CMP-4008Y Coursework 2 - Toll Road Program

100250071 (`fxg18asu`)

Sat, 23 Mar 2019 17:06

Contents

Vehicle.java

```java
public abstract class Vehicle {


    protected String licencePlate;
    protected String vehicleMake;


    public Vehicle(String reg, String make) {
        this.licencePlate = reg;
        this.vehicleMake = make;

    }

    public abstract int calculateBasicTripCost();

    public void setLicencePlate(String p) {
        licencePlate = p;
    }

    public void setVehicleMake(String m) {
        vehicleMake = m;
    }

    public String getLicencePlate() {
        return licencePlate;
    }

    public String getVehicleMake() {
        return vehicleMake;
    }


    public String toString() {
        return "Licence Plate:" + licencePlate + " Vehicle Make:" + vehicleMake;
    }
}
```

Car.java

```java
public class Car extends Vehicle{
    private int numberOfSeats;
    public Car(int nSeats, String reg, String make) {
        super(reg, make);
        this.numberOfSeats = nSeats;

    }

    @Override //this calculates the cost of a trip based on the vehicles
        information
    public int calculateBasicTripCost() {
        if (numberOfSeats < 6) {
            return 500;
        }
        else {
            return 600;
        }
    }

    @Override
    public String toString() {
        return "Licence Plate:" + licencePlate + " Vehicle Make:" + vehicleMake +
             " Number Of Seats:" + numberOfSeats;
    }

    public void setNumberOfSeats(int seats) {
        numberOfSeats = seats;
    }

    public int getNumberOfSeats() { return numberOfSeats; }
}
```

Van.java

```java
public class Van extends Vehicle{
    private double payload;
    public Van(double weight, String reg, String make) {
        super(reg, make);
        this.payload = weight;
    }

    @Override
    public int calculateBasicTripCost() { //this calculates the cost of a trip
       based on the vehicles information
        if (payload <= 600) {
            return 500;
        }
        else if (payload > 600 && payload <= 800) {
            return 750;
        }
        else {
            return 1000;
        }
    }

    @Override
    public String toString() {
        return "Licence Plate:" + licencePlate + " Vehicle Make:" + vehicleMake +
            " Payload(Kg):" + payload;
    }

    public void setPayload(double weight) {
        payload = weight;
    }

    public double getPayload() {
        return payload;
    }

}
```

Truck.java

```java
public class Truck extends Vehicle{
    private int numTrailers;
    public Truck(int nTrailers, String reg, String make) {
        super(reg, make);
        this.numTrailers = nTrailers;
    }

    @Override
    public int calculateBasicTripCost() { //this calculates the cost of a trip
       based on the vehicles information
        if ( numTrailers == 0 || numTrailers == 1) {
            return 1250;
        }
        else {
            return 1500;

        }
    }

    @Override
    public String toString() {
        return "Licence Plate:" + licencePlate + " Vehicle Make:" + vehicleMake +
            " Number Of Trailers: " + numTrailers;
    }

    public void setNumTrailers(int nTrailers) {
        numTrailers = nTrailers;
    }


    public int getNumTrailers() {
        return numTrailers;
    }


}
```

CustomerAccount.java

```java
public class CustomerAccount implements Comparable<CustomerAccount>{
    private String firstName;
    private String secondName;
    private Vehicle Vehicle;
    private double accountBalance;
    private enum DiscountType {NONE, STAFF, FRIENDS_AND_FAMILY}
    private DiscountType discountType;



    public CustomerAccount(String fName, String sName, Vehicle v, double aBalance) {

        this.firstName = fName; //Sets the first name of the customer account
        this.secondName = sName; //Sets the surname name of the customer account
        this.Vehicle = v; //Sets the Vehicle of the customer account
        this.accountBalance = aBalance; //Sets the balance of the customer account
        this.discountType = DiscountType.NONE; //Sets the discount type to the default of NONE

    }

    public int compareTo(CustomerAccount other) {
        return this.Vehicle.getLicencePlate().compareTo(other.Vehicle.getLicencePlate());

    }

    public void activateStaffDiscount() {
        discountType = DiscountType.STAFF;
    } //This activates the STAFF discount


    public void activateFriendsAndFamilyDiscount() { //This activates the FRIENDS_AND_FAMILY discount but only if the discount type is NONE to begin with
        if (discountType == DiscountType.NONE) {
            discountType = DiscountType.FRIENDS_AND_FAMILY;
        }
        else {
            return;
        }
    }

    public void deactivateDiscount() {
        discountType = DiscountType.NONE;
    } //This sets the discount type to NONE

    public void addFunds(int add) {
        accountBalance = accountBalance + add;
    } //this adds funds to the balance variable of the object

    public double makeTrip() { //This works out the cost of the trip based on the discount type and vehicle info
        if (discountType == DiscountType.STAFF) {
            return this.Vehicle.calculateBasicTripCost() * 0.5;
        }
        else if (discountType == DiscountType.FRIENDS_AND_FAMILY) {
            return this.Vehicle.calculateBasicTripCost() * 0.9;
        }
        else {
            return this.Vehicle.calculateBasicTripCost();
        }
    }
    public String getFirstName() { return firstName; }

    public String getSecondName() { return secondName; }

    public Vehicle getVehicle() { return Vehicle; }

    public double getAccountBalance() { return accountBalance; }

    public void setAccountBalance (double add) {
        accountBalance = accountBalance + add;
    }

    public String toString() {
        return ("Name:" + firstName + " "  + secondName + " " + Vehicle.toString
            () + " Balance:" + accountBalance + " Discount Type:" + discountType)
            ;
    }

}
```

TollRoad.java

```java
import java.util.ArrayList;

public class TollRoad {
    private ArrayList<CustomerAccount> CustomerAccount = new ArrayList<CustomerAccount>();
    private double moneyMade;

    public TollRoad(ArrayList<CustomerAccount> CA) {
        this.CustomerAccount = CA; //This sets the Customer Arraylist
        this.moneyMade = 0; //This sets the moneymade to a deafult of 0
    }

    public void addCustomer(CustomerAccount customerAccount) {
        CustomerAccount.add(customerAccount);
    } //this adds a customer account to the arraylist of customer accounts


    public CustomerAccount findCustomer(String regNo) throws CustomerNotFoundException { //this is used to find a customer based on their registration plate
        for (CustomerAccount CA : CustomerAccount) //This for loop searches through all the customer accounts for a customer account with a matching reg
            if (CA.getVehicle().getLicencePlate().equals(regNo))
                if (CA == null) {
                    throw new CustomerNotFoundException(regNo); //if the reg isnt found a CustomerNotFoundException is thrown
                }
                else {
                    return CA; //if it is found the whole customer account is returned
                }



        return null;


    }

    public void chargeCustomer(String regNo) throws InsufficientAccountBalanceException, CustomerNotFoundException { // this is used to charge the customer for a trip
            CustomerAccount found; // this stores the customer account that is returned in find customer
            double bBalance; // this stores the balance before the trip is made
            found = findCustomer(regNo); //this calls the findcustomer method with a registration plate and stores the result in the found variable
            if (found == null) { //throws CustomerNotFoundException if customer isn't found
                throw new CustomerNotFoundException(regNo);
            }
            else {
                bBalance = found.getAccountBalance(); //this works out the account balance before the transaction and stores it
                found.makeTrip(); //  this works out how much the trip is going to cost
                if (bBalance < found.makeTrip()) { // if the balance is less than the trip of the cost, then a
```

```
                    InsufficientAccountBalanceException is thrown
                    throw new InsufficientAccountBalanceException(regNo);
                }
                else {
                    found.setAccountBalance(bBalance - found.makeTrip());//this
                        deducts the trip cost from the account balance

                    moneyMade = moneyMade + found.makeTrip(); //this adds the
                        cost of the trip to the moneymade
                }
            }


    }

    public double getMoneyMade() { return moneyMade; }
}
```

CustomerNotFoundException.java

```
public class CustomerNotFoundException extends Exception {
    public CustomerNotFoundException(String reg) {
    }
}
```

InsufficientAccountBalanceException.java

```java
public class InsufficientAccountBalanceException extends Exception {
    public InsufficientAccountBalanceException(String reg) {
    }
}
```

Main.java

File not found.

Main.java

```java
import java.io.*;
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class Main {
    private String[] detail;
    private ArrayList<CustomerAccount> customerRecords = new ArrayList<>();
    private List<String> CR  = new ArrayList();
    private CustomerAccount found;
    private TollRoad road = new TollRoad(customerRecords);
    public Main() {
        customerRecords = this.initialiseTollRoadFromFile(); //this fills the
            customerrecords arraylist up with customer accounts
        simulateFromFile(road); //this processes all the transactions
        System.out.println("Money Made: " + road.getMoneyMade()); //this prints
            how much money was made
    }
    public static void main(String[] args) {
        Main main = new Main();

    }

    public ArrayList<CustomerAccount> initialiseTollRoadFromFile() {
        try {
            String record; //this stores the whole line of data from the
                customerdata txt file
            Scanner fileScan, recordScan; //filescan scans in all the text from
                the customerdata txt file, recordscan reads all the records after
                 they have been seperated up

            fileScan = new Scanner(new File("customerData.txt"));
            while (fileScan.hasNext()) {
                record = fileScan.nextLine();

                recordScan = new Scanner(record);
                recordScan.useDelimiter("#"); //this splits the records up into
                    an individual record by the #


                while (recordScan.hasNext()) {
                    detail = recordScan.next().split(","); //this splits the
                        record up by the commas

                    String type = detail[0]; //takes each bit of data and stores
                        it in the variable it corresponds to
                    String reg = detail[1];
                    String fName  = detail[2];
                    String lName  = detail[3];
                    String make  = detail[4];
                    int vDetail = Integer.valueOf(detail[5]);
                    double balance = Integer.valueOf(detail[6]);
                    String discount = detail[7];
                    int count = 0;


                    if ("Car".equals(type)) { // if type is equal to car this is
                        run
                         Car tempV = new Car(vDetail, reg, make); //this makes a
                             new car object and stores it in a temporary vehicle
                             object
                         CustomerAccount tempCA = new CustomerAccount(fName, lName
                             , tempV, balance); //this makes a new customeraccount
                              object based on the previous data
                         customerRecords.add(tempCA); //this adds the
                             customeraccount to the customerrecords arraylist
                    }
                    else if ("Van".equals(type)) { // if type is equal to van
                        this is run
                         Van tempV = new Van(vDetail, reg, make); //this makes a
                             new van object and stores it in a temporary vehicle
                             object
                         CustomerAccount tempCA = new CustomerAccount(fName, lName
                             , tempV, balance);
                         customerRecords.add(tempCA);
                    }
                    else if ("Truck".equals(type)){ // if type is equal to truck
                        this is run
                         Truck tempV = new Truck(vDetail, reg, make); //this makes
                              a new truck object and stores it in a temporary
                             vehicle object
                         CustomerAccount tempCA = new CustomerAccount(fName, lName
                             , tempV, balance);
                         customerRecords.add(tempCA);
                    }
                    if ("STAFF".equals(discount)) {
                         customerRecords.get(count).activateStaffDiscount();
                         count++;
                    }
                    else if ("FRIENDS_AND_FAMILY".equals(discount)) {
                         customerRecords.get(count).
                             activateFriendsAndFamilyDiscount();
                         count++;
                    }




                }


                System.out.println();
            }

        }
        catch(IOException ie) {
            ie.printStackTrace();

        }


        return customerRecords; // this returns the finished arraylist
    }

    public void simulateFromFile(TollRoad road) { //this method does all the
        transactions
        try {
            String record;
            Scanner fileScan, recordScan;

            fileScan = new Scanner(new File("transactions.txt"));
            while (fileScan.hasNext()) {
```

```
record = fileScan.nextLine();

recordScan = new Scanner(record);
recordScan.useDelimiter("\\$"); //this splits the records up into
    an individual record by the $

while (recordScan.hasNext()) {
    detail = recordScan.next().split(",");

    String transactionType = detail[0];
    String reg = detail[1];



    if (transactionType.equals("addFunds")) { //if
       transactiontype is equal to addfunds it the code will
       follow this path
        double amount = Integer.valueOf(detail[2]);
        try {
            found = road.findCustomer(reg);
            found.setAccountBalance(amount); //this will add the
               amount to accounts balance
            System.out.println(reg + ": " + amount + " added
               successfully."); //if it is successful this will
               be printed

        }
        catch (CustomerNotFoundException nFound) {
            System.out.println(reg + ": addFunds failed.
               CustomerAccount does not exist."); //if the
               account doesn't exist the exception thrown from
               the method will be caught here
        }

    }
    else if (transactionType.equals("makeTrip")) { //if the
       function is maketrip, then the reg will be added to this
       list and completed after all addfunds are comepleted
        CR.add(reg);

    }
}
System.out.println();
for (String reg : CR) { //this goes through every reg in the list
    and charges the customer
    try {
        road.chargeCustomer(reg); //this calls the method
           chargecustomer on the reg
        System.out.println(reg + ": Trip Completed Successfully."
           ); //if they have enough funds it will output this

    }
    catch (CustomerNotFoundException nFound) { // if the customer
        isnt found it will be caught here and this will be
       outputted
        System.out.println(reg + ": MakeTrip failed.
           CustomerAccount does not exist.");
    }
    catch (InsufficientAccountBalanceException nAmount) { //if
       the customer doesn't have enough funds the exception will
        be caught here, and this will be outputted
        System.out.println(reg + ": MakeTrip failed. Insufficient
            funds.");
```

```
                    }
                }
            }
        }
        catch(IOException ie) {
            ie.printStackTrace();

        }


    }



}
```