

INFORMATICS INSTITUTE OF TECHNOLOGY

In collaboration with

UNIVERSITY OF WESTMINSTER

Object Oriented Principles

5COSC007C

Coursework – Phase 1

Vehicle Rental System

Module Leader’s Name – Mr. Guhanathan Poravi

Dinuka Piyadigama

UoW ID – 17421047

IIT ID – 2018373

Contents

[Design 3](#_Toc23112539)

[1 & 2) Use Case Diagram 3](#_Toc23112540)

[3) Class Diagram 4](#_Toc23112541)

[Code 5](#_Toc23112542)

[4) Vehicle class 5](#_Toc23112543)

[5) Motorbike class 7](#_Toc23112544)

[6) Car 8](#_Toc23112545)

[7) RentalVehicleManager Interface 9](#_Toc23112546)

[8) WestminsterRentalVehicleManager class 10](#_Toc23112547)

[9) DateTime class 11](#_Toc23112548)

[10) Schedule class 13](#_Toc23112549)

[11) DatabaseController class 14](#_Toc23112550)

[12) ConApp class 15](#_Toc23112551)

[13) GUI class 15](#_Toc23112552)

# Design

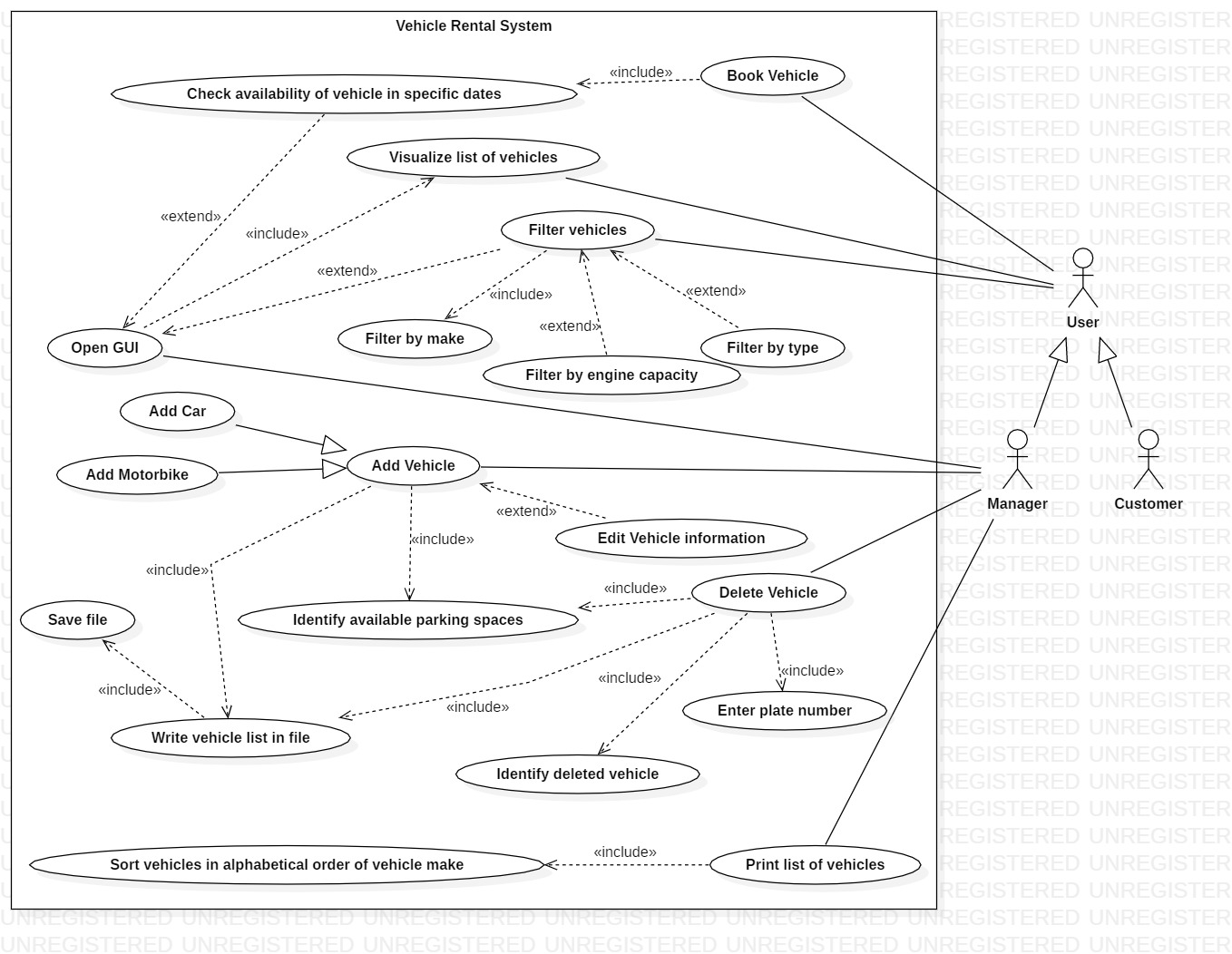
## 1 & 2) Use Case Diagram

*“You are* *required to develop a program that implements a basic vehicle rental system.”*

I have included the GUI section and the Console section in the same use case diagram as the assignments says that both of these are part of a single system.

***“Create******a******Graphical******User******Interface*** *(GUI) that can be opened selecting an option from the menu console”*

My use case diagram satisfies this condition as well.

But I have ensured that the customers can’t change the information in the system by using specialization, which clearly shows that the Manager is only allowed to perform managerial operations.

## 3) Class Diagram

# Code

## 4) Vehicle class

package lk.dinuka.VehicleRentalSystem.Model;

import java.math.BigDecimal;

import java.util.Objects;

public abstract class Vehicle implements Comparable<Vehicle>{

private String plateNo;

private String make;

private String model;

private boolean availability;

private Schedule schedule;

private String engineCapacity;

private BigDecimal dailyCost;

public static int count = 0;

public Vehicle(String plateNo, String make, String model, boolean availability, Schedule schedule, String engineCapacity, BigDecimal dailyCost) {

this.plateNo = plateNo;

this.make = make;

this.model = model;

this.availability = availability;

this.schedule = schedule;

this.engineCapacity = engineCapacity;

this.dailyCost = dailyCost;

count++;

}

@Override

public String toString() {

return "Vehicle{" +

"plateNo='" + plateNo + '\'' +

", make='" + make + '\'' +

", model='" + model + '\'' +

", availability=" + availability +

", schedule=" + schedule +

", engineCapacity='" + engineCapacity + '\'' +

", dailyCost=" + dailyCost +

'}';

}

public String getPlateNo() {

return plateNo;

}

public String getMake() {

return make;

}

public String getModel() {

return model;

}

public boolean isAvailability() {

return availability;

}

public Schedule getSchedule() {

return schedule;

}

public String getEngineCapacity() {

return engineCapacity;

}

public BigDecimal getDailyCost() {

return dailyCost;

}

public BigDecimal getCalculatedRent(){

return dailyCost; //calculate dailycost\*no of days and return!!!!!!!!!!!!!!!

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Vehicle vehicle = (Vehicle) o;

return availability == vehicle.availability &&

Objects.equals(plateNo, vehicle.plateNo) &&

Objects.equals(make, vehicle.make) &&

Objects.equals(model, vehicle.model) &&

Objects.equals(schedule, vehicle.schedule) &&

Objects.equals(engineCapacity, vehicle.engineCapacity) &&

Objects.equals(dailyCost, vehicle.dailyCost);

}

@Override

public int hashCode() {

return Objects.hash(plateNo, make, model, availability, schedule, engineCapacity, dailyCost);

}

@Override

public int compareTo(Vehicle obj) {

return this.make.compareTo(obj.getMake()); //used for sorting vehicle alphabetically according to make

}

}

## 5) Motorbike class

package lk.dinuka.VehicleRentalSystem.Model;

import java.math.BigDecimal;

import java.util.Objects;

public class Motorbike extends Vehicle {

private String startType;

private double wheelSize;

public Motorbike(String plateNo, String make, String model, boolean availability, Schedule schedule, String engineCapacity, BigDecimal dailyCost, String startType, double wheelSize) {

super(plateNo, make, model, availability, schedule, engineCapacity, dailyCost);

this.startType = startType; //making sure that this extra info is added when creating a new Motorbike object

this.wheelSize = wheelSize;

}

public String getStartType() {

return startType;

}

public void setStartType(String startType) {

this.startType = startType;

}

public double getWheelSize() {

return wheelSize;

}

public void setWheelSize(double wheelSize) {

this.wheelSize = wheelSize;

}

@Override

public String toString() {

return super.toString() + "Motorbike{" +

"startType='" + startType + '\'' +

", wheelSize=" + wheelSize +

'}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

if (!super.equals(o)) return false;

Motorbike motorbike = (Motorbike) o;

return Double.compare(motorbike.wheelSize, wheelSize) == 0 &&

Objects.equals(startType, motorbike.startType);

}

@Override

public int hashCode() {

return Objects.hash(super.hashCode(), startType, wheelSize);

}

}

## 6) Car

package lk.dinuka.VehicleRentalSystem.Model;

import java.math.BigDecimal;

import java.util.Objects;

public class Car extends Vehicle {

private String transmission;

private boolean hasAirCon;

public Car(String plateNo, String make, String model, boolean availability, Schedule schedule, String engineCapacity, BigDecimal dailyCost, String transmission, boolean hasAirCon) {

super(plateNo, make, model, availability, schedule, engineCapacity, dailyCost);

this.transmission = transmission; //making sure that this extra info is added when creating a new Car object

this.hasAirCon = hasAirCon;

}

public String getTransmission() {

return transmission;

}

public void setTransmission(String transmission) {

this.transmission = transmission;

}

public boolean isHasAirCon() {

return hasAirCon;

}

public void setHasAirCon(boolean hasAirCon) {

this.hasAirCon = hasAirCon;

}

@Override

public String toString() {

return super.toString() + "Car{" +

"transmission='" + transmission + '\'' +

", hasAirCon=" + hasAirCon +

'}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

if (!super.equals(o)) return false;

Car car = (Car) o;

return hasAirCon == car.hasAirCon &&

Objects.equals(transmission, car.transmission);

}

@Override

public int hashCode() {

return Objects.hash(super.hashCode(), transmission, hasAirCon);

}

}

## 7) RentalVehicleManager Interface

package lk.dinuka.VehicleRentalSystem.Model;

public interface RentalVehicleManager {

//constants

int MAX\_VEHICLES = 50;

//methods

void addVehicle(Vehicle newVehicle);

void deleteVehicle(Vehicle delVehicle);

void printList();

void save();

void viewGUI();

}

## 8) WestminsterRentalVehicleManager class

package lk.dinuka.VehicleRentalSystem.Controller;

import lk.dinuka.VehicleRentalSystem.Model.RentalVehicleManager;

import lk.dinuka.VehicleRentalSystem.Model.Vehicle;

import java.util.ArrayList;

import java.util.Scanner;

public class WestminsterRentalVehicleManager implements RentalVehicleManager {

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

protected static ArrayList<Vehicle> vehiclesInSystem = new ArrayList<>(); //making sure that customers can't modify the vehicles in the system

public static ArrayList<Vehicle> bookedVehicles = new ArrayList<>();

public static ArrayList<Vehicle> getVehiclesInSystem() { //accessed in GUI

return vehiclesInSystem;

}

@Override

public void addVehicle(Vehicle newVehicle) {

}

@Override

public void deleteVehicle(Vehicle delVehicle) {

}

@Override

public void printList() {

}

@Override

public void save() {

}

@Override

public void viewGUI() {

}

}

## 9) DateTime class

package lk.dinuka.VehicleRentalSystem.Model;

public class DateTime {

private int year;

private int month;

private int day;

private int hours;

private int mins;

private String ampm;

public DateTime(int year, int month, int day) { //this order of parameters needs to be maintained to properly validate day

this.year = year;

}

public DateTime(int year, int month, int day, int hours, int mins, String ampm) { //this order of parameters needs to be maintained to properly validate day

this.year = year;

setMonth(month); //validate month

setDay(day); //validate day

this.hours = hours; //have validations in setters!!!!!!!!!!!!!!

this.mins = mins; //have validations in setters!!!!!!!!!!!!!!

this.ampm = ampm; //have validations in setters!!!!!!!!!!!!!! and make sure that toString is given properly

System.out.printf("Date & Time entered is : %s\n", this); //checking input date & time

}

private void setMonth(int month) { //validate month

if (month > 0 && month <= 12) {

this.month = month;

} else {

System.out.printf("Invalid month (%d); set to 1\n", month);

this.month = 1; //inserted to maintain object in consistent state

}

}

private void setDay(int day) { //validate day

int[] daysPerMonth = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

if (month == 2 && day == 29 && (year % 400 == 0 || (year % 4 == 0 && year % 100 != 0))) {

this.day = day;

} else if (day > 0 && day <= daysPerMonth[month]) { //check if date is within range of month

this.day = day;

} else {

System.out.printf("Invalid day (%d); set to 1\n", day);

this.day = 1; //inserted to maintain object in consistent state

}

}

public int getYear() {

return year;

}

public int getMonth() {

return month;

}

public int getDay() {

return day;

}

public int getHours() {

return hours;

}

private void setHours(int hours) {

this.hours = hours;

}

public int getMins() {

return mins;

}

private void setMins(int mins) {

this.mins = mins;

}

public String getAmpm() {

return ampm;

}

private void setAmpm(String ampm) {

this.ampm = ampm;

}

@Override

public String toString() {

return "" + day +

"/" + month +

"/" + year +

" " + hours +

":" + mins +

"" + ampm +

"";

}

}

## 10) Schedule class

package lk.dinuka.VehicleRentalSystem.Model;

import java.util.Objects;

public class Schedule {

private DateTime pickUp;

private DateTime dropOff;

public Schedule(DateTime pickUp, DateTime dropOff) {

this.pickUp = pickUp;

this.dropOff = dropOff;

}

public DateTime getPickUp() {

return pickUp;

}

public void setPickUp(DateTime pickUp) { //Is there a point in having setters here!!!!!!!?????

this.pickUp = pickUp;

}

public DateTime getDropOff() {

return dropOff;

}

public void setDropOff(DateTime dropOff) {

this.dropOff = dropOff;

}

@Override

public String toString() {

return "Schedule{" +

"pick up=" + pickUp +

", drop off=" + dropOff +

'}';

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Schedule schedule = (Schedule) o;

return Objects.equals(pickUp, schedule.pickUp) &&

Objects.equals(dropOff, schedule.dropOff);

}

@Override

public int hashCode() {

return Objects.hash(pickUp, dropOff);

}

}

## 11) DatabaseController class

package lk.dinuka.VehicleRentalSystem.Controller;

public class DatabaseController {

// public static void addToDB() {

// //Adding a Motorbike to the Collection

// }

//

//

// public static void addToDB() {

// //Adding a car to the Collection

// }

public static void deleteFromDB(String plateNo) {

//Deleting an item from the Collection

}

public static void importDB() {

//Importing stored data in db to application

}

}

## 12) ConApp class

package lk.dinuka.VehicleRentalSystem;

public class ConApp {

public static void main(String[] args) {

}

}

## 13) GUI class

package lk.dinuka.VehicleRentalSystem.View;

public class GUI {

}