

**Name:** Charana Randunu Mayakaduwa

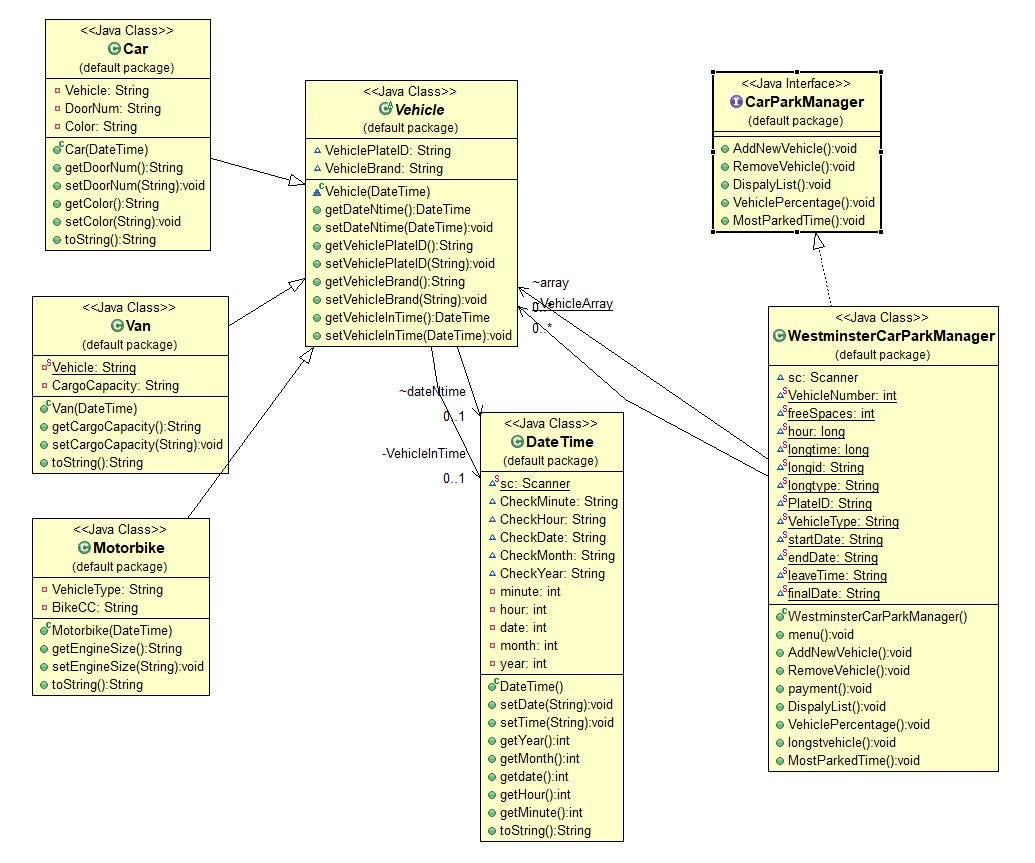
**UOW ID:** 16266632

**IIT ID:** 2016139

**Module:** [5COSC007C.1](https://learning.westminster.ac.uk/webapps/blackboard/execute/launcher?type=Course&id=_64090_1&url=) Object Oriented Programming

**Module Leader:** Mr. Guhanathan Poravi

**UML Diagram**



**Use case diagrams**

Car Park

User

<<include>>

Car Park

<<include>>

System Administrator

**Code**

Available Classes :

1. DateTime.java
2. Vehicle.java
3. Car.java
4. Van.java
5. Motorbike.java
6. CarParkManger.java (System interface)
7. CarParkSystem.java
8. WestminsterCarParkManager.java

1 - DateTime.java

**import** java.util.Scanner;

**public** **class** DateTime {

**static** Scanner *sc* = **new** Scanner(System.***in***);

//creating variables to check date and time

String CheckMinute ="([0-5]\\d|60)";

String CheckHour = "([0|1]\\d|2[0-4])";

String CheckDate = "((1|2)\\d|3(0|1)|0[1-9]|[1-9])";

String CheckMonth = "(0[1-9]|1[0-2])";

String CheckYear = "[2-9]\\d{3}";

//variables for DateTime attributes

**private** **int** minute;

**private** **int** hour;

**private** **int** date;

**private** **int** month;

**private** **int** year;

**public** **void** setDate(String inputDate){

//checking if the input date is correct

**if**(inputDate.matches(CheckDate + "\\/"+CheckMonth +"\\/" + CheckYear)){

String[] DDMMYY = (inputDate.split("\\/"));

year = Integer.*parseInt*(DDMMYY[0]);

month = Integer.*parseInt*(DDMMYY[1]);

date = Integer.*parseInt*(DDMMYY[2]);

}**else**{

//if the date is incorrect prompt the user

System.***out***.println("Enter a valid date on the format of \"DD/MM/YY\"");

setDate(*sc*.next());

}

}

**public** **void** setTime(String inputTime){

//checking if the input time is correct

**if**(inputTime.matches(CheckHour + "\\:"+CheckMinute )){

String[] HHMM = (inputTime.split("\\:"));

hour = Integer.*parseInt*(HHMM[0]);

minute = Integer.*parseInt*(HHMM[1]);

}**else**{

//if the time is incorrect prompt the user

System.***out***.println("Please enter a valid time on \"HH:mm\" format: ");

setTime(*sc*.next());

}

}

//getters and setters for DateTime attributes

**public** **int** getYear(){

**return** year ;

}

**public** **int** getMonth(){

**return** month ;

}

**public** **int** getdate(){

**return** date ;

}

**public** **int** getHour(){

**return** hour ;

}

**public** **int** getMinute(){

**return** minute ;

}

//generating constructor

**public** String toString(){

**return** String.*format*("%d/%d/%d %d:%d", year,month,date,hour,minute);

}

}

2 - Vehicle.java

**public** **abstract** **class** Vehicle {

//setting attributes

DateTime dateNtime;

String VehiclePlateID;

String VehicleBrand;

**private** DateTime VehicleInTime;

//vehicle parking-in time

Vehicle (DateTime InTime){

**this**.VehicleInTime = InTime;

}

//getter for date and time

**public** DateTime getDateNtime() {

**return** dateNtime;

}

//setter for date and time

**public** **void** setDateNtime(DateTime dateNtime) {

**this**.dateNtime = dateNtime;

}

//getter for plate ID

**public** String getVehiclePlateID() {

**return** VehiclePlateID;

}

//setter for plate ID

**public** **void** setVehiclePlateID(String vehiclePlateID) {

VehiclePlateID = vehiclePlateID;

}

//getter for vehicle brand

**public** String getVehicleBrand() {

**return** VehicleBrand;

}

//setter for vehicle brand

**public** **void** setVehicleBrand(String vehicleBrand) {

VehicleBrand = vehicleBrand;

}

//getter for vehicle in time

**public** DateTime getVehicleInTime() {

**return** VehicleInTime;

}

//setter for vehicle in time

**public** **void** setVehicleInTime(DateTime vehicleInTime) {

VehicleInTime = vehicleInTime;

}

}

3 - Car.java

**public** **class** Car **extends** Vehicle {

//setting car attributes

**private** String Vehicle = "Car";

**private** String DoorNum;

**private** String Color;

//car parking-in time

**public** Car(DateTime InTime) {

**super**(InTime);

}

//getter for number of doors

**public** String getDoorNum() {

**return** DoorNum;

}

//setter for number of doors

**public** **void** setDoorNum(String doorNum) {

DoorNum = doorNum;

}

//getter for car color

**public** String getColor() {

**return** Color;

}

//setter for car color

**public** **void** setColor(String color) {

Color = color;

}

//generating constructor

**public** String toString() {

**return** "ID : " + getVehiclePlateID() + "\n" + " Brand : " + getVehicleBrand() + "\n" + " Number Of Doors : " + getDoorNum();

}

}

4 - Van.java

**public** **class** Van **extends** Vehicle {

//setting van attributes

**private** **static** String *Vehicle* = "Van";

**private** String CargoCapacity;

//van parking-in time

**public** Van(DateTime InTime) {

**super**(InTime);

}

//getter for cargo capacity

**public** String getCargoCapacity() {

**return** CargoCapacity;

}

//setter for cargo capacity

**public** **void** setCargoCapacity(String cargoCapacity) {

CargoCapacity = cargoCapacity;

}

//generating constructor

**public** String toString() {

**return** "ID : " + getVehiclePlateID() + "\n" + " Brand : " + getVehicleBrand() + "\n" + " Cargo Volume : " + getCargoCapacity();

}

}

5 – Motorbike.java

**public** **class** Motorbike **extends** Vehicle {

//setting motorbike attributes

**private** String VehicleType = "Motorbike";

**private** String BikeCC;

//motorbike parking-in time

**public** Motorbike(DateTime VehicleInTime) {

**super**(VehicleInTime);

}

//getter for engine size

**public** String getEngineSize() {

**return** BikeCC;

}

//setter for engine size

**public** **void** setEngineSize(String EngineSize) {

**this**.BikeCC = EngineSize;

}

//generating constructor

**public** String toString() {

**return** "ID : " + getVehiclePlateID() + "\n" + " Brand : " + getVehicleBrand() + "\n" + " Engine Capacity : " + getEngineSize();

}

}

6 – CarParkManager.java

**interface** CarParkManager {

//system interface

//required main functionalities for the car park system

**public** **void** AddNewVehicle();

**public** **void** RemoveVehicle();

**public** **void** DispalyList();

**public** **void** VehiclePercentage();

**public** **void** MostParkedTime();

}

7 – CarParkSystem.java

**public** **class** CarParkSystem {

//main menu

**public** **static** **void** main(String[] args) {

System.***out***.println("===========================================");

System.***out***.println(" Westminster Car Park Manager ");

System.***out***.println("===========================================");

System.***out***.println();

//creating new object of WestminsterCarParkManager

WestminsterCarParkManager CPM = **new** WestminsterCarParkManager();

//invoking main menu

CPM.menu();

}

}

8 – WestminsterCarParkManager.java

import static com.sun.org.apache.xml.internal.security.keys.keyresolver.KeyResolver.length;

import java.io.File;

import java.io.FileOutputStream;

import java.io.ObjectOutputStream;

import static java.lang.String.format;

import static java.lang.String.format;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.Scanner;

public class WestminsterCarParkManager implements CarParkManager {

Scanner sc = new Scanner(System.in);

//creating array list

static ArrayList<Vehicle> VehicleArray = new ArrayList<Vehicle>(20);

Vehicle[] array;

static int VehicleNumber = 0;

static int freeSpaces;

static long hour;

static long longtime=0;

static String longid;

static String longtype;

static String PlateID;

static String VehicleType;

static String startDate;

static String endDate;

static String leaveTime;

static String finalDate;

//main menu

public void menu(){

System.out.println("Enter your letter of choice");

System.out.println("N - Adding a new vehicle");

System.out.println("R - Remove a vehicle");

System.out.println("L - List of parked vehicles");

System.out.println("P - Percentage of parked vehicle types");

System.out.println("T - Vehicle whick has the longest park time");

//getting the user input

String se = sc.next();

switch(se.toUpperCase())

//filtering the input

{

case "N" : AddNewVehicle();

break;

case "R" : RemoveVehicle();

break;

case "L" : DispalyList();

break;

case "P" : VehiclePercentage();

break;

case "T" : MostParkedTime();

break;

default :

System.out.println("Incorrect Input");

menu();

}

}

//method for adding a new vehicle

@Override

public void AddNewVehicle() {

//calculating the slots available

freeSpaces = 19-VehicleNumber;

//prompting the user for inputs

System.out.println("");

System.out.println("Enter a valid vehicle plate ID:");

String vehicleId = sc.next();

System.out.println("Enter a valid vehicle brand:");

String vehicleBrand = sc.next();

System.out.println("Enter a valid entry date: [DD/MM/YYYY]");

String vehicleDate = sc.next();

System.out.println("Enter a valid entry time: [HH/MM]");

String vehicleTime = sc.next();

//selecting the type of vehicle you are adding

System.out.println("----------------------------------------------");

System.out.println(" ADD VEHICLE TO CAR PARK ");

System.out.println("----------------------------------------------");

System.out.println("1 - To add a Car");

System.out.println("2 - To add a Van");

System.out.println("3 - To add a Motorbike");

int choise1 = sc.nextInt();

switch(choise1){

case 1:

//check whether car park has empty slots

if(VehicleNumber<19){

//displaying the slots available for cars

System.out.println("Available Free Slots: "+freeSpaces);

//prompting the user for car specific details

System.out.println("--- Add A Car ---");

DateTime dateTime = new DateTime();

Car car = new Car(dateTime);

System.out.println("");

System.out.println("Enter the car color:");

String carColor = sc.next();

System.out.println("Enter the number Of doors in the car:");

String carNoOfDoors = sc.next();

//set the values for the car

car.setVehiclePlateID(vehicleId);

car.setVehicleBrand(vehicleBrand);

car.setDoorNum(carNoOfDoors);

car.setColor(carColor);

dateTime.setDate(vehicleDate);

dateTime.setTime(vehicleTime);

//add the car to the array

VehicleArray.add(car);

//increase the vehicle count

VehicleNumber++;

car.getDateNtime();

//invoke the main menu

menu();

//if no empty slots are available

}else{

System.out.println("\*\*\* Car Park Full \*\*\*");

//invoke the main menu

menu();

}

break;

case 2:

//check whether car park has empty slots

if(VehicleNumber<19){

//displaying the slots available for vans

System.out.println("Free Slots: "+freeSpaces);

//prompting the user for van specific details

System.out.println("--- Add a Van ---");

DateTime dateTime2 = new DateTime();

Van van = new Van(dateTime2);

System.out.println("Enter the cargo volume of the Van: ");

String vanCargoVol = sc.next();

//set the values for the van

van.setVehiclePlateID(vehicleId);

van.setVehicleBrand(vehicleBrand);

van.setCargoCapacity(vanCargoVol);

dateTime2.setDate(vehicleDate);

dateTime2.setTime(vehicleTime);

//add the van to the array

VehicleArray.add(van);

//increase the vehicle count

VehicleNumber++;

//invoke the main menu

menu();

//if no empty slots are available

}else{

System.out.println("\*\*\* Car Park Full \*\*\*");

//invoke the main menu

menu();

}

break;

case 3:

//check whether car park has empty slots

if(VehicleNumber<19){

//displaying the slots available for motorbike

System.out.println("Free Slots: "+freeSpaces);

System.out.println("");

//prompting the user for motorbike specific details

System.out.println("--- Add A Motorbike ---");

DateTime dateTime3 = new DateTime();

Motorbike motorbike = new Motorbike(dateTime3);

System.out.println("Enter the engine capacitys: ");

String SizeOfEngine = sc.next();

//set the values for the motorbike

motorbike.setVehiclePlateID(vehicleId);

motorbike.setVehicleBrand(vehicleBrand);

motorbike.setEngineSize(SizeOfEngine);

dateTime3.setDate(vehicleDate);

dateTime3.setTime(vehicleTime);

//add the motorbike to the array

VehicleArray.add(motorbike);

//increase the vehicle count

VehicleNumber++;

//invoke the main menu

menu();

//if no empty slots are available

}else{

System.out.println("\*\*\* Car Park Full \*\*\*");

//invoke the main menu

menu();

}

break;

}

}

//method for removing a vehicle

@Override

public void RemoveVehicle() {

//get the vehicle id from the user

System.out.println("Enter the vehicle ID of the removing vehicle ");

String choice = sc.next();

//set the vehicles array index null and id plate availability to false

int arrayIndex = 0;

boolean found = false;

for (Vehicle h : VehicleArray) {

//if the vehicle id is correct, prompt for leaving time and date

if (h.getVehiclePlateID().equalsIgnoreCase(choice)) {

startDate =(h.getDateNtime()).toString();

System.out.println("Leave Date [YYYY/MM/DD]");

endDate = sc.next();

System.out.println("Leave Time [HH:MM]");

leaveTime =sc.next();

finalDate=(endDate+" "+leaveTime);

PlateID = h.getVehiclePlateID();

//set the id plate availability to true

found = true;

//prompt whick type of vehicle left

if (h instanceof Van) {

System.out.println("A van left");

VehicleType="Van";

} else if (h instanceof Car) {

System.out.println("A car left");

VehicleType = "Car";

} else if (h instanceof Motorbike) {

System.out.println("A bike left");

VehicleType = "Motorbike";

}

break;

}

//increase the array array index of removed vehicles

arrayIndex++;

}

//clear the array index of the removed vehicle

if (found) {

VehicleArray.remove(arrayIndex);

//if the plate id is incorrect, prompt the user

}else{

System.err.println("Invalid vehicle ID");

}

//calculate the payment of the removing vehicle

payment();

}

//method for calculating the payment

public void payment(){

String dateStart = startDate;

String dateStop = finalDate;

//HH converts hour in 24 hours format (0-23), day calculation

SimpleDateFormat format = new SimpleDateFormat("yyyy/MM/dd HH:mm");

Date d1 = null;

Date d2 = null;

try {

d1 = format.parse(dateStart);

d2 = format.parse(dateStop);

//in milliseconds

long diff = d2.getTime() - d1.getTime();

long diffMinutes = diff / (60 \* 1000) % 60;

long diffHours = diff / (60 \* 60 \* 1000) % 24;

long diffDays = diff / (24 \* 60 \* 60 \* 1000);

long hour24 = diffDays\*24;

hour=hour24+diffHours;

if(hour>=0){

if(hour>3){

long nexthour=hour-3;

long price = nexthour+9;

System.out.println("$"+price+" parking charge");

}else{

long price2 = hour\*3;

System.out.println("$"+price2+" parking charge");

}

}else{

System.out.println("Invalid time enterd");

}

} catch (Exception e) {

e.printStackTrace();

}

menu();

}

//method for displaying the current list

@Override

public void DispalyList(){

array = new Vehicle[0];

array = new Vehicle[VehicleArray.size()];

int counter = 0;

for (Vehicle h : VehicleArray) {

array[counter] = h;

counter++;

}

//setting a swap counter to swap the index of the vehicle ids using their in time

boolean swapped = true;

int j = 0;

Vehicle tmp;

//swapping the indexes by comparing two nearby indexes starting from the array's 0th index

//swap if the recorded vehicles' year is different

while (swapped) {

swapped = false;

j++;

for (int i = 0; i < array.length - 1; i++) {

String l = (array[i].getDateNtime()).toString();

String[] dateSeperater = (l.split("\\ "));

String date = dateSeperater[0];

String time= dateSeperater[1];

DateTime dateTime = new DateTime();

dateTime.setDate(date);

dateTime.setTime(time);

int year1 = dateTime.getYear();

String f = (array[i+1].getDateNtime()).toString();

String[] dateSeperater1 = (f.split("\\ "));

String date1 = dateSeperater1[0];

String time1= dateSeperater1[1];

DateTime dateTime1 = new DateTime();

dateTime.setDate(date1);

dateTime.setTime(time1);

int year2 = dateTime.getYear();

if (year1 > year2) {

tmp = array[i];

array[i] = array[i + 1];

array[i + 1] = tmp;

swapped = true;

}

}

}

swapped = true;

//swap if the recorded vehicles' year is same, but the month is different

while (swapped) {

swapped = false;

j++;

for (int i = 0; i < array.length - 1; i++) {

String l = (array[i].getDateNtime()).toString();

String[] dateSeperater = (l.split("\\ "));

String date = dateSeperater[0];

String time= dateSeperater[1];

DateTime dateTime = new DateTime();

dateTime.setDate(date);

dateTime.setTime(time);

int year1 = dateTime.getYear();

int month1 = dateTime.getMonth();

String f = (array[i+1].getDateNtime()).toString();

String[] dateSeperater1 = (f.split("\\ "));

String date1 = dateSeperater1[0];

String time1= dateSeperater1[1];

DateTime dateTime1 = new DateTime();

dateTime.setDate(date1);

dateTime.setTime(time1);

int year2 = dateTime.getYear();

int month2 = dateTime.getMonth();

if (year1 == year2) {

if (month1 > month2) {

tmp = array[i];

array[i] = array[i + 1];

array[i + 1] = tmp;

swapped = true;

}

}

}

}

swapped = true;

//swap if the recorded vehicles' year and month is same, but the day is different

while (swapped) {

swapped = false;

j++;

for (int i = 0; i < array.length - 1; i++) {

String l = (array[i].getDateNtime()).toString();

String[] dateSeperater = (l.split("\\ "));

String date = dateSeperater[0];

String time= dateSeperater[1];

DateTime dateTime = new DateTime();

dateTime.setDate(date);

dateTime.setTime(time);

int year1 = dateTime.getYear();

int month1 = dateTime.getMonth();

int day1 = dateTime.getdate();

String f = (array[i+1].getDateNtime()).toString();

String[] dateSeperater1 = (f.split("\\ "));

String date1 = dateSeperater1[0];

String time1= dateSeperater1[1];

DateTime dateTime1 = new DateTime();

dateTime.setDate(date1);

dateTime.setTime(time1);

int year2 = dateTime.getYear();

int month2 = dateTime.getMonth();

int day2 = dateTime.getdate();

if (year1 == year2) {

if (month1 == month2) {

if (day1 > day2) {

tmp = array[i];

array[i] = array[i + 1];

array[i + 1] = tmp;

swapped = true;

}

}

}

}

}

swapped = true;

//swap if the recorded vehicles' year, month and day is the same, but hour is different

while (swapped) {

swapped = false;

j++;

for (int i = 0; i < array.length - 1; i++) {

String l = (array[i].getDateNtime()).toString();

String[] dateSeperater = (l.split("\\ "));

String date = dateSeperater[0];

String time= dateSeperater[1];

DateTime dateTime = new DateTime();

dateTime.setDate(date);

dateTime.setTime(time);

int year1 = dateTime.getYear();

int month1 = dateTime.getMonth();

int day1 = dateTime.getdate();

int hour1 = dateTime.getHour();

String f = (array[i+1].getDateNtime()).toString();

String[] dateSeperater1 = (f.split("\\ "));

String date1 = dateSeperater1[0];

String time1= dateSeperater1[1];

DateTime dateTime1 = new DateTime();

dateTime.setDate(date1);

dateTime.setTime(time1);

int year2 = dateTime.getYear();

int month2 = dateTime.getMonth();

int day2 = dateTime.getdate();

int hour2 = dateTime.getHour();

if (year1 == year2) {

if (month1 == month2) {

if (day1 == day2) {

if (hour1 > hour2) {

tmp = array[i];

array[i] = array[i + 1];

array[i + 1] = tmp;

swapped = true;

}

}

}

}

}

}

swapped = true;

//swap if the recorded vehicles' year, month, day and hour is the same, but minute is different

while (swapped) {

swapped = false;

j++;

for (int i = 0; i < array.length - 1; i++) {

String l = (array[i].getDateNtime()).toString();

String[] dateSeperater = (l.split("\\ "));

String date = dateSeperater[0];

String time= dateSeperater[1];

DateTime dateTime = new DateTime();

dateTime.setDate(date);

dateTime.setTime(time);

int year1 = dateTime.getYear();

int month1 = dateTime.getMonth();

int day1 = dateTime.getdate();

int hour1 = dateTime.getHour();

int minute1 = dateTime.getMinute();

String f = (array[i+1].getDateNtime()).toString();

String[] dateSeperater1 = (f.split("\\ "));

String date1 = dateSeperater1[0];

String time1= dateSeperater1[1];

DateTime dateTime1 = new DateTime();

dateTime.setDate(date1);

dateTime.setTime(time1);

int year2 = dateTime.getYear();

int month2 = dateTime.getMonth();

int day2 = dateTime.getdate();

int hour2 = dateTime.getHour();

int minute2 = dateTime.getMinute();

if (year1 == year2) {

if (month1 == month2) {

if (day1 == day2) {

if (hour1 > hour2) {

if (minute1 > minute2 ) {

tmp = array[i];

array[i] = array[i + 1];

array[i + 1] = tmp;

swapped = true;

}

}

}

}

}

}

}

for(int i = 0; i < VehicleArray.size(); i++){

System.out.println(array[i].getVehiclePlateID());

System.out.println(array[i].getDateNtime());

if (array[i] instanceof Van) {

System.out.println("A van is leaving.");

} else if (array[i] instanceof Car) {

System.out.println("A car is leaving.");

} else if (array[i] instanceof Motorbike) {

System.out.println("A bike is leaving.");

}

System.out.println(" ");

}

menu();

}

//method for displaying the vehicle percentage

@Override

public void VehiclePercentage() {

//set the counters 0 for all the vehicle types

int carCount=0;

int vanCount=0;

int motorbikeCount=0;

//increase the count, one by one, by checking the vehicle type

for (Vehicle h : VehicleArray) {

if( h !=null){

if (h instanceof Van) {

vanCount++;

} else if (h instanceof Car) {

carCount++;

} else if (h instanceof Motorbike) {

motorbikeCount++;

}

}

}

//get the total count of parked vehicles

int total = carCount+vanCount+motorbikeCount;

//calculate the percentages for each vehicle type

System.out.println(" ");

System.out.println("Cars Preantage"+carCount/total\*100+"%");

System.out.println("Vans Preantage"+vanCount/total\*100+"%");

System.out.println("Motorbikes Preantage"+motorbikeCount/total\*100+"%");

System.out.println(" ");

menu();

}

//method to check for the plate id which has the longest parking time

public void longstvehicle(){

if(longtime<hour){

longtime=hour;

longid=PlateID;

longtype=VehicleType;

}

}

//method to display the info of the longest parked vehicle

@Override

public void MostParkedTime() {

System.out.println(longid);

System.out.println(longtype);

System.out.println(longtime);

}

}