INFORMATICS INSTITUTE OF TECHNOLOGY

In collaboration with

UNIVERSITY OF WESTMINSTER

Object Oriented Principles

5COSC007C

Coursework – Phase 2

Vehicle Rental System

Module Leader’s Name – Mr. Guhanathan Poravi

Dinuka Piyadigama
UoW ID – 17421047
IIT ID – 2018373

Contents

ConsoleApp

```
package lk.dinuka.VehicleRentalSystem;

import lk.dinuka.VehicleRentalSystem.Controller.DatabaseController;
import lk.dinuka.VehicleRentalSystem.Controller.WestminsterRentalVehicleManager;

import java.util.Scanner;

public class ConApp {

    public static void main(String[] args) {
        int chooseOption;

        do {
            System.out.println("\n\t\\~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~/");
            System.out.println("\t~~\tVehicle Rental System\t~~");
            System.out.println("\t/~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\\");

            //display main menu
            System.out.println("\n1)Add item");
            System.out.println("2)Delete item");
            System.out.println("3)Print list of items");
            System.out.println("4)Open GUI");
            System.out.println("5)Exit program");
            Scanner sc = new Scanner(System.in);
            System.out.print("\nEnter Option:\n>>");

            while (!sc.hasNextInt()) {           //validation for integer input
                System.out.println("Only integer numbers are allowed! Please provide a valid input");
//error handling message for characters other than integers
                sc.next();                                        //removing incorrect input entered
            }

            chooseOption = sc.nextInt();

            WestminsterRentalVehicleManager managementAction = new
WestminsterRentalVehicleManager();       //new object


            switch (chooseOption) {
                case 1:        //add vehicle
                    managementAction.addVehicle();
                    break;

                case 2:        //delete vehicle
                    managementAction.deleteVehicle();
                    break;

                case 3:        //print list of vehicles
```

```
                managementAction.printList();
                break;

            case 4:     //open GUI
                managementAction.viewGUI();
                break;

            case 5:     //display exit message
                System.out.println("\nThank you for using the Vehicle Management System");
                System.out.println("\tLooking forward to assist you in the future.");
                System.out.println("\tExiting Program...");
                System.exit(0);

            default:
                System.out.println("Invalid input. Please try again");
            }
        } while (chooseOption != 5);
    }
}
```

WestminsterRentalVehicleManager

```
package lk.dinuka.VehicleRentalSystem.Controller;

import lk.dinuka.VehicleRentalSystem.Model.*;
import lk.dinuka.VehicleRentalSystem.View.GUI;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Scanner;

public class WestminsterRentalVehicleManager implements RentalVehicleManager {

    private static Scanner scanInput = new Scanner(System.in);

    protected static HashMap<String, Vehicle> allVehicles = new HashMap<>();        //used to
check whether the plate No already exists in the system
    protected static ArrayList<Vehicle> vehiclesInSystem = new ArrayList<>();      //used for
sorting and printing.    protected: making sure that customers can't modify the vehicles in the
system
```

```
    public static HashMap<String, Schedule> bookedVehicles = new HashMap<>();       //used to
record pick up & drop off dates of vehicles   (plateNo, Schedule)

    public static ArrayList<Vehicle> getVehiclesInSystem() {        //accessed in GUI
        return vehiclesInSystem;
    }



    private static String plateNo;
    private static String make;
    private static String model;
    private static boolean availability;
    private static Schedule schedule;          //used in GUI controller, when booking is made???
(Java/ Angular??)
    private static String engineCapacity;
    private static double dailyCostD;
    private static BigDecimal dailyCostBigD;
    private static String startType;
    private static double wheelSize;
    private static String transmission;
    private static boolean hasAirCon;
    private static String type;

    private static boolean replaceVeh;         //used to check whether vehicle data is being added
or edited


    @Override
    public void addVehicle() {

        if (Vehicle.getCount() <= MAX_VEHICLES) {      //checking whether the vehicles existing in
the system has occupied all the available parking lots

            System.out.println("\nChoose the type of Vehicle to be added:");
            System.out.println("1)Car\n2)Motorbike");
            System.out.print(">");
            intInputValidation();
            int typeSelection = scanInput.nextInt();
            scanInput.nextLine();            //to consume the rest of the line


            System.out.println("\nEnter Plate No:");
            System.out.print(">");
            plateNo = scanInput.nextLine();

            if (allVehicles.containsKey(plateNo)) {
                System.out.println("This Plate No exists in the system.");
                System.out.println();         //to keep space for clarity
                replaceVeh = false;

                //print information of vehicle
                System.out.println("Make: " + allVehicles.get(plateNo).getMake());
                System.out.println("Model: " + allVehicles.get(plateNo).getModel());
                System.out.println("Availability: " + allVehicles.get(plateNo).isAvailability());
                System.out.println("Engine Capacity: " + allVehicles.get(plateNo).getEngineCapacity());
                System.out.println("Daily Cost: " + allVehicles.get(plateNo).getDailyCost());
                System.out.println("Type: " + allVehicles.get(plateNo).getType());

                if (allVehicles.get(plateNo) instanceof Car) {
                   System.out.println("Transmission: " + ((Car)
allVehicles.get(plateNo)).getTransmission());
                   System.out.println("Has Air Conditioning: " + ((Car)
allVehicles.get(plateNo)).isHasAirCon());
                } else {
                   System.out.println("Start Type: " + ((Motorbike)
allVehicles.get(plateNo)).getStartType());
                   System.out.println("Wheel Size: " + ((Motorbike)
allVehicles.get(plateNo)).getWheelSize());
                }


                System.out.println();         //to keep space for clarity
                System.out.println("Do u want to edit information related to this vehicle?");
                System.out.print(">");

                boolean edit = yesOrNo();

                if (edit) {

                   replaceVeh = true;

                   addInfo(typeSelection);           //add information related to a Vehicle of identified
plateNo.

                } else {
                   System.out.println();      //keeps space and goes back to main menu
                }
            } else {

                addInfo(typeSelection);           //add information related to a Vehicle of identified
plateNo.
                save();
            }


        } else {
            System.out.println("There are no available spaces. 50 vehicles have been added!");
```

```
        }
    }

    @Override
    public void deleteVehicle() {                //delete item by entering plate no. of vehicle
        System.out.println("Enter the plate number of the vehicle that u desire to delete:");
        System.out.print(">");             //get plateNo from user to choose vehicle to be deleted
        String searchNo = scanInput.nextLine();

        if (allVehicles.containsKey(searchNo)) {
            Vehicle vehicleToBeDeleted = findVehicle(searchNo);

            type = vehicleToBeDeleted.getType();

            System.out.println("\nA " + type + " has been deleted from the system.");
            System.out.println("The details of the vehicle that was deleted:" +
vehicleToBeDeleted.toString());      //displaying information of deleted vehicle

            vehiclesInSystem.remove(vehicleToBeDeleted);
            allVehicles.remove(searchNo);
            Vehicle.count -= 1;         //decreasing the number of vehicles from the system by one

            System.out.println("There are " + (MAX_VEHICLES - Vehicle.getCount()) + " parking lots
left in the garage.");

//          save();     //save changes to file??

        } else {
            System.out.println("There's no item related to the item ID: " + searchNo);
        }
    }


    @Override
    public void printList() {       //prints list of vehicles in the system

        Collections.sort(vehiclesInSystem);     //sort vehicles alphabetically, according to make


        // print the plate number, the type of vehicle (car/ van/ motorbike).

        String leftAlignFormat = "| %-14s | %-12s |%n";

        System.out.format("+-----------------+--------------+%n");
        System.out.format("|   Plate ID      |   Type       |%n");
        System.out.format("+-----------------+--------------+%n");

        for (Vehicle item : vehiclesInSystem) {
            if (item instanceof Car) {
```

```
            System.out.format(leftAlignFormat, item.getPlateNo(), "Car");
        } else if (item instanceof Motorbike) {
            System.out.format(leftAlignFormat, item.getPlateNo(), "Motorbike");
        }
    }
    System.out.println("+--------------------------------+");
}

@Override
public void save() {        //saves the information of vehicles entered into the system

    //Rewrite the file every time a change is made.
    try {       //creating the file
        File myFile = new File("allVehicles.txt");
        myFile.createNewFile();


        for (Vehicle vehicle1 : vehiclesInSystem) {
            soldFile.write(vehicle1.toString());
            soldFile.write(System.getProperty("line.separator"));       //line break
        }
        soldFile.close();

    } catch (IOException e) {
        System.out.println("\nAn error occurred.");
        e.printStackTrace();
    }

}

@Override
public void viewGUI() {
}


//    ---- repeated methods ----

    private static void addInfo(int typeSelection) {         //method to add information related to a
Vehicle of identified plateNo.

        if (replaceVeh) {
            vehiclesInSystem.remove(allVehicles.get(plateNo));            //removing vehicle from
ArrayList, if editing it's information
        }

        if (typeSelection == 1) {       //new Car chosen
            addCommonInfo();

            type = "Car";
            System.out.println("\nEnter the type of transmission:");
            System.out.print(">");
            transmission = scanInput.nextLine();


            System.out.println("\nDoes this car have A/C?");
            System.out.print(">");

            hasAirCon = yesOrNo();


            Vehicle newCar = new Car(plateNo, make, model, availability, engineCapacity,
dailyCostBigD, type, transmission, hasAirCon);

            allVehicles.put(plateNo, newCar);          //adding a car into the allVehicles hashMap
            vehiclesInSystem.add(newCar);

            System.out.println(newCar);       //displaying added vehicle

        } else if (typeSelection == 2) {        //new Motorbike chosen
            addCommonInfo();

            type = "Motorbike";

            System.out.println("\nEnter start type:");
            System.out.print(">");
            startType = scanInput.nextLine();

            System.out.println("\nEnter wheel size:");
            System.out.print(">");
            wheelSize = scanInput.nextDouble();
            scanInput.nextLine();          //to consume the rest of the line


            Vehicle newBike = new Motorbike(plateNo, make, model, availability, engineCapacity,
dailyCostBigD, type, startType, wheelSize);

            allVehicles.put(plateNo, newBike);          //adding a motorbike into the allVehicles
hashMap
            vehiclesInSystem.add(newBike);

            System.out.println(newBike);       //displaying added vehicle
        }

        System.out.println("\nThere are " + (MAX_VEHICLES - Vehicle.getCount()) + " parking lots
left, to park vehicles.");

            save();    //save changes to file??
```

```
    }


    private static void addCommonInfo() {       //common information related to Car & Motorbike
in addVehicle


        System.out.println("\nEnter Make:");
        System.out.print(">");
        make = scanInput.nextLine();

        System.out.println("\nEnter Model:");
        System.out.print(">");
        model = scanInput.nextLine();

        availability = true;        //availability is set to true when vehicle data is entered to the
system;

        System.out.println("\nEnter Engine Capacity:");
        System.out.print(">");
        engineCapacity = scanInput.nextLine();

        System.out.println("\nEnter Daily cost (in $):");
        System.out.print(">$");
        doubleInputValidation();
        dailyCostD = scanInput.nextDouble();

        dailyCostBigD = BigDecimal.valueOf(dailyCostD);     //converting double to BigDecimal, to
use for calculations


        scanInput.nextLine();            //to consume the rest of the line

    }


    private static boolean yesOrNo() {          //gets yes/ no input

        while (!scanInput.hasNextBoolean()) {                              //check whether this works
as expected!!!!!!!!!!!
            String inputYN = scanInput.nextLine().toLowerCase();
            if (inputYN.equals("y") || inputYN.equals("yes")) {
                return true;
            } else if (inputYN.equals("n") || inputYN.equals("no")) {
                return false;
            } else {
                System.out.println("Invalid input. Please try again.");
            System.out.print(">");
        }
      }
      return false;         //won't reach this point (added to get rid of the missing return
statement error)
  }


  private static void intInputValidation() {                  //validating integer input

      while (!scanInput.hasNextInt()) {
        System.out.println("Only integer numbers are allowed! Please provide a valid input");
//error handling message for characters other than integers
        scanInput.next();                                      //removing incorrect input entered
      }
  }

  private static void doubleInputValidation() {                  //validating double input

      while (!scanInput.hasNextDouble()) {
        System.out.println("Only numbers are allowed! Please provide a valid input");
//error handling message for characters other than integers
        scanInput.next();                                      //removing incorrect input entered
      }
  }



}
```