:Test
Surname:Employee
Password:123
Enter Branch Id:2
Employee adding failed. Same employee already exist or wrong branch id
```

Remove Branch

```
Enter Branch Id:3
Success removeAll
Branch, its employees and its products are successfully removed
```

```
Enter Branch Id:3
Branch removing failed. There is no branch with this id
```

Remove Employee

```
Choice:4
Employee List
Employee:Test Employee Branch Id: 3

Name:Test
Surname:EMployee
Remove Employee Failed. Employee does not match
```

EMPLOYEE PANEL

```
Choice:4
Employee Panel
Name:Test
Surname:Employee
Password:123
Login Success
```

Add Product

```
Choice:1
Name:Desk
Type:2
Color(0 for colorless):0
Stock:25
Product Added Successfully
```

```
Choice:1
Name:Desk
Type:2
Color(0 for colorless):0
Stock:40
Product Added Successfully
```

```
Choice:6
Employee's Branch Product List
Product Name:Desk        Type:2 Stock:65
```

Add Customer

```
Choice:2
Name:Test
Surname:Customer
Password:123
Mail:test@customer.com
Adding Customer Successful
```

```
Choice:2
Name:Test
Surname:Customer
Password:134
Mail:45
Adding Customer Failed. Same Customer already exist
```

Remove Product

```
Choice:3
Employee's Branch Product List
Product Name:Desk        Type:2 Stock:65

Name:invalid
Type:Test
Wrong input. Enter an int. Try Again
3
Color(0 for colorless):5
Stock:1
Removing Product stock Failed
```

Remove customer

```
Customers List
Customer:Test Customer   ID: 0 Mail:test@customer.com

Name:invalid
Surname:customer
This item not in the array.
Remove Customer Failed, There is no match
```

CUSTOMER PANEL

Login as a first time.

```
Choice:5
Customer Panel
First time customer? (Y),Type Y for Yes. Other types considered No.
Y
Enter your Info
Name:Customer
Surname:Test
Password:123
Mail:Test@custom.com
Login Success
```

Buy online

```
Choice:1
First Time Online Buyers Should Provide Address and Phone
Address:GTU
Phone:155
Enter Desired Product Information
Name:Desk
Type:2
Color(0 for colorless):0
Stock:30
Purchase Failed. Wrong Product Info
```

Buy offline

```
Choice:2
Branch Id:3
Name:Test
Type:5
Color(0 for colorless):5
Stock:20
Purchase Success
```

Purchase History

```
Choice:3
Product Name:Test        Type:5 Color:5 Stock:20
```

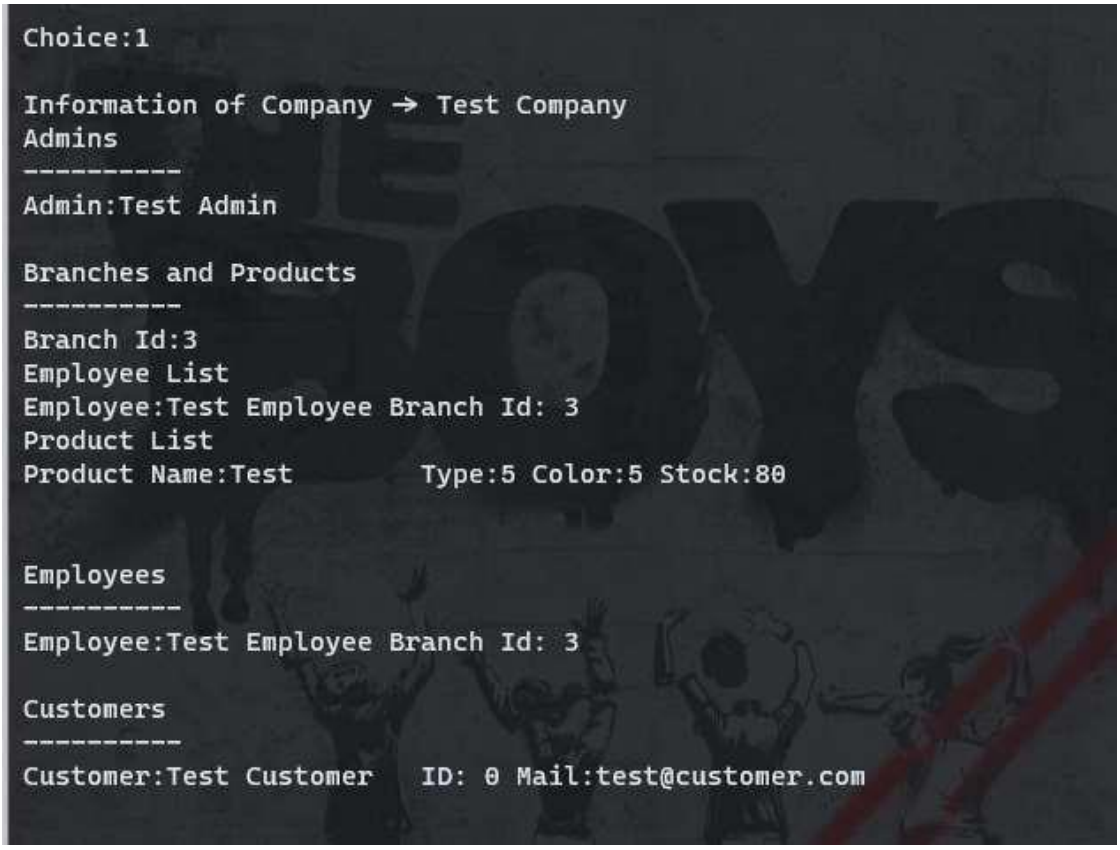
Purchase history from an employee

```
Employee Menu
0-My Profile
1-Add Product
2-Add Customer
3-Remove Product
4-Remove Customer
5-See a Customer's Purchase History
6-Branch Product List
7-See All Customers
8-Log Out

Choice:5
Customers List
Customer:Test Customer   ID: 0 Mail:test@customer.com

Customer ID:0
Product Name:Test        Type:5 Color:5 Stock:20
```

All Info about company


Choice:1
Information of Company → Test Company
Admins

Admin:Test Admin
Branches and Products

Branch Id:3
Employee List
Employee:Test Employee Branch Id: 3
Product List
Product Name:Test Type:5 Color:5 Stock:80
Employees

Employee:Test Employee Branch Id: 3
Customers

Customer:Test Customer ID: 0 Mail:test@customer.com