CS 1083

Assignment #6

Author: Yulong Wang

Id: 3713596

1. Source Code and Test Cases Part One

a.Vehicle.java

```
import java.io.Serializable;
import java.text.NumberFormat;

/**
This class represents a vehicle.
@author Leah Bidlake
*/
public abstract class Vehicle implements Comparable<Vehicle>, Serializable {

    /**
    The vehicle's code.
    */
    private String code;

    /**
    The vehicle's mileage in km.
    */
    private int mileage;

    /**
    Constructs a vehicle with a specific code and mileage.
    @param code the code of the vehicle.
    @param mileage the mileage of the vehicle.
    */
    public Vehicle(String code, int mileage){
        this.code = code;
        this.mileage = mileage;
    }

    /**
    Returns the vehicle's code.
    @return returns the vehicle's code.
    */
    public String getCode(){
        return code;
    }

    /**
    Returns the current mileage of the vehicle.
    @return returns the current mileage of the vehicle.
    */
    public int getMileage(){
        return mileage;
    }

    /**
    Increases the mileage by the distance the vehicle was driven in the test drive (in
```

```
km).
    @param distance the distance of the test drive in km.
    */
    public void testDrive(int distance){
        mileage += distance;
    }

    /**
    Sorts vehicle alphabetically by type in the order cars, trucks, vans, and each
type is
    then sorted by the calculated price in ascending order (lowest to highest price).
    @param other the vehicle being compared to this vehicle.
    @return the result of the comparison.
    */
    public int compareTo(Vehicle other){
        if(code.charAt(0) == other.code.charAt(0)){
            if(this.calculatePrice() - other.calculatePrice() > 0){
                return 1;
            }
            else{
                return -1;
            }
        }
        return this.code.compareTo(other.code);
    }

    /**
    Calculates the cost of the vehicle.
    @return the calcuated cost of the vehicle.
    */
    public abstract double calculatePrice();

    /**
    Returns a formatted textual string containing information about the vehicle
including
    code, mileage, and calculated price.
    @return textual string containing the code, mileage, and cost of the vehicle.
    */
    public String toString(){
        NumberFormat form = NumberFormat.getCurrencyInstance();
        return code + "\t" + "Mileage: " + mileage + "km" +
                "\n\tCost: " + form.format(calculatePrice());
    }

}
```

b. Car.java

```
/**
Represents a car.
@author Leah Bidlake
*/
```

```java
public class Car extends Vehicle{

    /**
    The type of trunk.
    */
    private char type;

    /**
    The number of doors the car has.
    */
    private int doors;

    /**
    Constructs a Car object.
    @param code the code for the car.
    @param mileage the initial mileage on the car.
    @param type the type of car trunk.
    @param doors the number of doors the car has.
    */
    public Car(String code, int mileage, char type, int doors) throws
InvalidVehicleException{
        super(code, mileage);
        if (type!='T' && type!='H'){
            throw new InvalidVehicleException("Wrong car type");
        }
        if (doors!=2 && doors!=4){
            throw new InvalidVehicleException("Wrong car door number");
        }
        this.type = type;
        this.doors = doors;
    }

    /**
    Calculates the cost of the vehicle.
    @return the calcuated cost of the vehicle.
    */
    public double calculatePrice(){
        double cost = 10000;
        if(doors == 4){
            cost += cost * 1.05;
        }
        if(type == 'H'){
            cost += 1000;
        }
        return cost;
    }

}
```

c. Truck.java

```java
/**
Represents a truck.
@author Leah Bidlake
*/
public class Truck extends Vehicle{

    /**
    The bed size of the truck.
    */
    private String bed;

    /**
    Create a Truck object.
    @param code the code for the truck.
    @param mileage the initial mileage on the truck.
    @param bed the size of the truck bed.
    */
    public Truck(String code, int mileage, String bed) throws InvalidVehicleException{
        super(code, mileage);
        if (!bed.equals("short") && !bed.equals("standard") && !bed.equals("long")){
            throw new InvalidVehicleException("Wrong truck bed type");
        }
        this.bed = bed;
    }

    /**
    Calculates the cost of the vehicle.
    @return the calcuated cost of the vehicle.
    */
    public double calculatePrice(){
        double cost = 50000;
        if(bed.equals("short")){
            cost = cost * 0.9;
        }
        else if(bed.equals("long")){
            cost = cost * 1.1;
        }
        return cost;
    }
}
```

d. Van.java

```java
/**
Represents a van.
@author Leah Bidlake
*/
public class Van extends Vehicle{

    /**
    Electric door closure if true, manual if false.
    */
```

```
    private boolean isElectric;

    /**Creates a Van object.
    @param code the code for the van.
    @param mileage the initial mileage on the van.
    @param isElectric the door closure type.
    */
    public Van(String code, int mileage, boolean isElectric){
        super(code, mileage);
        this.isElectric = isElectric;
    }

    /**
    Calculates the cost of the vehicle.
    @return the calcuated cost of the vehicle.
    */
    public double calculatePrice(){
        double cost = 25000;
        if(isElectric){
            cost = cost * 1.15;
        }

        return cost;
    }

}
```

e. InvalidVehicleException.java

```
public class InvalidVehicleException extends Exception{
    public InvalidVehicleException(String message) {
        super(message);
    }
}
```

f. Driver.java

```
import java.io.*;
import java.util.Scanner;
import java.text.NumberFormat;

public class Driver {
    public static void main(String[] args) {
        NumberFormat form = NumberFormat.getCurrencyInstance();
        try {
            if(args.length<2){
                throw new FileNotFoundException("Missing parameters");
            }
            File file = new File(args[0]);
            Scanner scan = new Scanner(file);

            String name = scan.nextLine();
```

```
            int size = scan.nextInt();
            scan.nextLine();

            Vehicle[] inventory = new Vehicle[size];
            int counter = 0;
            for (int i = 0; i < size; i++) {
                String line = scan.nextLine();
                Scanner sc = new Scanner(line);
                String code = sc.next();
                int mileage = sc.nextInt();
                Vehicle temp;
                if (code.charAt(0) == 'C') {
                    int doors = sc.nextInt();
                    char type = sc.next().charAt(0);
                    try {
                        temp = new Car(code, mileage, type, doors);
                    } catch (InvalidVehicleException e) {
                        System.out.println(e.getMessage());
                        continue;
                    }

                } else if (code.charAt(0) == 'T') {
                    String bed = sc.next();
                    try {
                        temp = new Truck(code, mileage, bed);
                    } catch (InvalidVehicleException e) {
                        System.out.println(e.getMessage());
                        continue;
                    }

                } else {
                    String bool = sc.next();
                    boolean isElect = false;
                    if (bool.equals("true")) {
                        isElect = true;
                    }
                    temp = new Van(code, mileage, isElect);
                }
                counter ++;
                inventory[i] = temp;
            }
            Vehicle[] temp = new Vehicle[counter] ;
            int j=0;
            for (Vehicle vehicle:inventory){
                if(vehicle!=null){
                    temp[j] = vehicle;
                    j++;
                }
            }

            Dealership dealer = new Dealership(name, temp);
```

```
            System.out.println(dealer);

            System.out.println("\nSorted Data:\n");

            Sorter<Vehicle> sort = new Sorter<Vehicle>();
            Vehicle[] copy = dealer.getInventory();

            sort.selectionSort(copy);

            System.out.println(dealer.getName());

            for (Vehicle v : copy) {
                System.out.println(v.getCode() + "\t" +
form.format(v.calculatePrice()) + "\n");
            }

            while (scan.hasNext()) {
                String searchCode = scan.nextLine();
                System.out.println("Vehicle " + ((dealer.search(searchCode) == null) ?
"not" : "") + " found");
            }
            FileOutputStream fo = new FileOutputStream(args[1]);
            ObjectOutputStream outputStream = new ObjectOutputStream(fo);
            for (Vehicle vehicle : copy) {
                outputStream.writeObject(vehicle);
            }
        }catch (FileNotFoundException e){
            System.out.println(e.getMessage());
        }catch (IOException e){
        System.out.println(e.getMessage());
        }
    }
}
```

g. Test Case

1. testcase.dat: This test case test if the program can catch and recovered from wrong car type, wrong car door number and wrong truck bed size.

```
Cars R Us
8
T4172 0 standard
C3913 1004 3 H
C3911 1000 4 B
V5532 12980 true
T4908 775 short
T4901 775 high
V5163 15 false
C3511 4152 4 T
V5532
C3917
T4908
```

Output:

```
(base) yulongwang@briannas-iphone src % java Driver testcase.dat binary2
Wrong car door number
Wrong car type
Wrong truck bed type
Cars R Us
T4172   Mileage: 0km
        Cost: 50,000.00
V5532   Mileage: 12980km
        Cost: 28,750.00
T4908   Mileage: 775km
        Cost: 45,000.00
V5163   Mileage: 15km
        Cost: 25,000.00
C3511   Mileage: 4152km
        Cost: 20,500.00
```

```
Sorted Data:

Cars R Us
C3511   20,500.00

T4908   45,000.00

T4172   50,000.00

V5163   25,000.00

V5532   28,750.00

Vehicle  found
Vehicle not found
Vehicle  found
```

2. test FileNotFoundException: This test case test if program can recover from non-exist file names. Output:

```
(base) yulongwang@briannas-iphone src % java Driver somefile.dat binary2
somefile.dat (No such file or directory)
```

3. test IOException: This test case test if program can recover from writing to a read-only file. Output:

```
(base) yulongwang@briannas-iphone src % java Driver testcase.dat binary1
Wrong car door number
Wrong car type
Wrong truck bed type
Cars R Us
T4172   Mileage: 0km
```

```
        Cost: 50,000.00
V5532   Mileage: 12980km
        Cost: 28,750.00
T4908   Mileage: 775km
        Cost: 45,000.00
V5163   Mileage: 15km
        Cost: 25,000.00
C3511   Mileage: 4152km
        Cost: 20,500.00
```

```
Sorted Data:

Cars R Us
C3511   20,500.00

T4908   45,000.00

T4172   50,000.00

V5163   25,000.00

V5532   28,750.00

Vehicle  found
Vehicle not found
Vehicle  found
binary1 (Operation not permitted)
```

2. Source Code and Test Cases Part Two

a. MergeVehicles.java

```java
import java.io.*;

/**
 * @author Yulong Wang
 * @date 2021/10/26
 */
public class MergeVehicles {
    /**
     * Read two binary file and print their vehicle objects.
     * @param args Two file names
     */
    public static void main(String[] args) {
        if(args.length!=2){
            System.out.println("Missing parameters");
            System.exit(0);
        }
```

```
        for(String fileName : args){
            try {
                FileInputStream fi = new FileInputStream(fileName);
                ObjectInputStream inputStream = new ObjectInputStream(fi);
                boolean eof = false;
                while(!eof){
                    try{
                        Vehicle temp = (Vehicle)inputStream.readObject();
                        System.out.println(temp.toString());
                    }catch (EOFException e){
                        System.out.println("Reach end of file");
                        eof = true;
                    }
                }
            }catch (FileNotFoundException e){
                System.out.println("Not able to access file.");
            }catch (ClassNotFoundException e) {
                System.out.println("Class not found");
            }catch (IOException e){
                System.out.println("Problem reading from file");
            }
        }
    }
}
```

b. Sample Output:

```
(base) yulongwang@briannas-iphone src % java MergeVehicles binary1 binary2
C3511   Mileage: 4152km
        Cost: 20,500.00
T4908   Mileage: 775km
        Cost: 45,000.00
T4172   Mileage: 0km
        Cost: 50,000.00
V5163   Mileage: 15km
        Cost: 25,000.00
V5532   Mileage: 12980km
        Cost: 28,750.00
Reach end of file
C0923   Mileage: 23000km
        Cost: 10,000.00
C5562   Mileage: 2500km
        Cost: 21,500.00
T1123   Mileage: 5000km
        Cost: 45,000.00
V9981   Mileage: 1500km
        Cost: 25,000.00
V1922   Mileage: 0km
        Cost: 28,750.00
V1982   Mileage: 15500km
        Cost: 28,750.00
Reach end of file
```