GIT Department of Computer Engineering

CSE 222/505 – Spring 2020

Homework 1 Report

Fatih Kaan Salgır

171044009

1. SYSTEM REQUIREMENTS

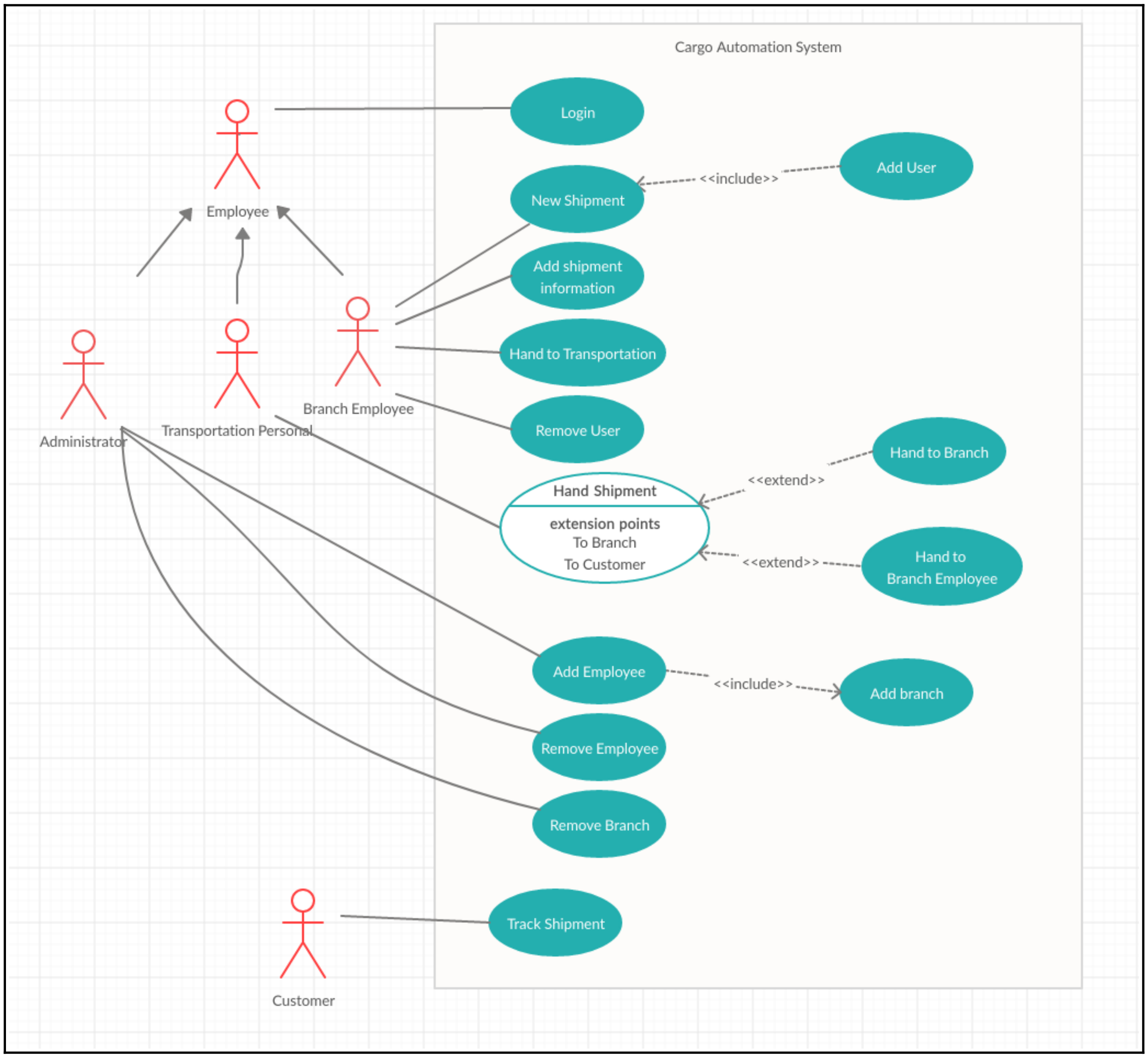
1.1 Functional Requirements

- The program will be used by both company workers and customers. It must support a multi-user interface.
- Branch employees should be able to accept a new shipment or hand shipment to transportation personal.
- Transportation personal should be able to change shipment status to ‘delivered’ or he can hand shipment to another branch employee.
- Admins can add/remove other employees.
- Users can only search for shipments by tracking number.
- Branch employees need to access only the shipments they are responsible for (only if they have the shipments either from customers or transportation employees). So I store both shipments and branch employees in branches.

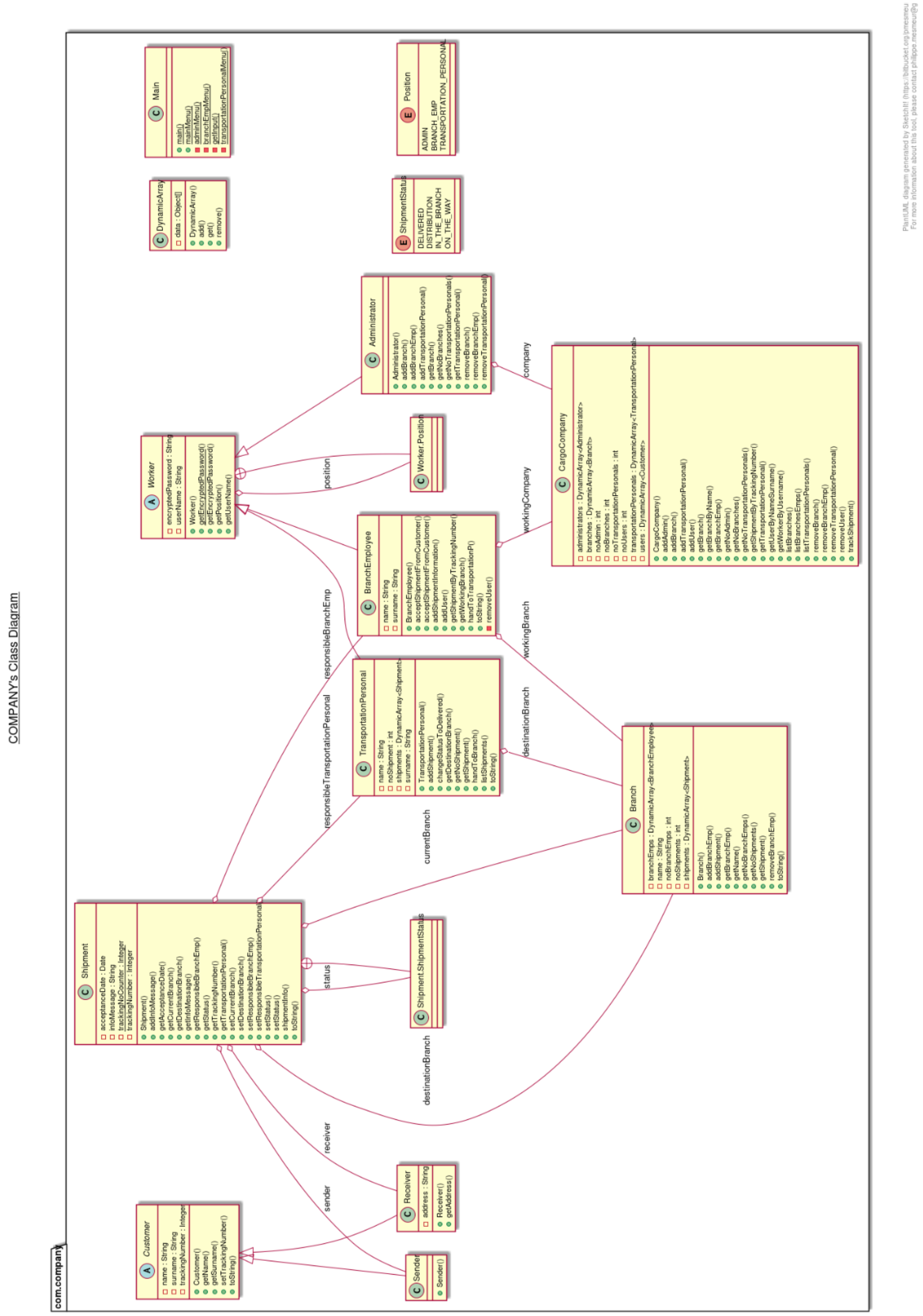
1.2 Non Functional Requirements

- The system must be portable since it needs to appeal to not only workers but also users. It must be supported by different platforms.
- The program must be able to store users' login information securely.
- The program must be maintainable since it will be further improvement requests by users of the program.
- We store user, worker and shipment data in the program and these data tend to be enlarged as the company uses it. It should be scalable.

2. USE CASE DIAGRAMS



3. CLASS DIAGRAMS



4. PROBLEM SOLUTION APROACH

The assignment is developing an automation system for a cargo company. It supposed to be multi-user system. It will be different users and different tasks they can perform. Also, there are shipments which they have all the information about its delivery process. Therefore we can treat all these as set of objects which they have mostly "has a relationship". It's convenient to use Object oriented Paradigm.

Since its a good software developing practice, I designed my class hierarchy by considering encapsulation. However; there will be lots of classes that need to access the same data. To get over this issue I decided to create a class (`CargoCompany`) that stores main data (Admins, Branches, Transportation Personals).

The program will be used by different users, both user and company workers. Therefore it would be better to create different classes for users. Company workers have some common features, so I have created an abstract class to implement common features.

Another problem I encountered is data needs to be stored in arrays with unknown initial sizes. There are lots of arrays of data types that need to be resized every adding and removing operations. Instead of repeating the same code for every data type, I created a generic class `DynamicArray<T>` and I store data as `Object` arrays, and I do the casting to get the data, even though it is unchecked casting. I thought its better to avoid code repetition since it might lead to further maintainability problems.

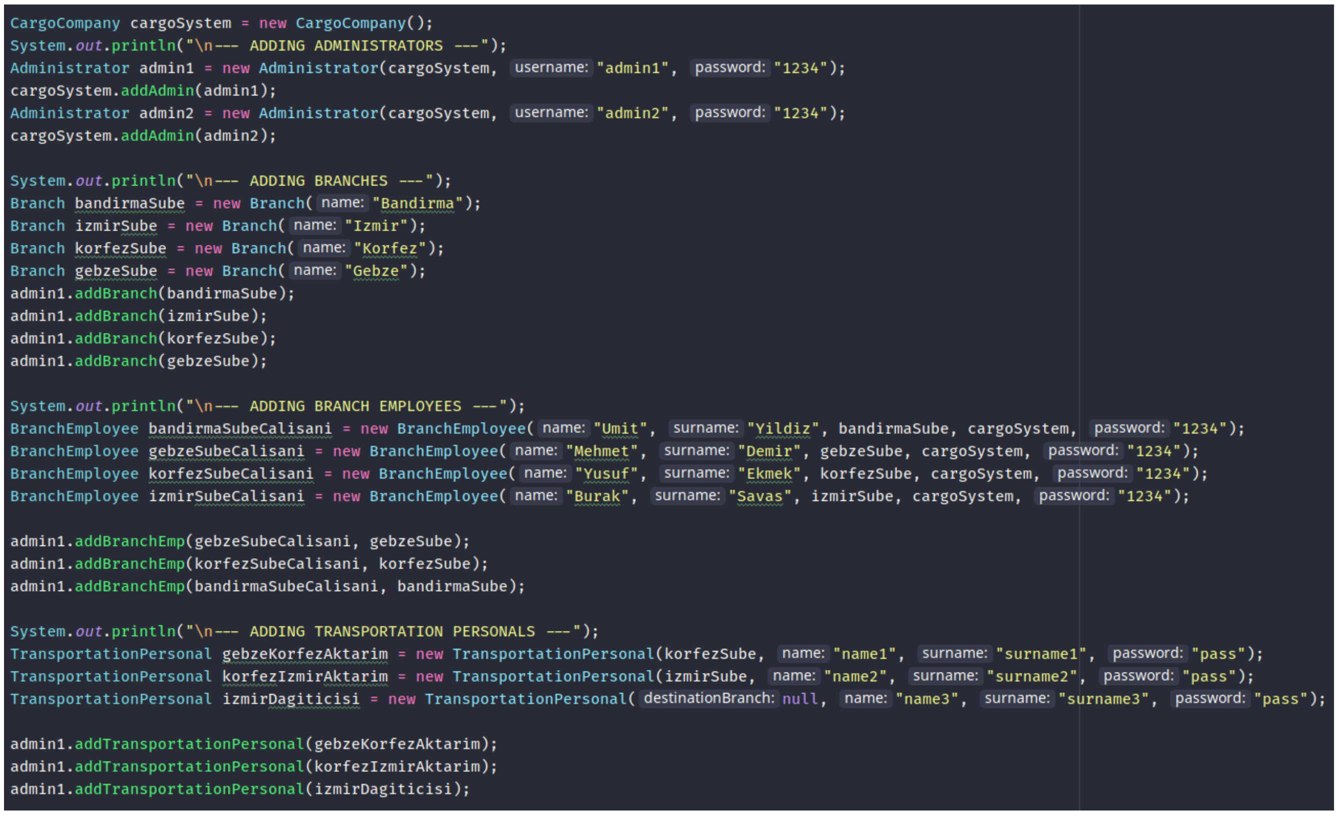
The program must be secure in terms of storing user data. So, I store passwords as encrypted instead of storing them as raw text by concerns of security. I use the "SHA-256" encryption algorithm in java.security to store passwords.

I encountered accessing problems from some classes to others, but I figured it out by creating another object as a field of the first class. And it makes sense semantically. (workingBranch in BranchEmployee, responsibleBranch in Shipment)

Another issue was creating unique tracking numbers. I used Integer data type to store tracking numbers, and also used static tracking number counter value to keep track of the latest tracking number. I also update my algorithm. When I need to find a shipment by its tracking number I search through array in reverse order, because latest cargos will be inquired often compare to older ones.

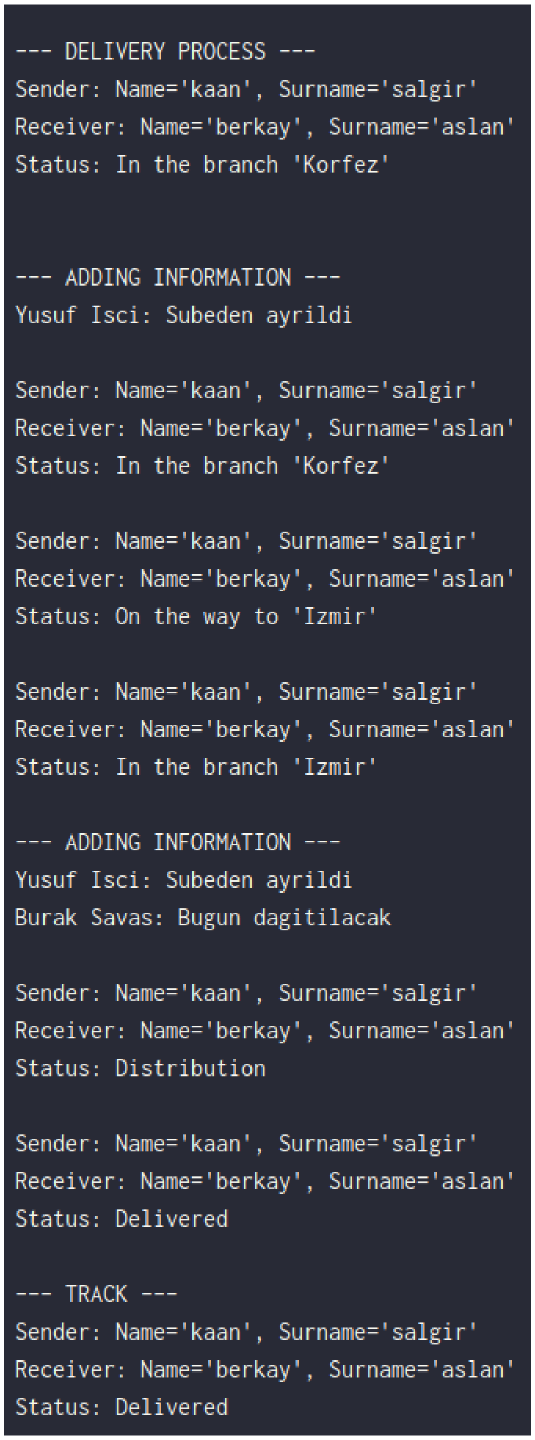
5. TEST CASES

I tested basic functions of the program in the beginning of main function;



```
CargoCompany cargoSystem = new CargoCompany();
System.out.println("\n--- ADDING ADMINISTRATORS ---");
Administrator admin1 = new Administrator(cargoSystem, username: "admin1", password: "1234");
cargoSystem.addAdmin(admin1);
Administrator admin2 = new Administrator(cargoSystem, username: "admin2", password: "1234");
cargoSystem.addAdmin(admin2);

System.out.println("\n--- ADDING BRANCHES ---");
Branch bandirmaSube = new Branch( name: "Bandirma");
Branch izmirSube = new Branch( name: "Izmir");
Branch korfezSube = new Branch( name: "Korfez");
Branch gebzeSube = new Branch( name: "Gebze");
admin1.addBranch(bandirmaSube);
admin1.addBranch(izmirSube);
admin1.addBranch(korfezSube);
admin1.addBranch(gebzeSube);

System.out.println("\n--- ADDING BRANCH EMPLOYEES ---");
BranchEmployee bandirmaSubeCalisani = new BranchEmployee( name: "Umit", surname: "Yildiz", bandirmaSube, cargoSystem, password: "1234");
BranchEmployee gebzeSubeCalisani = new BranchEmployee( name: "Mehmet", surname: "Demir", gebzeSube, cargoSystem, password: "1234");
BranchEmployee korfezSubeCalisani = new BranchEmployee( name: "Yusuf", surname: "Ekmek", korfezSube, cargoSystem, password: "1234");
BranchEmployee izmirSubeCalisani = new BranchEmployee( name: "Burak", surname: "Savas", izmirSube, cargoSystem, password: "1234");

admin1.addBranchEmp(gebzeSubeCalisani, gebzeSube);
admin1.addBranchEmp(korfezSubeCalisani, korfezSube);
admin1.addBranchEmp(bandirmaSubeCalisani, bandirmaSube);

System.out.println("\n--- ADDING TRANSPORTATION PERSONALS ---");
TransportationPersonal gebzeKorfezAktarim = new TransportationPersonal(korfezSube, name: "name1", surname: "surname1", password: "pass");
TransportationPersonal korfezIzmirAktarim = new TransportationPersonal(izmirSube, name: "name2", surname: "surname2", password: "pass");
TransportationPersonal izmirDagiticisi = new TransportationPersonal( destinationBranch: null, name: "name3", surname: "surname3", password: "pass");

admin1.addTransportationPersonal(gebzeKorfezAktarim);
admin1.addTransportationPersonal(korfezIzmirAktarim);
admin1.addTransportationPersonal(izmirDagiticisi);
```

Terminal output of tested methods while the shipment is delivering;

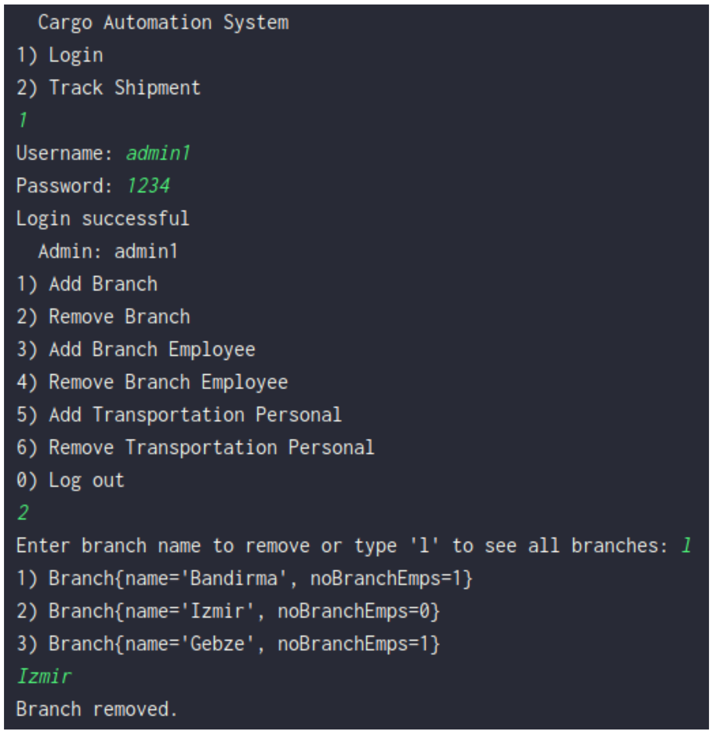


```
--- DELIVERY PROCESS ---
Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: In the branch 'Korfez'


--- ADDING INFORMATION ---
Yusuf Isci: Subeden ayrildi

Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: In the branch 'Korfez'

Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: On the way to 'Izmir'

Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: In the branch 'Izmir'

--- ADDING INFORMATION ---
Yusuf Isci: Subeden ayrildi
Burak Savas: Bugun dagitilacak

Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: Distribution

Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: Delivered

--- TRACK ---
Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: Delivered
```

6. RUNNING AND RESULTS

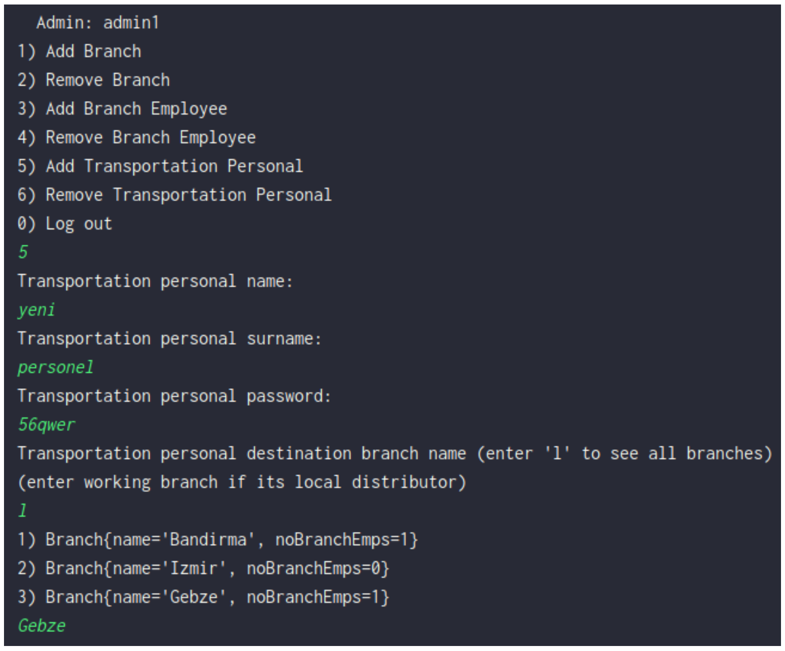
Caputred screenshots of terminal output of different operations;

Removing branch by administrator:



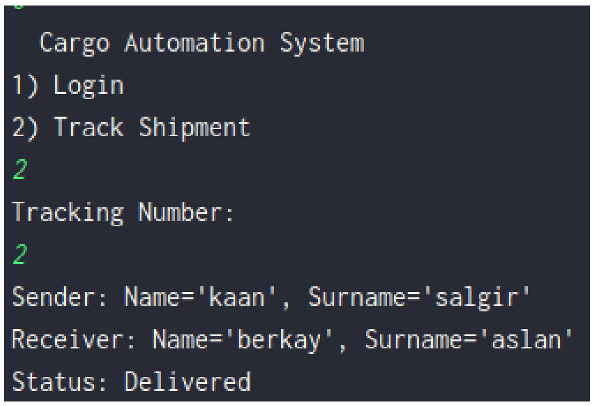
```
  Cargo Automation System
1) Login
2) Track Shipment
1
Username: admin1
Password: 1234
Login successful
  Admin: admin1
1) Add Branch
2) Remove Branch
3) Add Branch Employee
4) Remove Branch Employee
5) Add Transportation Personal
6) Remove Transportation Personal
0) Log out
2
Enter branch name to remove or type 'l' to see all branches: l
1) Branch{name='Bandirma', noBranchEmps=1}
2) Branch{name='Izmir', noBranchEmps=0}
3) Branch{name='Gebze', noBranchEmps=1}
Izmir
Branch removed.
```

Adding new transportation personal:

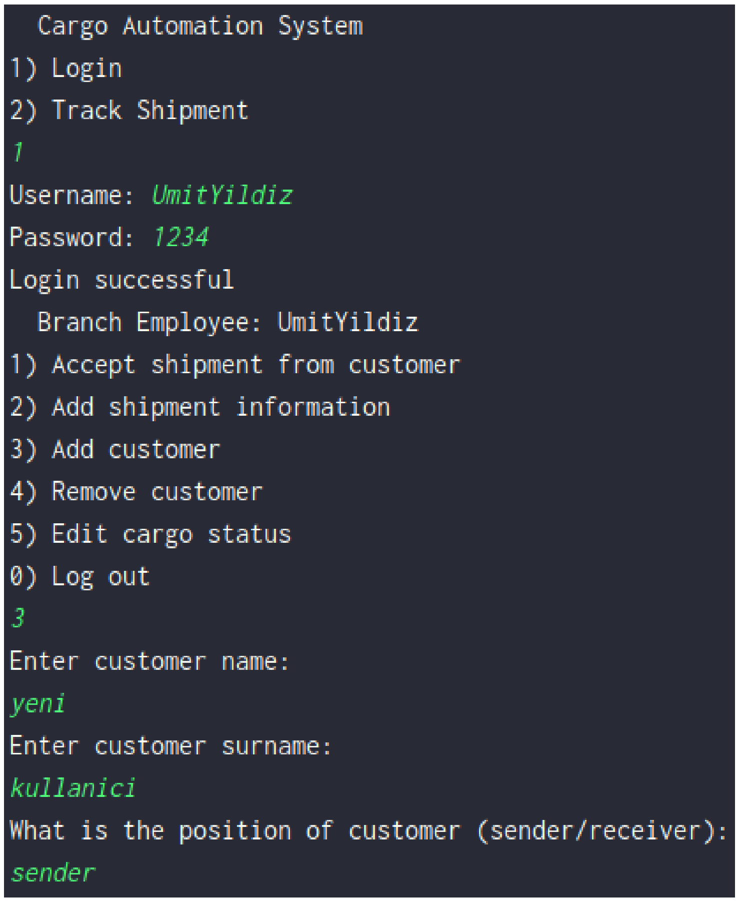


```
  Admin: admin1
1) Add Branch
2) Remove Branch
3) Add Branch Employee
4) Remove Branch Employee
5) Add Transportation Personal
6) Remove Transportation Personal
0) Log out
5
Transportation personal name:
yeni
Transportation personal surname:
personel
Transportation personal password:
56qwer
Transportation personal destination branch name (enter 'l' to see all branches)
(enter working branch if its local distributor)
l
1) Branch{name='Bandirma', noBranchEmps=1}
2) Branch{name='Izmir', noBranchEmps=0}
3) Branch{name='Gebze', noBranchEmps=1}
Gebze
```

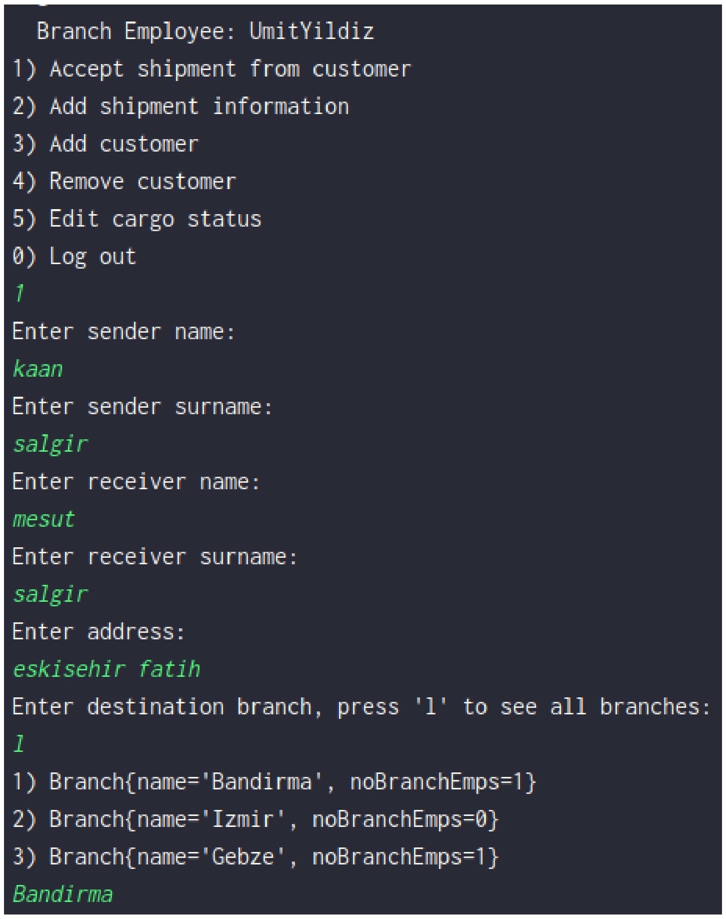
Track shipment:


```
  Cargo Automation System
1) Login
2) Track Shipment
2
Tracking Number:
2
Sender: Name='kaan', Surname='salgir'
Receiver: Name='berkay', Surname='aslan'
Status: Delivered
```

Branch employee adds new customer:


```
  Cargo Automation System
1) Login
2) Track Shipment
1
Username: UmitYildiz
Password: 1234
Login successful
  Branch Employee: UmitYildiz
1) Accept shipment from customer
2) Add shipment information
3) Add customer
4) Remove customer
5) Edit cargo status
0) Log out
3
Enter customer name:
yeni
Enter customer surname:
kullanici
What is the position of customer (sender/receiver):
sender
```

Branch employee adds new shipment;



```
  Branch Employee: UmitYildiz
1) Accept shipment from customer
2) Add shipment information
3) Add customer
4) Remove customer
5) Edit cargo status
0) Log out
1
Enter sender name:
kaan
Enter sender surname:
salgir
Enter receiver name:
mesut
Enter receiver surname:
salgir
Enter address:
eskisehir fatih
Enter destination branch, press 'l' to see all branches:
l
1) Branch{name='Bandirma', noBranchEmps=1}
2) Branch{name='Izmir', noBranchEmps=0}
3) Branch{name='Gebze', noBranchEmps=1}
Bandirma
```