GTU Department of Computer Engineering

CSE 222- Spring 2020

Homework 1 Part 2 Report

Fatih OĞUZ

151044025

SYSTEM REQUIREMENTS

Company System has knowledge of it .(It is administrator, branch employee, branch, customer, and transportation personnel).

Company adds and removes administrator.

Administrator manages this system. İt adds and removes branch employee, branch, and transportation personnel.

Branch has shipments.

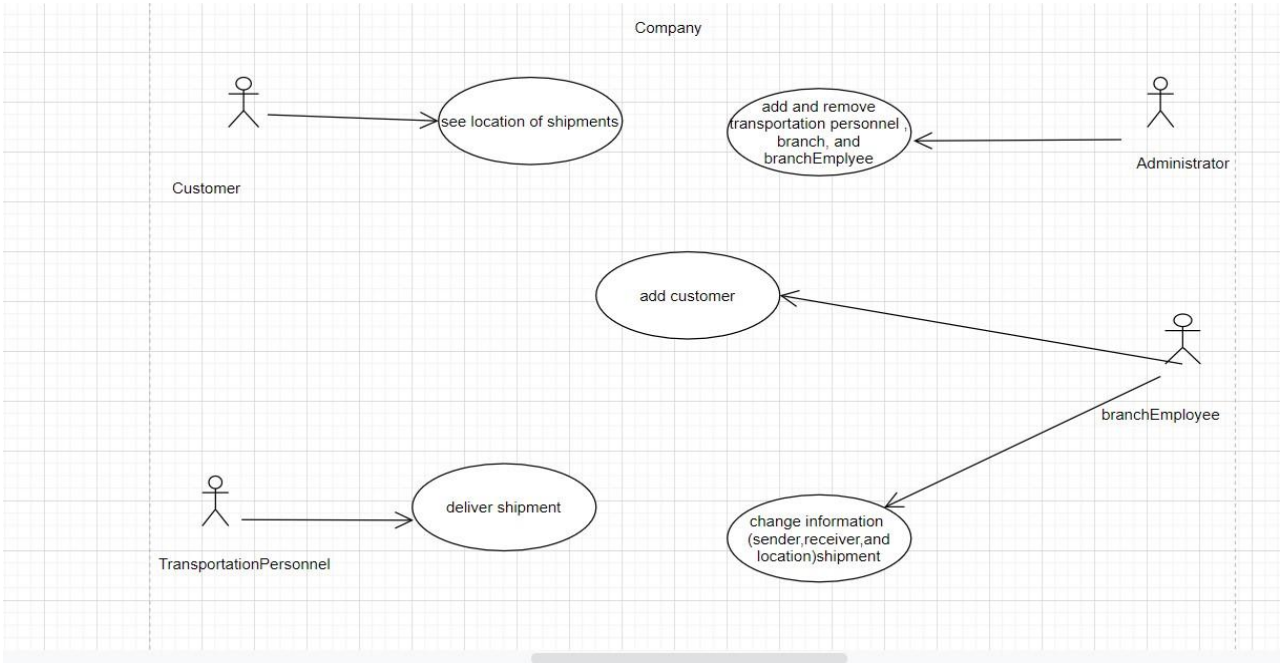
Branch employee manages branch. Branch employee adds and removes customer in system.

Branch employee controls information of shipments when necessary.

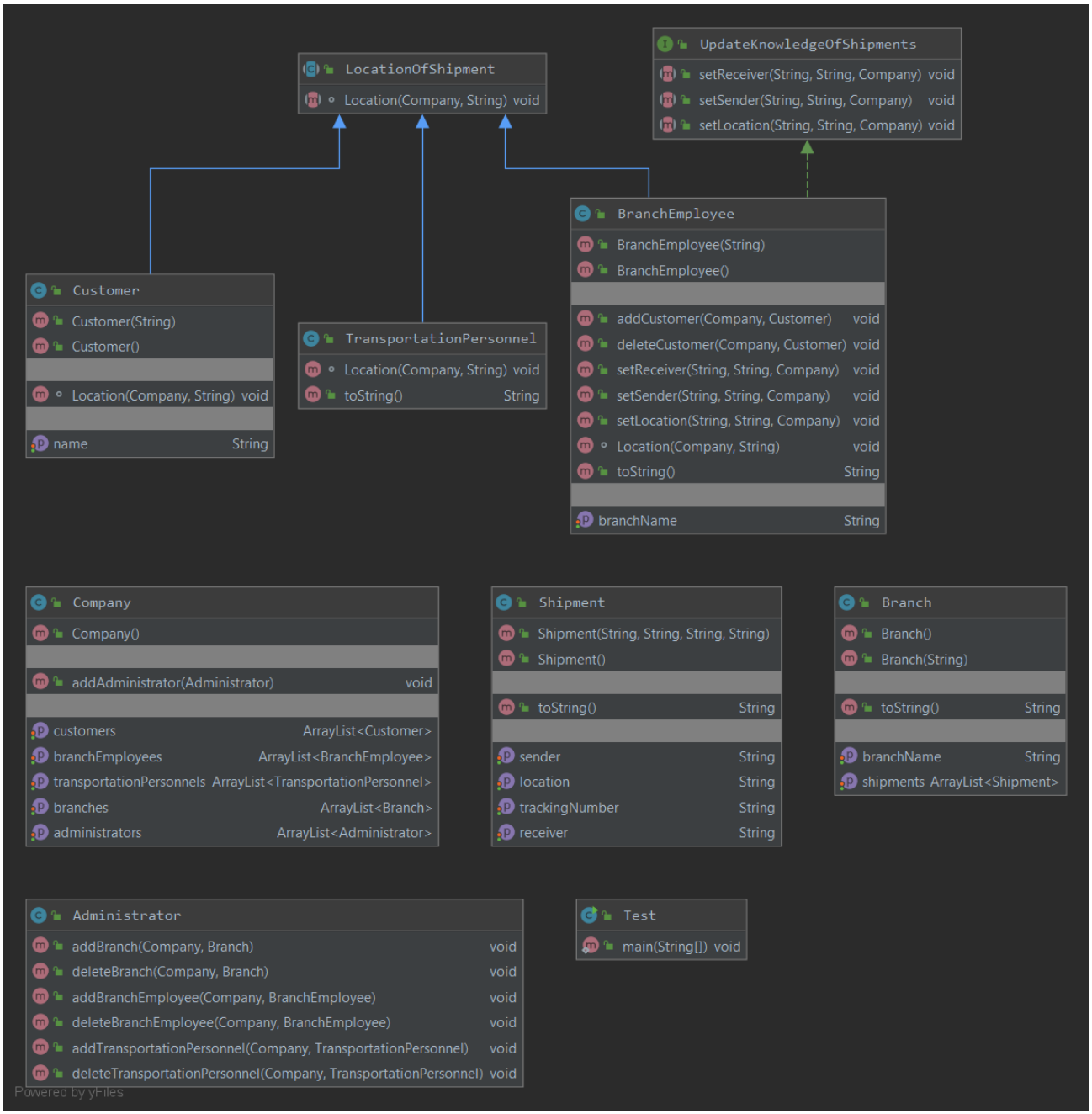
Transportation personnel delivers shipment to customer. Informs the system when delivery takes place.

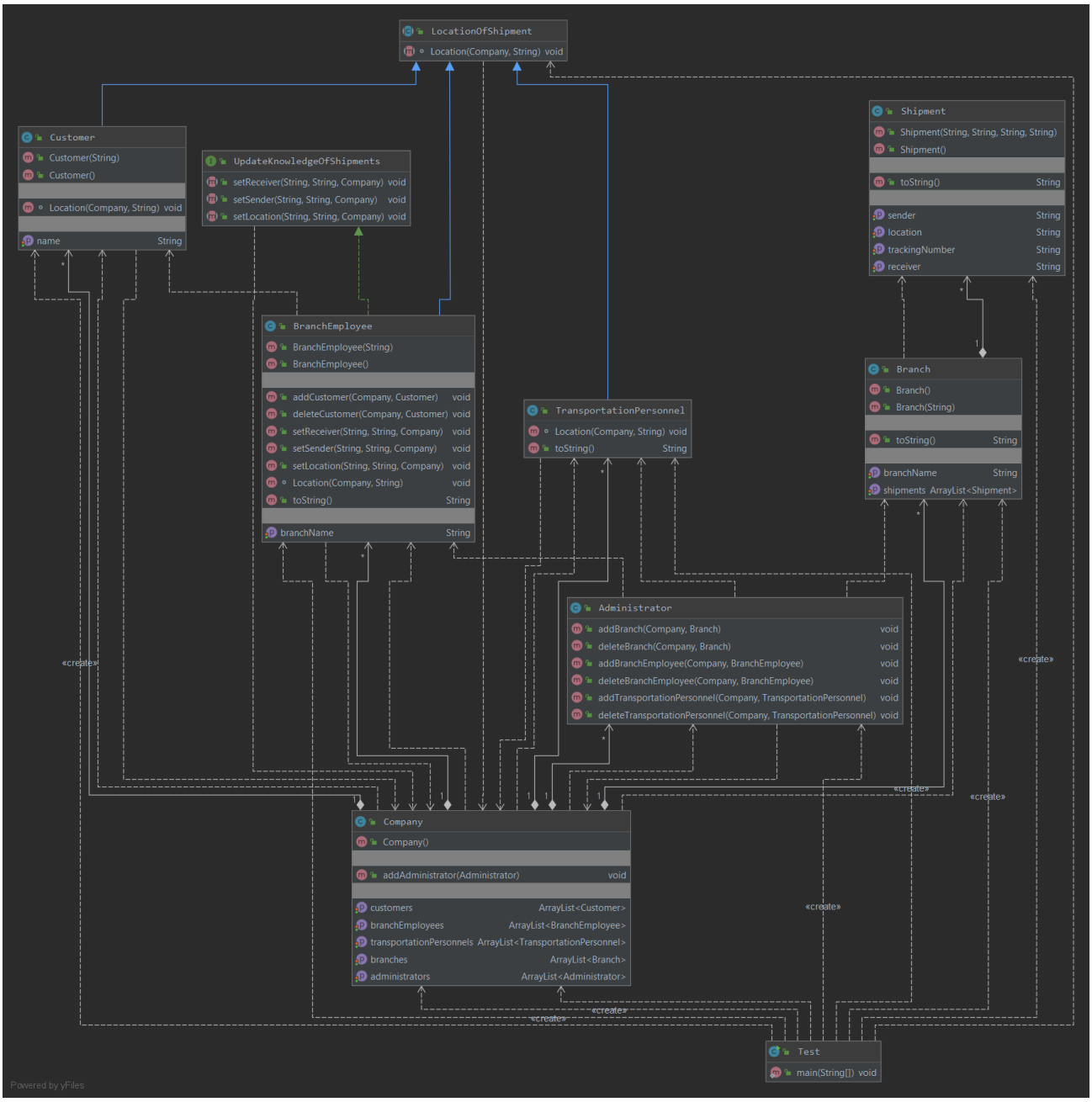
Customer can sees when customer want to see location of shipment.

USE CASE DIAGRAM



CLASS DIAGRAMS





PROBLEM SOLUTION APPROACH

Subproblems

1-) Manage (add and remove)

Solution:

Adding and removing classes can be done easily with the arraylist. do arraylist each of class.

2-) Branch employee controls information of shipments when necessary.

Solution:

reaching the relevant shipment in the system, change the relevant data with data of parameter via tracking number.(tracking number is unique number for shipment)

3-) Transportation personnel delivers shipment to customer. Informs the system when delivery takes place.

Solution:

Transportation personnel reachig relevant shipment in the system , change the place of shipment to delivered when it delivers to receiver. İt reaching via tracking number .

4-) Customer can sees when customer want to see location of shipment.

Solution:

Customer reaches shipment via tracking number in system.

Can write an abstract class . it has abstract method for each shipment in system.

Can write interface that brach employee set knowledge of shipment (knowledge receiver , sender, location and tracking number necessary)

TEST CASES

Company , Administrator, BranchEmployee, Branch adds data via ArrayList

BranchEmployee adds and removes Customer via ArrayList.

LocationOfShipment is abstract class. This class has abstract methot.This method's name Location. BranchEmployee , Customer and TransportationPersonnel extends this class. Can arrayList of this class do polymorphism via Location method?

UpdateKnowledgeOfShipment is interface. BranchEmployee implements this interface .

Do methods of this interface works correctly?

RUNNING AND RESULTS

Company , Administrator, BranchEmployee, Branch adds data via ArrayList

BranchEmployee adds and removes Customer via ArrayList.

Test code 1

```
import java.util.ArrayList;
public class Test {
    public static void main(String [] args) throws Exception {
          Company company = new Company();

        /**
         *  array list do adding  and removing
         */
        company.addAdministrator(new Administrator());
            company.getAdministrators().get(0).addBranch(company,new Branch( branchName: "branchA"));
          System.out.println( company.getBranches().get(0));
          company.getAdministrators().get(0).addBranchEmployee(company,new BranchEmployee( branchName: "branchA"));
          System.out.println(company.getBranchEmployees().get(0));
         company.getAdministrators().get(0).addBranchEmployee(company,new BranchEmployee( branchName: "branchB"));
          System.out.println(company.getBranchEmployees().get(1));
          company.getAdministrators().get(0).addBranchEmployee(company,new BranchEmployee( branchName: "branchC"));
           System.out.println(company.getBranchEmployees().get(2));
          company.getAdministrators().get(0).addTransportationPersonnel(company,new TransportationPersonnel());
         company.getAdministrators().get(0).addTransportationPersonnel(company,new TransportationPersonnel());
          System.out.println(company.getTransportationPersonnels().get(0));

          company.getBranches().get(0).getShipments().add(new Shipment( sender: "fatih oguz ", receiver: "mesut oguz",  location: "branchA", trackingNu
          company.getBranches().get(0).getShipments().add(new Shipment( sender: "mesut oguz", receiver: "ozgun tuncbilek", location: "branchB", trackin
           System.out.println(company.getBranches().get(0).getShipments());

           company.getBranchEmployees().get(0).addCustomer(company,new Customer(company.getBranches().get(0).getShipments().get(0).getSender(
          company.getBranchEmployees().get(0).addCustomer(company,new Customer(company.getBranches().get(0).getShipments().get(0).getReceiver
          company.getBranchEmployees().get(0).addCustomer(company , new Customer());
        /**
```

Result of test code 1

```
BranchEmployee{branchName='branchB'}
BranchEmployee{branchName='branchC'}
TransportationPersonnel{}
[Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}, Shipment{Sender='mesut oguz', Receiver='ozgun tuncbilek', Location='branchB', TrackingNumber='2362'
customer fatih oguz locationShipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}
customer mesut oguz location Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}
This customer cannot access (neither the receiver nor the sender)
Branch Employee Location: Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchC', TrackingNumber='2323'}
Transportation Personnel Location Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='Delivered', TrackingNumber='2323'}
```

Delete

```
company.getAdministrators().get(0).deleteBranch(company,company.getBranches().get(0));

company.getAdministrators().get(0).addBranch(company,new Branch( branchName: "branchA"));
System.out.println( company.getBranches().get(0));
```

```
deleted branch Branch{shipments=[Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}, Shipment{Sender='mesut oguz', Receiver='ozgun tuncbilek', Location=
deleted branch employee BranchEmployee{branchName='branchA'}
deleted shipments Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}
Branch{shipments=[], branchName='branchA'}
deleted branch employee BranchEmployee{branchName='branchB'}
there is no such branch employee
```

LocationOfShipment is abstract class. This class has abstract methot.This method's name Location. BranchEmployee , Customer and TransportationPersonnel extends this class. Can arrayList of this class do polymorphism via Location method?

Test code 2

```
 * @param company is company system . It has administrator,branch,branchEmployee,TransportationPersonnel,Customer in it.
 * @param TrackingNumber is unique number for shipments
 *  location is abstract method. Customer , Transportation Personnel , and BranchEmployee extends LocationOfShipment abstract class .
 *   these three class use polymorphism with  in  test class (abstract method Location)
 */
@Override
void Location(Company company, String TrackingNumber) {

        for (int j = 0; j < company.getBranches().size(); j++) {
            for (int k = 0; k < company.getBranches().get(j).getShipments().size(); k++) {
                if (company.getBranches().get(j).getShipments().get(k).getTrackingNumber() == TrackingNumber) {
                    if(company.getBranches().get(j).getShipments().get(k).getSender()== this.getName())
                    {
                        System.out.println("customer " + this.getName() +" location :"+  company.getBranches().get(j).getShipments().get(k
                    }
                    else if(company.getBranches().get(j).getShipments().get(k).getReceiver()== this.getName())
                    {
                        System.out.println("customer " + this.getName()+  " location :" + company.getBranches().get(j).getShipments().get(
                    }
                    else
                    {
                        System.out.println("This customer cannot access (neither the receiver nor the sender)");
                    }

                }
            }
        }
    }
}
```

```
/**
 *
 * @param company is company system . It has administrator,branch,branchEmployee,TransportationPersonnel,Customer in it.
 * @param TrackingNumber is unique number for shipment
 */

@Override
void Location(Company company, String TrackingNumber) {

        for (int j = 0; j < company.getBranches().size(); j++) {
            for (int k = 0; k < company.getBranches().get(j).getShipments().size(); k++) {
                if (company.getBranches().get(j).getShipments().get(k).getTrackingNumber() == TrackingNumber) {
                    company.getBranches().get(j).getShipments().get(k).setLocation(this.branchName);
                    System.out.println("Branch Employee Location: " +  company.getBranches().get(j).getShipments().get(k));
                }

            }
        }
    }
```

```java
public class TransportationPersonnel extends LocationOfShipment{
    /**
     *
     * @param company is company system . It has administrator,branch,branchEmployee,TransportationPersonnel,Customer in it.
     * @param TrackingNumber is unique number for shipment
     * this method abstract. and this class , branchEmployee, and  customer extends Locationofshipment abstract class.
     *  this method says  delivered . change to place of shipment
     */
    @Override
    void Location(Company company, String TrackingNumber) {

            for (int j = 0; j < company.getBranches().size(); j++) {
                for (int k = 0; k < company.getBranches().get(j).getShipments().size(); k++) {
                    if (company.getBranches().get(j).getShipments().get(k).getTrackingNumber() == TrackingNumber) {
                        company.getBranches().get(j).getShipments().get(k).setLocation("Delivered");
                        System.out.println("Transportation Personnel Location: " + company.getBranches().get(j).getShipments().get(k));
                    }
                }
            }
        }
```

```java
    /**
     * polymorphism: Location method in abstract class of LocationOfShipment
     * customer class,branchEmployee class ,TransportationPersonnel class can use
     */
    /**
     * Location method of customer :only sender and receiver can reach
     */

        ArrayList <LocationOfShipment> c = new ArrayList<~>();
        c.add(company.getCustomers().get(0));
        c.get(0).Location(company, TrackingNumber: "2323");
        c.add( company.getCustomers().get(1));
        c.get(1).Location(company, TrackingNumber: "2323");
        c.add( company.getCustomers().get(2));
        c.get(2).Location(company, TrackingNumber: "2323");
        c.add( company.getBranchEmployees().get(2));
        c.get(3).Location(company, TrackingNumber: "2323");
        c.add(company.getTransportationPersonnels().get(0));
        c.get(4).Location(company, TrackingNumber: "2323");
    /**
     * It can be done as in the comments line.
     * array list of abstract class save .get using
     */

    //company.getCustomers().get(0).Location(company,"2323");
          //company.getCustomers().get(1).Location(company,"2323");
          //company.getCustomers().get(2).Location(company,"2323");
          //company.getBranchEmployees().get(2).Location(company,"2323");
          //company.getTransportationPersonnels().get(0).Location(company,"2323");
```

If polymorphism didn't exist, it would do as comment

Result of test code 2

```
customer fatih oguz  location Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}
customer mesut oguz location Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchA', TrackingNumber='2323'}
This customer cannot access (neither the receiver nor the sender)
Branch Employee Location: Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='branchC', TrackingNumber='2323'}
Transportation Personnel Location Shipment{Sender='fatih oguz ', Receiver='mesut oguz', Location='Delivered', TrackingNumber='2323'}
```

UpdateKnowledgeOfShipment is interface. BranchEmployee implements this interface .

Do methods of this interface works correctly?

Test code 3

```
    /**
     * interface methods
     */
    company.getBranchEmployees().get(0).setReceiver( receiver: "atilla hun", TrackingNumber: "2323",company);
    company.getBranchEmployees().get(0).setSender( sender: "kubilay metehan tac", TrackingNumber: "2362",company);
    company.getBranchEmployees().get(0).setLocation( location: "ISTANBUL", TrackingNumber: "2323",company);
}
```

```
public interface UpdateKnowledgeOfShipments {
    /**
     * this interface implements BranchEmployee
     * branchemployee change information of shipments via method
     */
    /**
     * @param receiver is name of receiver
     * @param TrackingNumber is unique number for shipment
     * @param company is company system . It has administrator,branch,branchEmployee,TransportationPersonnel,Customer in it.
     */
    void setReceiver(String receiver,String TrackingNumber, Company company);
    /**
     *
     * @param sender is name of sender
     * @param TrackingNumber is unique number for shipment
     * @param company is company system . It has administrator,branch,branchEmployee,TransportationPersonnel,Customer in it.
     */
    void setSender(String sender,String TrackingNumber,Company company);
    /**
     *
     * @param location is location  of shipment
     * @param TrackingNumber is unique number for shipment
     * @param company is company system . It has administrator,branch,branchEmployee,TransportationPersonnel,Customer in it.
     */
    void setLocation(String location,String TrackingNumber,Company company);
}
```

```
public class BranchEmployee extends LocationOfShipment implements UpdateKnowledgeOfShipments {
private String branchName;
```

Result of test code 3

```
Shipment{Sender='fatih oguz ', Receiver='atilla hun', Location='Delivered', TrackingNumber='2323'}
Shipment{Sender='kubilay metehan tac', Receiver='ozgun tuncbilek', Location='branchB', TrackingNumber='2362'}
Shipment{Sender='fatih oguz ', Receiver='atilla hun', Location='ISTANBUL', TrackingNumber='2323'}
```