CMP-4008Y Coursework 2 - Toll Road

100392753 (afn22ehu)

Sat, 6 May 2023 13:31

Contents

TollRoadMain.java 2

Vehicle.java 5

Car.java 6

Van.java 7

Truck.java 8

CustomerAccount.java 9

TollRoad.java 12

InsufficientAccountBalanceException.java 14

CustomerNotFoundException.java 15

Application 16

Compiler Invocation 16
Compiler Messages 16
Application Invocation 16
Messages to STDOUT 16
Messages to STDERR 17

TollRoadMain.java

```java
import java.io.*;
import java.util.LinkedList;
import java.util.Scanner;

public class TollRoadMain {

    //method is used to read the txt file and assign the customers into the list link by the variables that are
    //created in CustomerAccount class, the method uses scanner for easy split of the document. The method also
    //uses switch to find what type of vehicle customer owns.
    public static TollRoad initialiseTollRoadFromFile() throws FileNotFoundException {

        TollRoad myTollRoad = new TollRoad();
        File inputFile = new File ("customerData.txt");
        Scanner scan = new Scanner(inputFile);
        scan.useDelimiter("#");
        Scanner lineScan;
        LinkedList<CustomerAccount> customerAccountsData= new LinkedList<>();
        String firstName, lastName, vehicleRegistration, manufactureOfVehicle, typeOfVehicle;
        double accountBalance;
        CustomerAccount.Discount discounts;
        int vehicleLimits;

        while(scan.hasNext()){
            lineScan= new Scanner(scan.next());
            lineScan.useDelimiter(",");

                typeOfVehicle=lineScan.next();
                vehicleRegistration=lineScan.next();
                firstName=lineScan.next();
                lastName=lineScan.next();
                manufactureOfVehicle=lineScan.next();
                vehicleLimits=Integer.parseInt(lineScan.next());
                accountBalance=Double.parseDouble(lineScan.next());
                discounts= CustomerAccount.Discount.valueOf(lineScan.next());

            switch (typeOfVehicle) {
                case "Van" : {
                    Vehicle car = new Van(vehicleRegistration, manufactureOfVehicle, vehicleLimits);
                    CustomerAccount firstCustomer = new CustomerAccount(firstName, lastName, accountBalance, car,discounts);
                    customerAccountsData.add(firstCustomer);
                    break;
                }
                case "Car" : {
                    Vehicle car = new Car(vehicleRegistration, manufactureOfVehicle, vehicleLimits);
                    CustomerAccount firstCustomer = new CustomerAccount(firstName, lastName, accountBalance, car,discounts);
                    customerAccountsData.add(firstCustomer);
                    break;
                }
                case "Truck" : {
                    Vehicle car = new Truck(vehicleRegistration, manufactureOfVehicle, vehicleLimits);
                    CustomerAccount firstCustomer = new CustomerAccount(firstName, lastName, accountBalance, car,discounts);
                    customerAccountsData.add(firstCustomer);
                    break;
                }
            }

        }

        myTollRoad.setCustomerAccountData(customerAccountsData);
        return myTollRoad;
    }

    //method is used to read the file and execute two different methods, the file is
        read and split with use of scanner
    //while splitting the method finds which of the method should be called add
        founds or make trip, it will be then
    //call the right method and run a try though it, if there is not enough funds to
        make the tip the exception will be thrown
    //if the customer doesn't exist the exception also will be thrown. In successful
        tries the methods will run correctly

    public void simulateFromFile(TollRoad road) throws FileNotFoundException {
        File inputFile2= new File("transactions.txt");
        Scanner scan2= new Scanner(inputFile2);
        scan2.useDelimiter("\\$");
        Scanner lineScan2;
        String whichMethod, vehicleRegistration;
        int amount;

        while(scan2.hasNext()){
            lineScan2= new Scanner(scan2.next());
            lineScan2.useDelimiter(",");
            whichMethod=lineScan2.next();

            if(whichMethod.equals("addFunds")){
                vehicleRegistration=lineScan2.next();
                amount=Integer.parseInt(lineScan2.next());
                try {
                    CustomerAccount person = road.findCustomer(vehicleRegistration);
                    int indexToFind = road.getCustomerAccountData().indexOf(person);
                    road.getCustomerAccountData().get(indexToFind).addFunds(amount);
                    System.out.println(vehicleRegistration + ": " + amount + " amount
                         added successfully ");
                }
                catch(CustomerNotFoundException e){
                    System.out.println(vehicleRegistration + ": " + " add Funds
                        failed. CustomerAccount does not exist");
                }
            }
            else if(whichMethod.equals("makeTrip")){
                vehicleRegistration=lineScan2.next();

                try{
                    road.chargeCustomer(vehicleRegistration);
                    System.out.println(vehicleRegistration + ": " + " Trip completed
                        successfully" + road.getMoneyMade() );
                }
                catch(CustomerNotFoundException er){
                    System.out.println(vehicleRegistration +": " + " makeTrip failed.
                         CustomerAccount does not exist");
                }
                catch(InsufficientAccountBalanceException e){
                    System.out.println(vehicleRegistration + ": " + "makeTrip failed.
                         Insufficient funds");
                    }
                }
            }


        }

        public static void main(String[] args) throws FileNotFoundException,
            InsufficientAccountBalanceException, CustomerNotFoundException {

                TollRoad file1 = initialiseTollRoadFromFile();
                TollRoadMain mainRoad = new TollRoadMain();
                System.out.println(file1);
                mainRoad.simulateFromFile(file1);
                System.out.println("total of the money made : " + file1.getMoneyMade());
}}
```

Vehicle.java

```java
//Base class for the Vehicle type, aka SuperClass. This class will work as our core for other vehicle type so we don't
//need type extra data to every sub-class. We running here our base toString method and calculateBasicTripCost.
public abstract class Vehicle {

    protected String vehicleRegistration;
    protected String manufactureOfVehicle;

    public Vehicle (String vehicleRegistration, String manufactureOfVehicle){
        this.vehicleRegistration=vehicleRegistration;
        this.manufactureOfVehicle=manufactureOfVehicle;
    }
    public String getVehicleRegistration() {return this.vehicleRegistration;}

    public String getManufactureOfVehicle() {return this.manufactureOfVehicle;}

    public String toString() { return "Vehicle registration : " + vehicleRegistration +
             " and manufacture of vehicle: " + manufactureOfVehicle; }

    public abstract int calculateBasicTripCost();
}
```

Car.java

```java
//Car class that represents car vehicle used to divide vehicles easier due to different payment rules and data.
//Here we need number of seats that car has to calculate the right amount the person needs to pay.
public class Car extends Vehicle {

    private int numberOfSeats;

    public Car (String vehicleRegistration, String manufactureOfVehicle, int numberOfSeats){
        super(vehicleRegistration,manufactureOfVehicle);
        this.numberOfSeats=numberOfSeats;
    }

    public int getNumberOfSeats(){return this.numberOfSeats;}

    //Overriding calculateBasicTripCost method from vehicle to calculate the right amount for customer
    //when customer has 5 or fewer seats in the car they have to pay 500p hence £5 or if they have 6 or more they will
    //have to pay 600 p hence £6
    @Override
    public int calculateBasicTripCost() {
        if (numberOfSeats <= 5){
            return 500;
        }
        else {
            return 600;
        }
    }

    @Override
    public String toString() {  return super.toString() + " Your total = "  + calculateBasicTripCost() + "p " ;}


    public static void main(String[] args) {
        Car firstCarTest= new Car("74747G", "Honda", 4);
        Car secondCarTest= new Car("ABH6J", "Toyota", 20);
        Car thirdCarTest= new Car("ALA9J", "Nisan", 5);

        System.out.println(firstCarTest);
        System.out.println(secondCarTest);
        System.out.println(thirdCarTest);
    }


}
```

Van.java

```java
//Van class that represents van vehicle used to divide vehicles easier due to
    different payment rules and variables.
//In Van class we need payload data and we will be calculating the cost of trip based
     on how heavy is the payload.
public class Van extends Vehicle{

    private int payload;

    public Van(String vehicleRegistration, String manufactureOfVehicle, int payload){
        super(vehicleRegistration, manufactureOfVehicle);
        this.payload=payload;
    }

    public int getPayload(){return this.payload;}


    //overriding calculateBasicTripCost method from Vehicle to calculate how much
        does it cost to carry certain amount of
    //payload when payload weights less or equal to 600kg it costs 500p when payload
        weights over 600kg but less
    //or equal to 800k the cost is 750p when payload is over 800kg then cost goes up
        to 1000p
    @Override
    public int calculateBasicTripCost() {
        if (payload <= 600){
            return 500;
        }
        else if (payload <= 800){
            return 750;
        }
        else {
            return 1000;
        }
    }

    @Override
    public String toString() {return super.toString() + " your total = " +
        calculateBasicTripCost() + "p";}

    public static void main(String[] args) {
     Van firstVanTest= new Van("AEE45", "Citroen", 600);
     Van secondVanTest= new Van("LALA3", "Ford", 300);
     Van thirdVanTest= new Van("ABEN32", "Opel", 800);
     Van fourthVanTest= new Van("ALEL321", "BMW", 700);
     Van fifthVanTest= new Van("AOEPA022", "Ford", 893);

        System.out.println(firstVanTest);
        System.out.println(secondVanTest);
        System.out.println(thirdVanTest);
        System.out.println(fourthVanTest);
        System.out.println(fifthVanTest);
    }

}
```

Truck.java

```java
// truck class that represents truck vehicle used to divide vehicles easier due to
    different payment rules and
//data needed. In Truck class we need number of trailers that are attached to the
    truck to be able to calculate the right
//cost of the trip the truck will take.
public class Truck extends Vehicle{

    private int numTrailers;

    public Truck(String vehicleRegistration, String manufactureOfVehicle, int
        numTrailers){
        super(vehicleRegistration, manufactureOfVehicle);
        this.numTrailers=numTrailers;
    }

    public int getNumTrailers(){return this.numTrailers;}


    //overriding calculateBasicTripCost method to adjust the price for the number of
        trailers that the truck can carry
    //if truck can carry one or less it will be charged 1250p if it carries more than
        1500p will be charged
    @Override
    public int calculateBasicTripCost() {
        if(numTrailers <= 1){
            return 1250;
        }
        else {
            return 1500;
        }
    }

    @Override
    public String toString() {return super.toString() + " your total = " +
        calculateBasicTripCost() + "p";}

    public static void main(String[] args) {
        Truck firstTruckTest= new Truck("ABABABA123", "lOD", 0);
        Truck secondTruckTest= new Truck("yayay1", "SA",1);
        Truck thirdTruckTest= new Truck ("AFJSAL13", "ororo", 2);

        System.out.println(firstTruckTest);
        System.out.println(secondTruckTest);
        System.out.println(thirdTruckTest);
    }


}
```

CustomerAccount.java

```java
//class is used to decleare the data for customer account when new customer is
    created the values from this class will
//be applied. This class will also charge customers for their joruney accordingly to
    the vehicle they own.
public class CustomerAccount implements Comparable <CustomerAccount>{

    public enum Discount{NONE, STAFF,FRIENDS_AND_FAMILY}
    private String firstName;
    private String lastName;
    private double accountBalance;
    private  Vehicle car;
    private Discount discounts;

    //I didn't fallow the specification about the constructor because I found that
        for testing this worked the best
    //and also later on in assigning the discounts from splitting the file it was
        more clear and nicer to read.
    public CustomerAccount(String firstName, String lastName, double accountBalance,
        Vehicle car, Discount discounts){
         this.firstName=firstName;
         this.lastName=lastName;
         this.accountBalance=accountBalance;
         this.car=car;
         this.discounts=discounts;


    }

    public String getFirstName(){return firstName;}
    public String getLastName(){return lastName;}
    public double getAccountBalance(){return accountBalance;}
    public Vehicle getCar (){return car;}
    public Discount getDiscount(){return discounts;}


    //method which is triggered by makeTrip method when Staff discount is applied, it
         returns a value of the trip cost from
    //the right Vehicle sub-class and then it works out 50% of the value which will
        be substructure from the total of
    // calculateBasicTripCost method from right sub-class
    public double activateStaffDiscount()
    {return Math.floor(car.calculateBasicTripCost()*50 /100);}

    //same as above but this time it gets triggered by applying Friends and family
        discount and this discount is 10%
    //so this method works out 10% of the total value from calculateBasicTripCost
        method from Vehicle sub-class
    public double activateFriendsAndFamilyDiscount()
    {return Math.floor(car.calculateBasicTripCost()*10/100);}

    //this method is to remove a discount which has been applied within account at
        fist but they might not have it anymore
    public Discount deactivateDiscount()
    {return discounts = Discount.NONE;}

    //this method allows to add more funds to the account
    public int addFunds(int amount){
        int newAccountBalance = (int) (amount + accountBalance);
        this.accountBalance=newAccountBalance;
        return  newAccountBalance;
    }

    //In this method we can apply the right trigger for the discount so we have 3
       cases of ifs and they will be triggered
    //while the customer account is made so they have right discount associated with
       them it will be triggered everytime
    //they make a trip and the ifs will call the right methods which will calculate
       te 50% or 10% off the total value
    //this value is then subtracted here and returns the total of 50% for staff or
       90% for friends and family
    public  int makeTrip() throws InsufficientAccountBalanceException {
        int total = car.calculateBasicTripCost();

        if (discounts == Discount.STAFF){
            double appliedStaffDiscount  = total - activateStaffDiscount();
            if (accountBalance-appliedStaffDiscount<0) {
                throw new InsufficientAccountBalanceException("Insufficient funds");
            }
            accountBalance=accountBalance-appliedStaffDiscount;
            return (int)  (appliedStaffDiscount);
        }
        else if (discounts == Discount.FRIENDS_AND_FAMILY){
            double appliedFAFDiscount= total- activateFriendsAndFamilyDiscount();
            if (accountBalance-appliedFAFDiscount<0) {
                throw new InsufficientAccountBalanceException("Insufficient funds");
            }
            accountBalance=accountBalance-appliedFAFDiscount;
            return (int) (appliedFAFDiscount);
        }

        if (accountBalance-total<0) {
            throw new InsufficientAccountBalanceException("Insufficient funds");
        }
        accountBalance=  accountBalance-total;
            return total;
    }
    //compare to method exists to check if different customer have the same vehicle
       registration, it will return 0 if
    //they have the same vehicle registration, -1 when they are different but
       alphabetically the registration of the
    //first vehicle is second and 1 when fist vehicle registration comes first
       alphabetically

    public int compareTo(CustomerAccount anotherCustomerAccount){
        int check= car.vehicleRegistration.compareTo(anotherCustomerAccount.car.
           vehicleRegistration);
        if (check<0){
            return 1;
        }
        else if (check>0){
            return -1;
        }
        else {
            return 0;
        }
    }

    public String toString() {
        try {
            return "Your account balance is: " + accountBalance + " After this trip
               it will be: "+
                    makeTrip();
        } catch (InsufficientAccountBalanceException e) {
            throw new RuntimeException(e);
        }
    }


    //test
    public static void main(String[] args) throws InsufficientAccountBalanceException
        {
        Vehicle firstVehicleTest= new Car("ABC123", "Opel", 3);
        Vehicle secondVehicleTest= new Car( "BDP123", "Volvo",7);
        Vehicle thirdVehicleTest = new Truck("ACD20","Kia",2);


        CustomerAccount firstCustomerTest= new CustomerAccount("Mel","Sam", 500,
                firstVehicleTest,Discount.STAFF);

        CustomerAccount secondCustomerTest = new CustomerAccount("Josh","Lai",1000,
                secondVehicleTest,Discount.NONE);

        CustomerAccount thirdCustomerTest = new CustomerAccount("Megan","Salty",1500,
                thirdVehicleTest,Discount.FRIENDS_AND_FAMILY);

        CustomerAccount fourthCustomerTest = new CustomerAccount("Student","Poor",0,
                thirdVehicleTest,Discount.FRIENDS_AND_FAMILY);



        //test if the Staff discount works
        System.out.println(firstCustomerTest);
        //test if no discount works
        System.out.println(secondCustomerTest);
        //test if F&F discount works
        System.out.println(thirdCustomerTest);
        //Insufficient funds exception test
        //System.out.println(fourthCustomerTest.makeTrip());

        //addFunds test
        System.out.println(firstCustomerTest.addFunds(100));

        //CompareTo test
        System.out.println(firstCustomerTest.compareTo(secondCustomerTest));
        System.out.println(thirdCustomerTest.compareTo(firstCustomerTest));
        System.out.println(fourthCustomerTest.compareTo(thirdCustomerTest));
    }


}
```

TollRoad.java

```java
import java.util.LinkedList;
import java.util.Objects;

//this class is used to add customers into customer account, find customers by their
    registration number and also charge them for their journey.
public class TollRoad {

    //I decided to use linked list instead of array list because it seems more
        efficient in this case, the linked list
    // is faster to use when adding or removing objects. So, in our case is more
        useful to use something faster to add
    //the customers quicker so the code can process quicker as well.
    private  LinkedList<CustomerAccount> customerAccountData= new LinkedList<>(); //
        deleted static
    private int moneyMade =0;



    public LinkedList<CustomerAccount> getCustomerAccountData(){return
        customerAccountData;}
    public int getMoneyMade(){return moneyMade;}

    public void setCustomerAccountData(LinkedList<CustomerAccount>
        customerAccountData){this.customerAccountData=customerAccountData;}

    public boolean addCustomer(CustomerAccount account){
        return customerAccountData.add(account);
    }


    //method used to find customer by registration number, when registration is given
         the method will go through the list
    //to find customer when customer is found it will return the customer number from
         the list and if the customer
    //is not found it will throw an exception
    public  CustomerAccount findCustomer(String regNum) throws
        CustomerNotFoundException{
        int customerNumber=0;
        boolean carNotFound=false;

        for (int i =0; i<customerAccountData.size(); i++) {
            if (Objects.equals(customerAccountData.get(i).getCar().
                getVehicleRegistration(), regNum)) {
                carNotFound = false;
                break;
            }
            carNotFound = true;
            customerNumber++;
        }
        if (carNotFound){
            throw new CustomerNotFoundException("Customer not found");
        }
        return customerAccountData.get(customerNumber);
    }

    public void chargeCustomer (String registrationNumber) throws
        CustomerNotFoundException, InsufficientAccountBalanceException{
            CustomerAccount p1 = findCustomer(registrationNumber);
            moneyMade += p1.makeTrip();
    }


    public static void main(String[] args) throws InsufficientAccountBalanceException
        , CustomerNotFoundException {

        TollRoad road1= new TollRoad();

        Truck firstTruck= new Truck("AF1","opel",3);
        Truck secondTruck= new Truck("BA3","opel",3);

        CustomerAccount persona1= new CustomerAccount("Yola","L",2000,firstTruck,
            CustomerAccount.Discount.NONE);
        CustomerAccount persona2= new CustomerAccount("Jhon","B",2031,secondTruck,
            CustomerAccount.Discount.NONE);
        //testing adding the customers into the tollRoad
        road1.addCustomer(persona1);
        road1.addCustomer(persona2);
        //testing the money being made in tollRoad
        road1.chargeCustomer("AF1");
        road1.chargeCustomer("BA3");
        System.out.println(road1.moneyMade);



    }


}
```

InsufficientAccountBalanceException.java

```
public class InsufficientAccountBalanceException extends Throwable {

    public InsufficientAccountBalanceException (String info){
        super(info);
    }
}
```

CustomerNotFoundException.java

```java
public class CustomerNotFoundException extends Throwable {
    public CustomerNotFoundException (String noCustomer){
        super(noCustomer);
    }
}
```

Application

Compiler Invocation

```
javac -Xlint:unchecked -Xlint:deprecation -encoding UTF-8 -d
    prepasg7574845565620477604classes TollRoadMain.java Vehicle.java Car.java Van.
    java Truck.java CustomerAccount.java TollRoad.java
    InsufficientAccountBalanceException.java CustomerNotFoundException.java
```

Compiler Messages

None.

Application Invocation

```
java TollRoadMain
```

Messages to STDOUT

```
TollRoad@d716361
OE08OQF: 900 amount added successfully
CJ65DVB: 600 amount added successfully
UJ12CRW: 900 amount added successfully
KR60XNS:  Trip completed successfully250
ZM02EUX:  Trip completed successfully925
CE02ESF:  Trip completed successfully2050
JT11YMK:  makeTrip failed. CustomerAccount does not exist
LO17FFK:  Trip completed successfully2500
GV14RRM:  Trip completed successfully3100
ZR60JGI: makeTrip failed. Insufficient funds
GP66KQL:  Trip completed successfully3600
ZR60JGI: 4000 amount added successfully
GV14RRM:  Trip completed successfully4200
QJ55HTR:  Trip completed successfully4950
HQ09WIJ:  Trip completed successfully5700
WQ10FZI:  makeTrip failed. CustomerAccount does not exist
ZR60JGI:  Trip completed successfully6150
YS06ESV:  Trip completed successfully6900
ZJ04XSG:  Trip completed successfully7400
ZJ04XSG:  Trip completed successfully7900
ZJ04XSG: 2000 amount added successfully
TR52YTU:  makeTrip failed. CustomerAccount does not exist
CJ65DVB:  Trip completed successfully8400
OG57MWO:  makeTrip failed. CustomerAccount does not exist
ZQ11PMI:  makeTrip failed. CustomerAccount does not exist
BB14UPJ:  Trip completed successfully9400
RD16GDY:  Trip completed successfully10750
IL12KYE:  Trip completed successfully11250
BQ63HWH:  makeTrip failed. CustomerAccount does not exist
ZR60JGI:  Trip completed successfully11700
QM17VBV:  makeTrip failed. CustomerAccount does not exist
ND53BHO:  Trip completed successfully12150
OD14STV:  Trip completed successfully12650
CJ65DVB:  Trip completed successfully13150
NK62WPC: makeTrip failed. Insufficient funds
HE52JEL: 500 amount added successfully
IL12KYE:  Trip completed successfully13650
HI14WRW:  Trip completed successfully14250
FL55YST:  makeTrip failed. CustomerAccount does not exist
EK02DEU:  makeTrip failed. CustomerAccount does not exist
JH67HRE:  Trip completed successfully14875
LU16KLS: 1500 amount added successfully
RD16GDY:  Trip completed successfully16225
```

```
HE52JEL:  Trip completed successfully17475
HE52JEL: 1000 amount added successfully
HE57YSO:  Trip completed successfully17975
NK62WPC: 15000 amount added successfully
CI60ESO:  makeTrip failed. CustomerAccount does not exist
ZJ04XSG:  Trip completed successfully18475
YL09PTN:  Trip completed successfully19825
VS56SYX:  Trip completed successfully20425
OD14STV:  Trip completed successfully20925
QJ55HTR: 900 amount added successfully
CE02ESF:  Trip completed successfully22050
RD16GDY: makeTrip failed. Insufficient funds
YJ55XPM: 1000 amount added successfully
LU16KLS:  Trip completed successfully22550
TY62BSA:  makeTrip failed. CustomerAccount does not exist
UJ12CRW:  Trip completed successfully23050
JH67HRE: makeTrip failed. Insufficient funds
YJ55XPM:  Trip completed successfully23550
CJ65DVB:  Trip completed successfully24050
HQ09WIJ: 1100 amount added successfully
HE57YSO:  Trip completed successfully24550
WZ14EWB:  Trip completed successfully25000
ID08GMV:  Trip completed successfully25500
ZM02EUX:  Trip completed successfully26175
OZ53PUX:  Trip completed successfully26625
QJ55HTR:  Trip completed successfully27375
ND53BHO: 2000 amount added successfully
VS56SYX:  Trip completed successfully27975
JF15BCM:  Trip completed successfully28475
QJ55HTR:  Trip completed successfully29225
ZD04UKP: makeTrip failed. Insufficient funds
ZD04UKP: 1500 amount added successfully
ZD04UKP:  Trip completed successfully30725
ZM02EUX:  Trip completed successfully31400
WZ14EWB:  Trip completed successfully31850
ES66WHE:  makeTrip failed. CustomerAccount does not exist
VB58LNG:  makeTrip failed. CustomerAccount does not exist
NK62WPC:  Trip completed successfully32350
KY09MDZ:  Trip completed successfully33350
KR60XNS:  Trip completed successfully33600
YJ55XPM:  Trip completed successfully34100
HC17NRJ:  Trip completed successfully34600
HH01VMY:  makeTrip failed. CustomerAccount does not exist
KY09MDZ: makeTrip failed. Insufficient funds
CJ65DVB:  Trip completed successfully35100
FM66JPE:  Trip completed successfully36225
NK62WPC:  Trip completed successfully36725
CB13IIZ:  Trip completed successfully37025
GV14RRM:  Trip completed successfully37625
ZM02EUX:  Trip completed successfully38300
JF15BCM: 1000 amount added successfully
JF15BCM:  Trip completed successfully38800
KY09MDZ: makeTrip failed. Insufficient funds
VS56SYX:  Trip completed successfully39400
total of the money made : 39400
```

Messages to STDERR

None.