Documentation on

Car Park Manager

Table of Contents

LIST OF FIGURES

Introduction

This is a car park management system which was designed for University of Westminster. This system stores data about the vehicles in the car park and the system automatically calculates the charge for a particular vehicle depending on the entry time and the leaving time of the vehicle. The system holds a record for each and every vehicle that was parked in the system. As a result of that the user can search for vehicle by the date of entry date of the vehicle.

Functional Requirements

1) Adding a vehicle
   When the user needs to enter a vehicle to the car park first of all the system should check whether there are any free slots available. If the user is adding a van, then there must be two free parking slots together. The maximum number of the parking slots available is 20. When adding a vehicle, the system must get vehicle type, entry time and date, vehicle id plate number and brand. Other than that if a car is adding number of doors and the color of the car should be stored. For a van the cargo volume and for a motor bike the engine size also should be recorded. After adding the vehicle, the number of the number of available free parking slots should be printed.

2) Removing a vehicle
   To remove a vehicle from a parking slot the user should enter the vehicle id plate number. Then the corresponding vehicle instance should be removed from the slot. But the system should maintain a record of that vehicle for future requirements (ex: Search by date). After deleting the removed vehicle type should be displayed as well.

3) Print a list of vehicles currently parked
   The system should give a list of vehicles which are currently parked at the car park. Vehicle id plate number. Vehicle entry date and time, vehicle type should be displayed. The vehicles list must be ordered chronologically from last entered vehicle to earliest.

4) Print a list of vehicles parked in a specific day
   When user enters a particular date then the system should list down the parked vehicles on that day. If no vehicles were parked them a relevant message should be printed.

5) Calculate parking charge for each customer
   3£ per hour is charged for the first three hours and the car park charges additional 1£ per hour after first 3 three hours. But the maximum charge for a day is 30£. Parking charge for each customer should be displayed with vehicle plate id number and the price.

6) Print statistical data
   Percentages of cars, vans, motor bikes currently parked should be displayed and the vehicle which was parked for the longest time period should be printed with relevant details (vehicle plate id number, type, entry time). Also the last parked vehicle should be printed with relevant details.

Non-Functional Requirements

1) Usability
2) Quality
3) Reliability
4) Performance

Domain Model

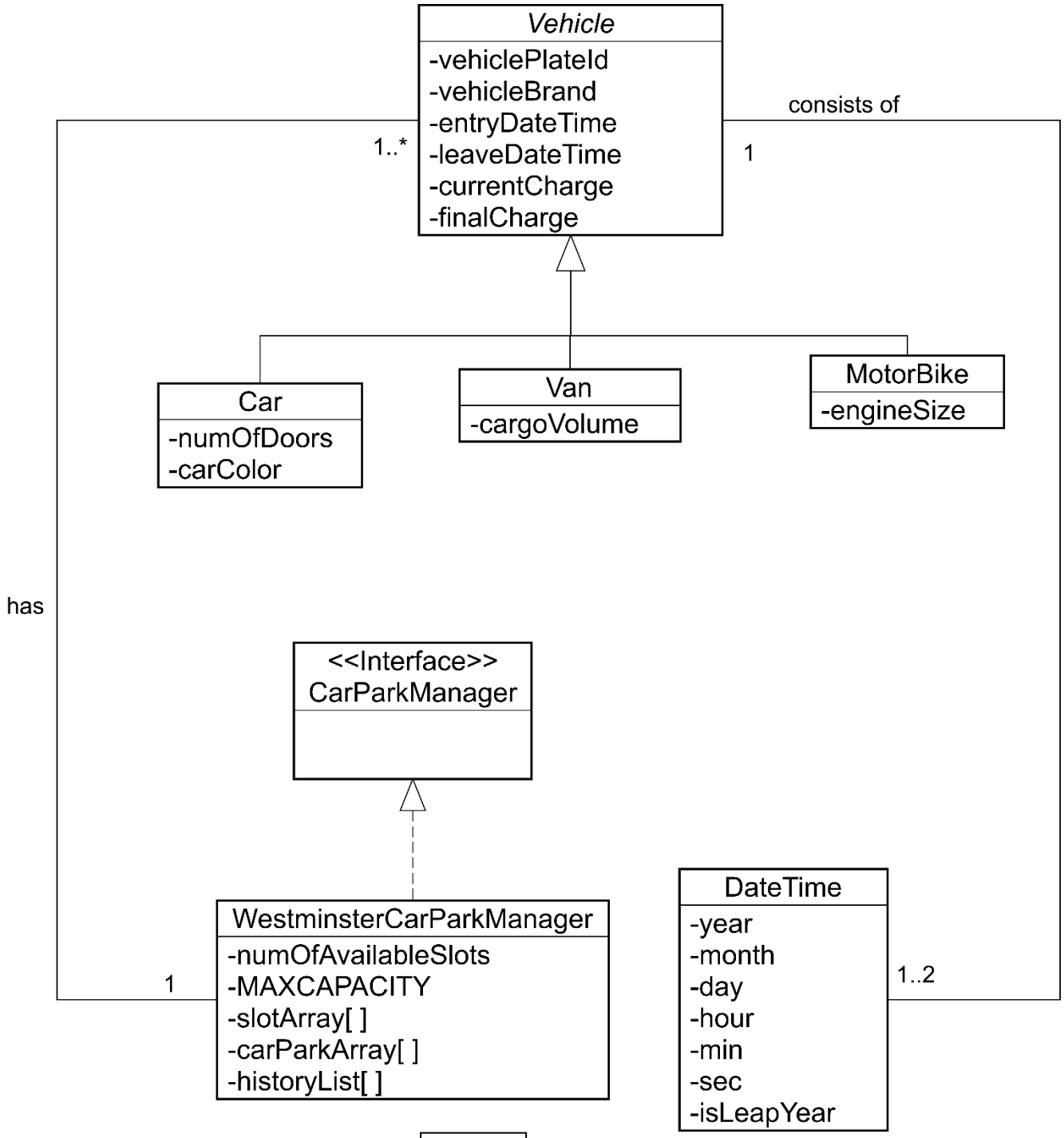

Figure 1

Class Diagram

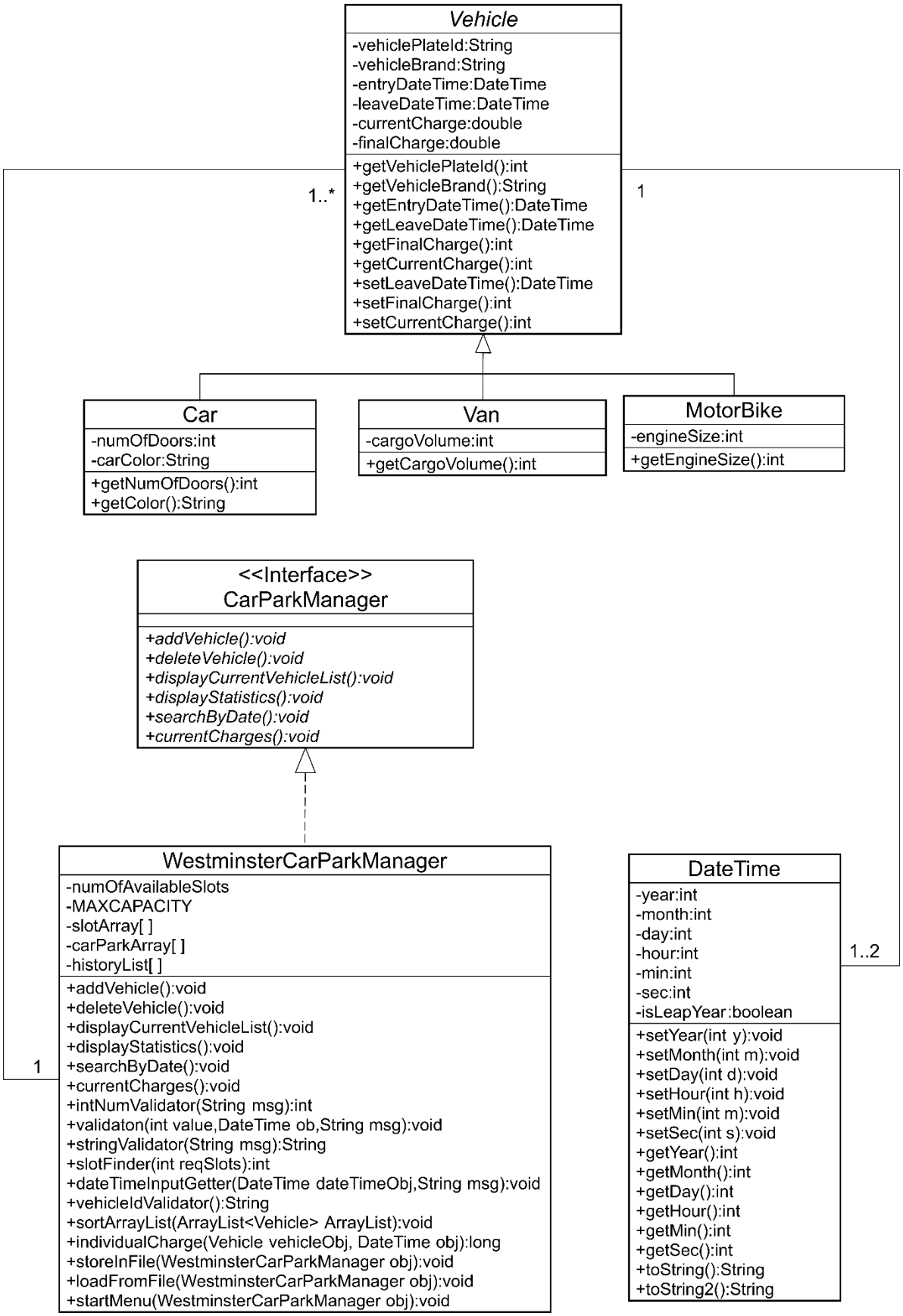

Figure 2

Use Case Diagrams

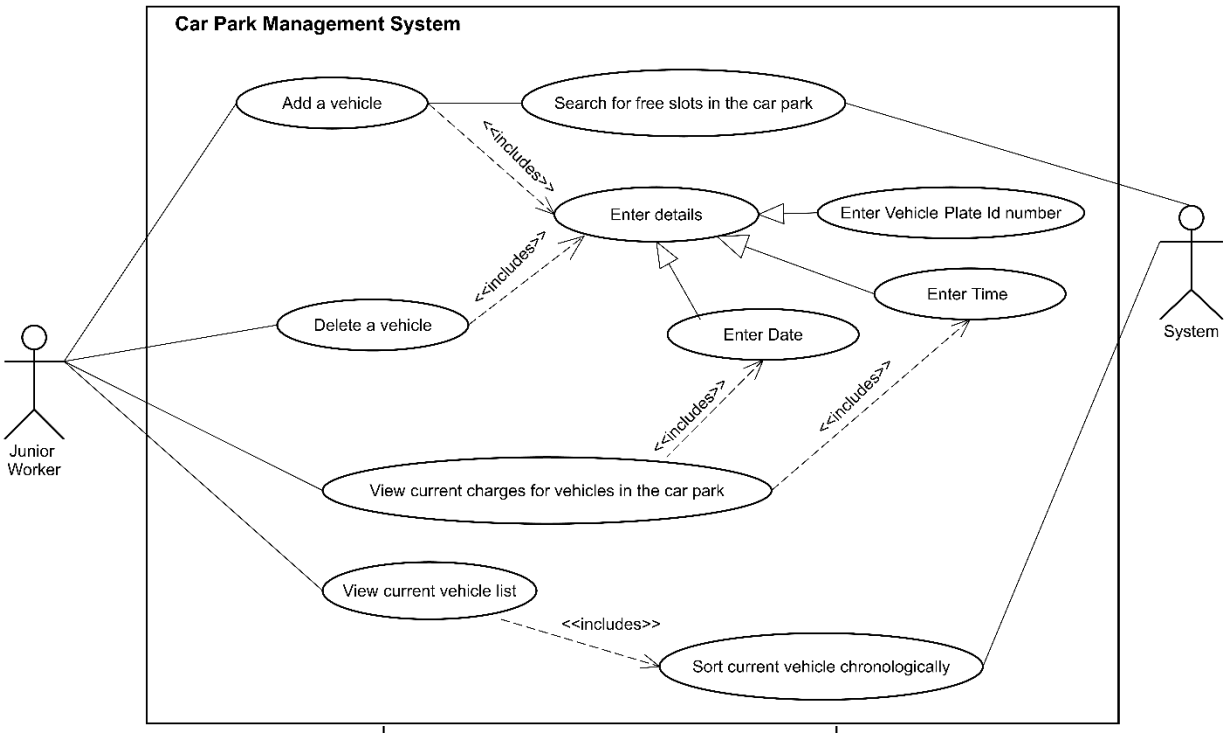

Figure 3: Junior Worker point of view

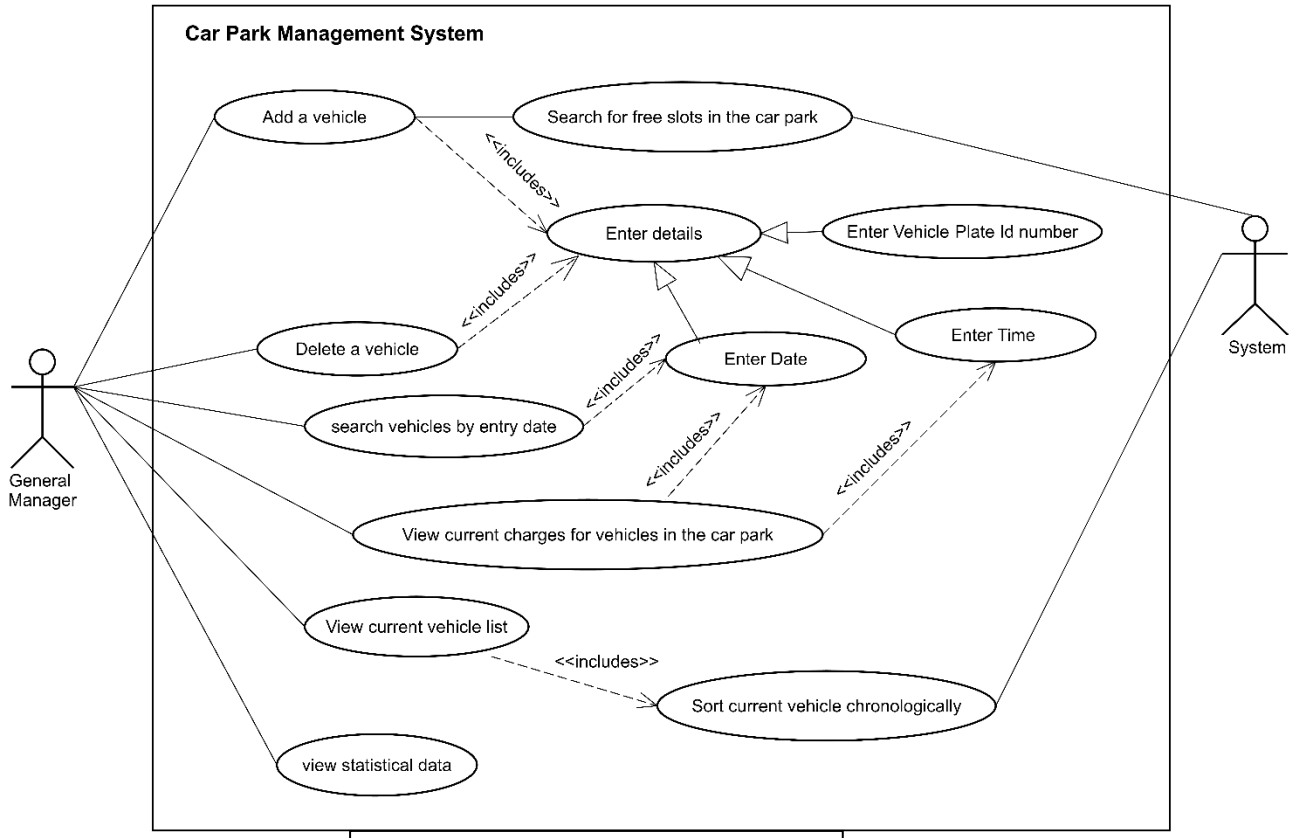

Figure 4: General Manager point of view

Use Case Descriptions

Use Case Name:	Add a vehicle	
Use Case No:	01	
Priority:	High	
Participating Actors:	User (General Manager/Junior Worker), System	
Pre Conditions:	numOfAvailableSlots>0	
Trigger Event:	Selecting “Add Vehicle” from the main menu	
	User	System
Main flow of events:	2)Select vehicle type 4) <<include>> Enter details	1) Prompt for vehicle type 3) <<include>> Search free slots in the car park 5) Allocate a parking slot 6) Store the vehicle details in the system 7)Display the remaining parking slots 8) Go back to main menu
Alternate flow 1:		3) Display “Invalid input” 4) Go to 1)
Exceptional Flow 1:		1) numOfAvailableSlots<0 2) Display “Sorry. No available free slots in the car park” 3) Go to 8)
Exceptional Flow 2:		4) Display” Sorry. No enough parking slots for a van” 3)Go to 8)
Post-Conditions:	-	
Inclusions	Enter Details, Search free slots in the car park	
Extensions	-	

Use Case Name:	Enter Details	
Use Case No:	02	
Priority:	High	
Participating Actors:	User (General Manager/Junior Worker), System	
Pre Conditions:	-	
Trigger Event:	Enter details option requested	
	User	System
Main flow of events:	2) Enter entry date 4) Enter entry time 6) Enter vehicle Id plate number 8)Enter brand	1) Prompt for vehicle entry date 3) Prompt for vehicle entry time 5) Prompt for vehicle Id plate number 7) Prompt for vehicle brand 9) Go back to main menu
Alternate flow 1:		3) Display “Invalid date” 4) Go to 1)
Alternate flow 2:		5) Display “Invalid time” 6) Go to 3)
Exceptional Flow 1:	-	
Post-Conditions:	-	
Inclusions		
Extensions	-	

Use Case Name:	Delete a vehicle	
Use Case No:	03	
Priority:	High	
Participating Actors:	User (General Manager/Junior Worker), System	
Pre Conditions:	-	
Trigger Event:	Selecting “Delete a Vehicle” from the main menu	
	User	System
Main flow of events:	1) <<include>> Enter details	2) Remove the vehicle details from current parking information 3) Print the type of removed vehicle 4) Go back to main menu
Alternate Flow:	-	
Exceptional Flow:	-	
Post-Conditions:	-	
Inclusions	Enter Details	
Extensions	-	

Use Case Name:	View current charge for the vehicles in the car park	
Use Case No:	04	
Priority:	High	
Participating Actors:	User (General Manager/Junior Worker), System	
Pre Conditions:	-	
Trigger Event:	Selecting “View Vehicle Charges” from the main menu	
	User	System
Main flow of events:	2)Enter date 4) Enter time	1) Prompt for current date 3) Prompt for current time 5)Calculate charge by reducing the entry date and time from current date and time 6) Display the vehicle list with charge and id plate number 7) Go back to main menu
Alternate flow 1:		3) Display “Invalid date” 4) Go to 1)
Alternate flow 2:		5) Display “Invalid time” 6) Go to 3)
Exceptional Flow:	-	
Post-Conditions:	-	
Inclusions		
Extensions	-	

Use Case Name:	View current vehicle list	
Use Case No:	05	
Priority:	High	
Participating Actors:	User (General Manager/Junior Worker), System	
Pre Conditions:	-	
Trigger Event:	Selecting “View current vehicle list” from the main menu	
	User	System
Main flow of events:		1) <<include>> Sort current vehicles chronologically 2) Display vehicle list with vehicle plate id, entry date and time, type 7) Go back to main menu
Alternate flow:	-	
Exceptional Flow:	-	
Post-Conditions:	-	
Inclusions	Sort current vehicles chronologically	
Extensions	-	

Use Case Name:	View statistical data	
Use Case No:	06	
Priority:	High	
Participating Actors:	General Manager, System	
Pre Conditions:	-	
Trigger Event:	Selecting “Get statistical data” from the main menu	
	General Manager	System
Main flow of events:		1) Count cars, vans, motor bikes, total vehicles currently parked in the system 2) Display each vehicle type percentages 3) <<include>> Sort current vehicles chronologically 4) Display longest parked vehicle(overall), longest parked vehicle (Currently) and last parked vehicle With their id plate numbers, entry date and time and vehicle type 8) Go back to main menu
Alternate flow:	-	
Exceptional Flow:	-	
Post-Conditions:	-	
Inclusions	<<include>> Sort current vehicles chronologically	
Extensions	-	

Sequence Diagrams

- Adding a vehicle

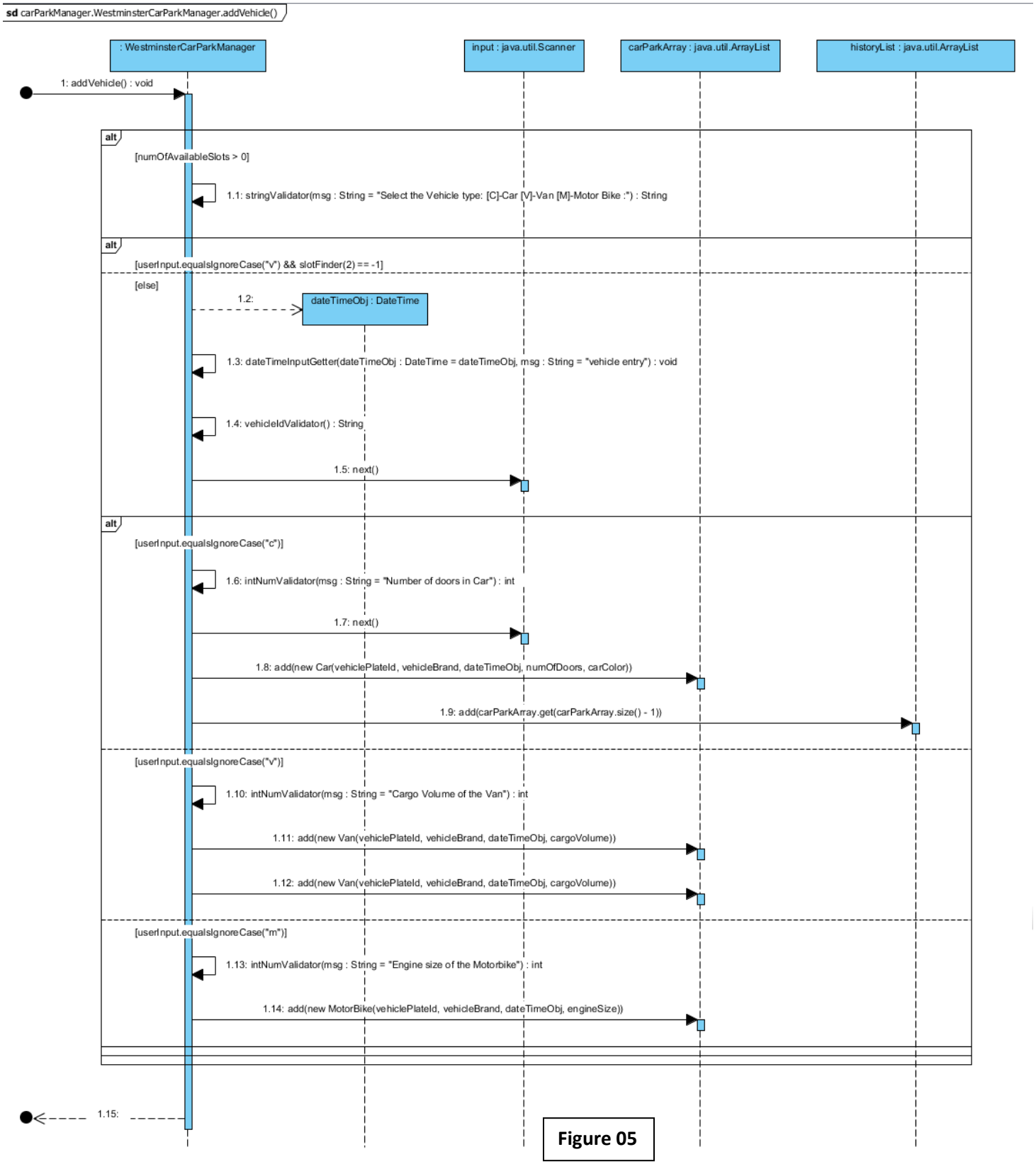

Figure 05

- Deleting a vehicle

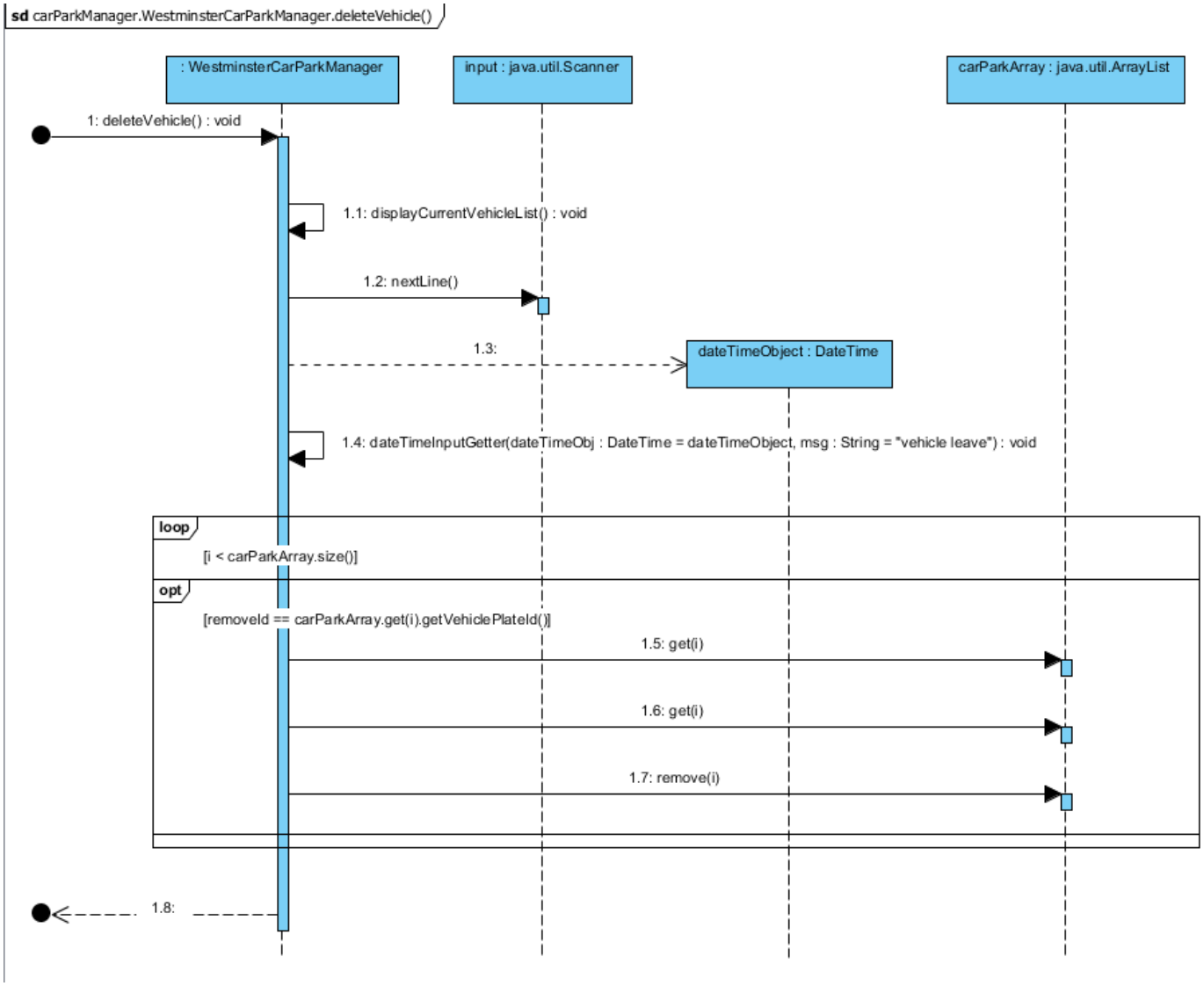

Figure 06

- Displaying the list of current vehicles in the car park

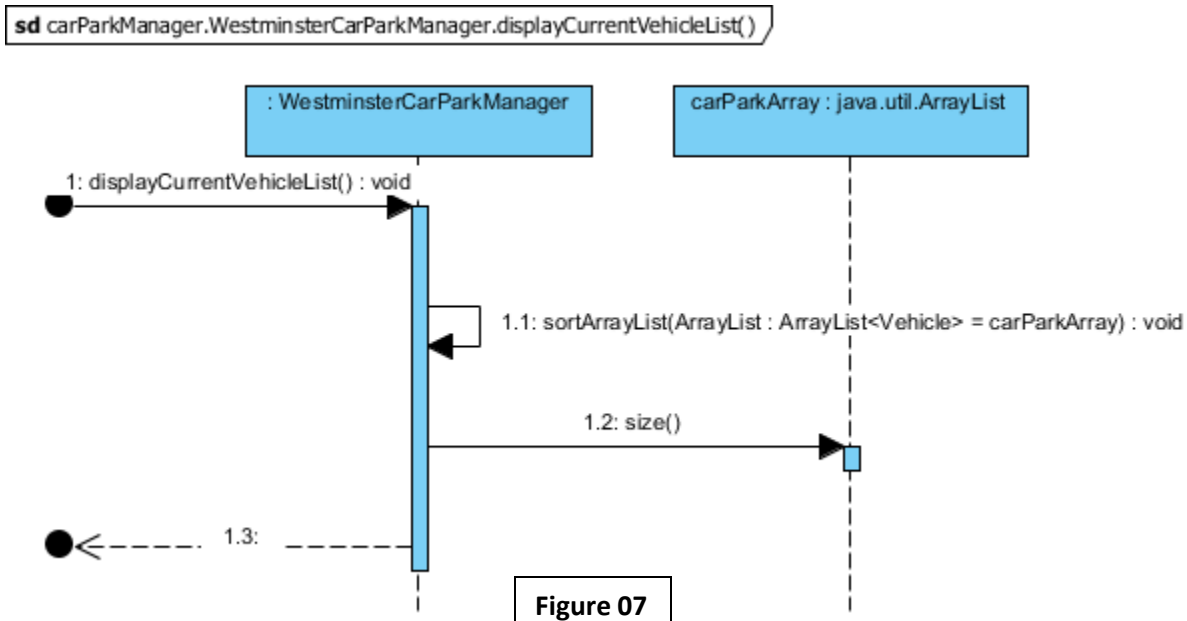

Figure 07

- Sorting the array list

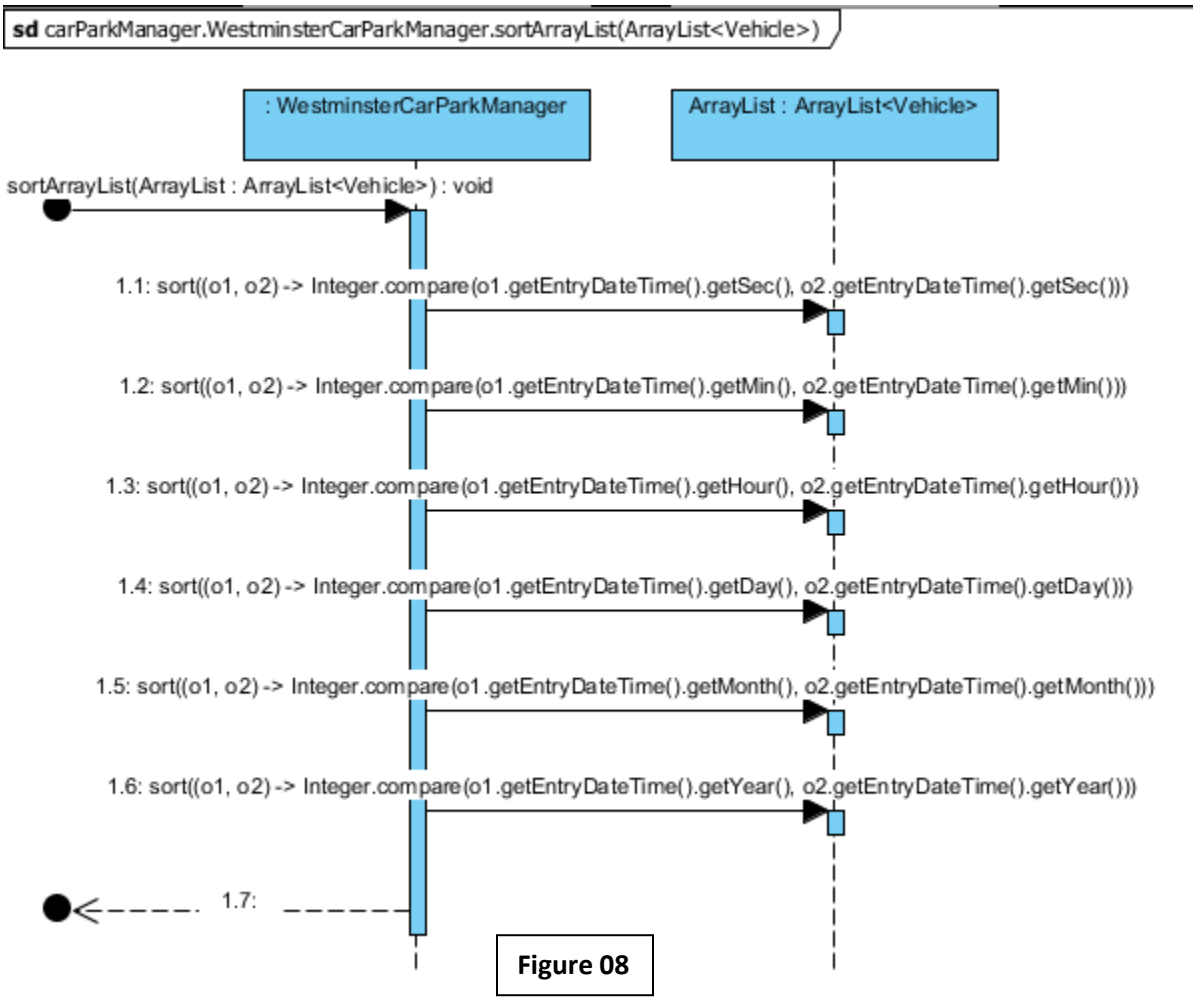

Figure 08

- Displaying the current charging for vehicles

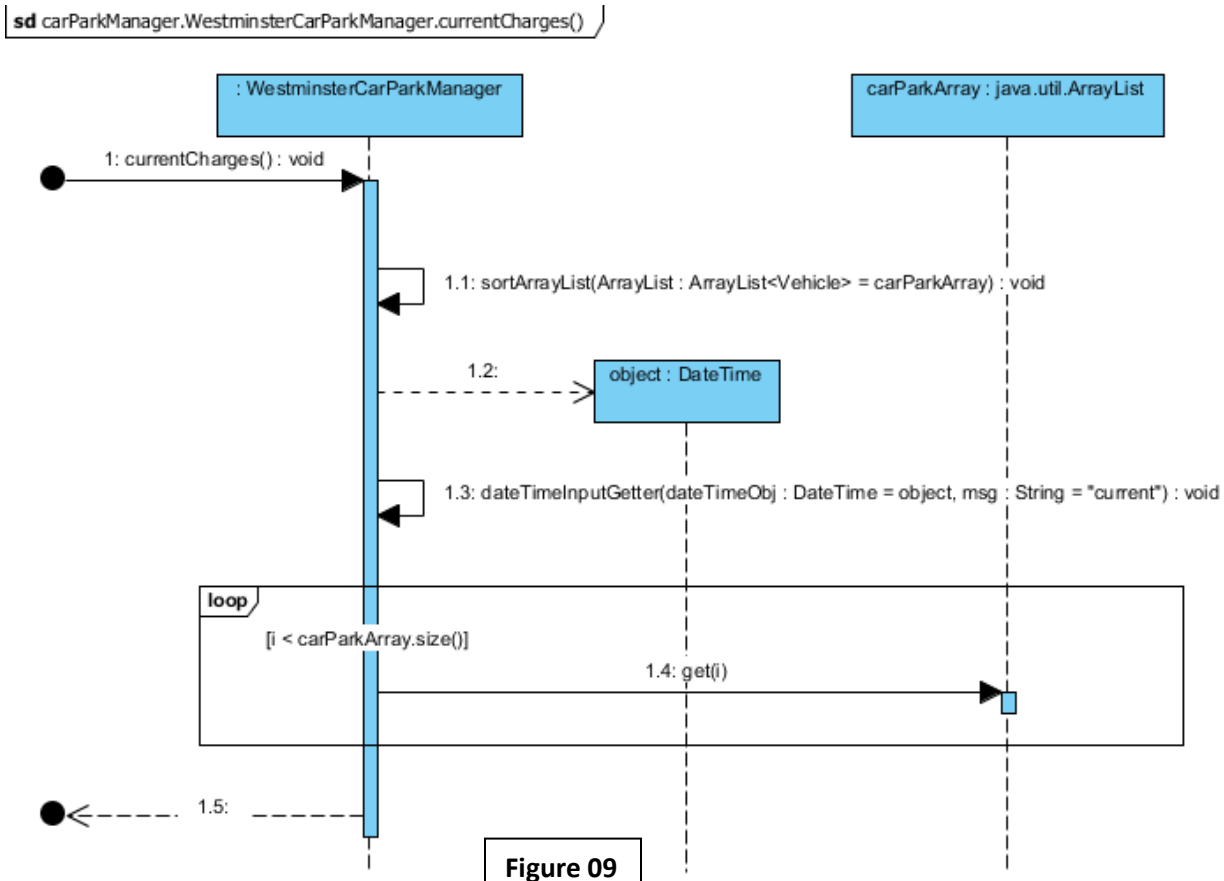

Figure 09

- Calculating the charge individually for a vehicle currently in the car park

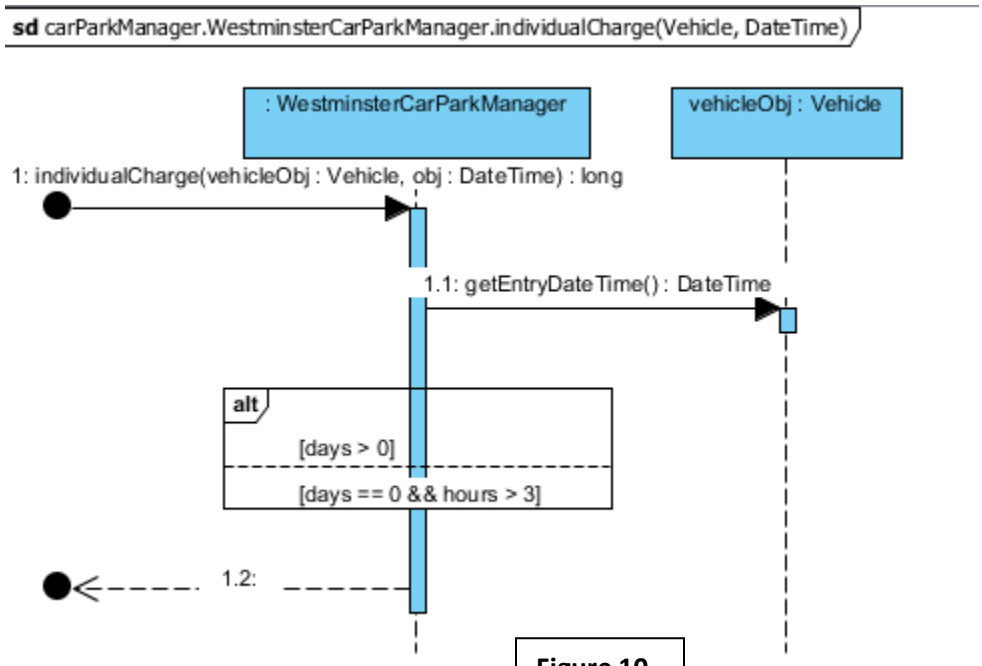

Figure 10

- Search vehicles by date

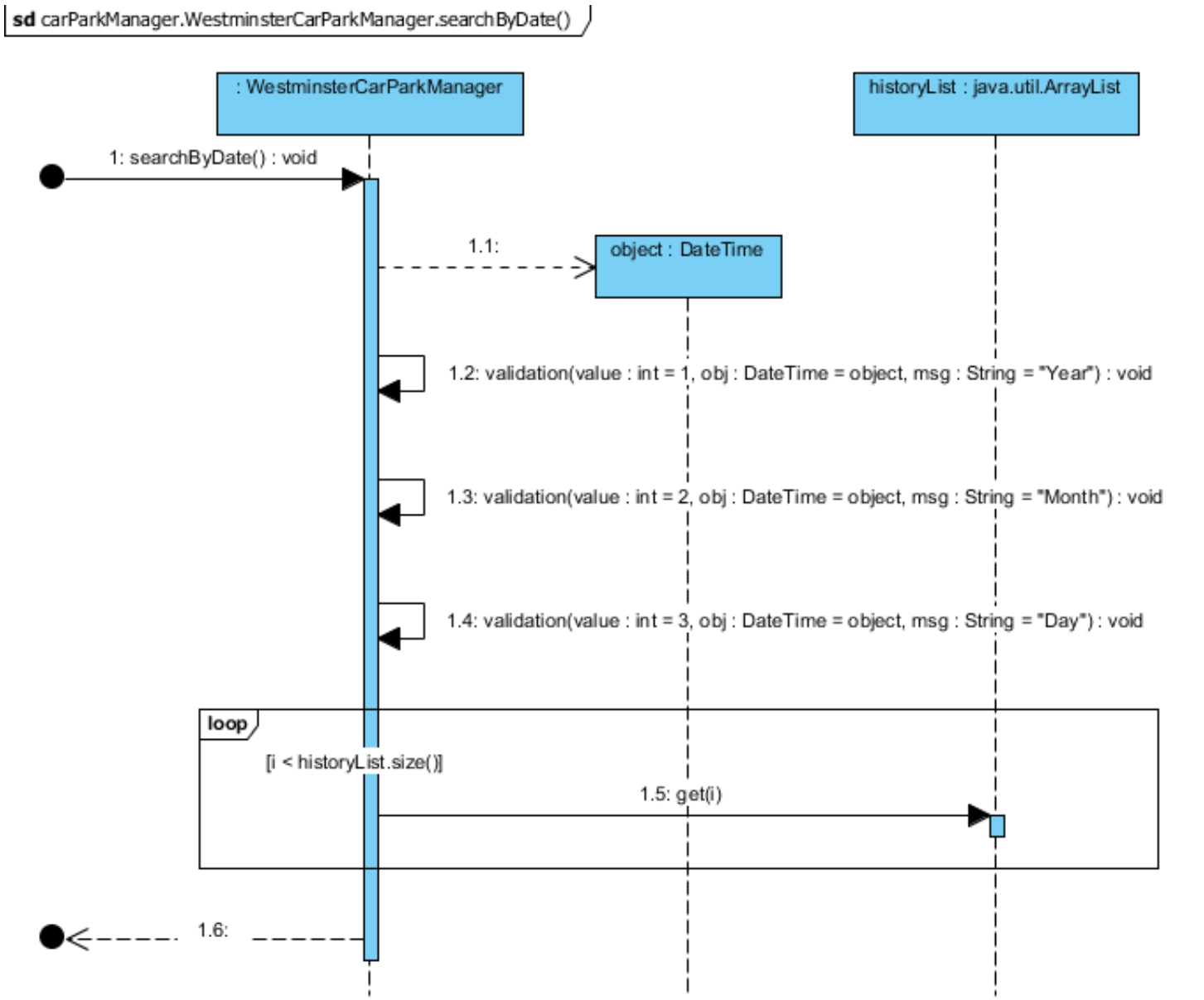

Figure 11

- Displaying statistical data

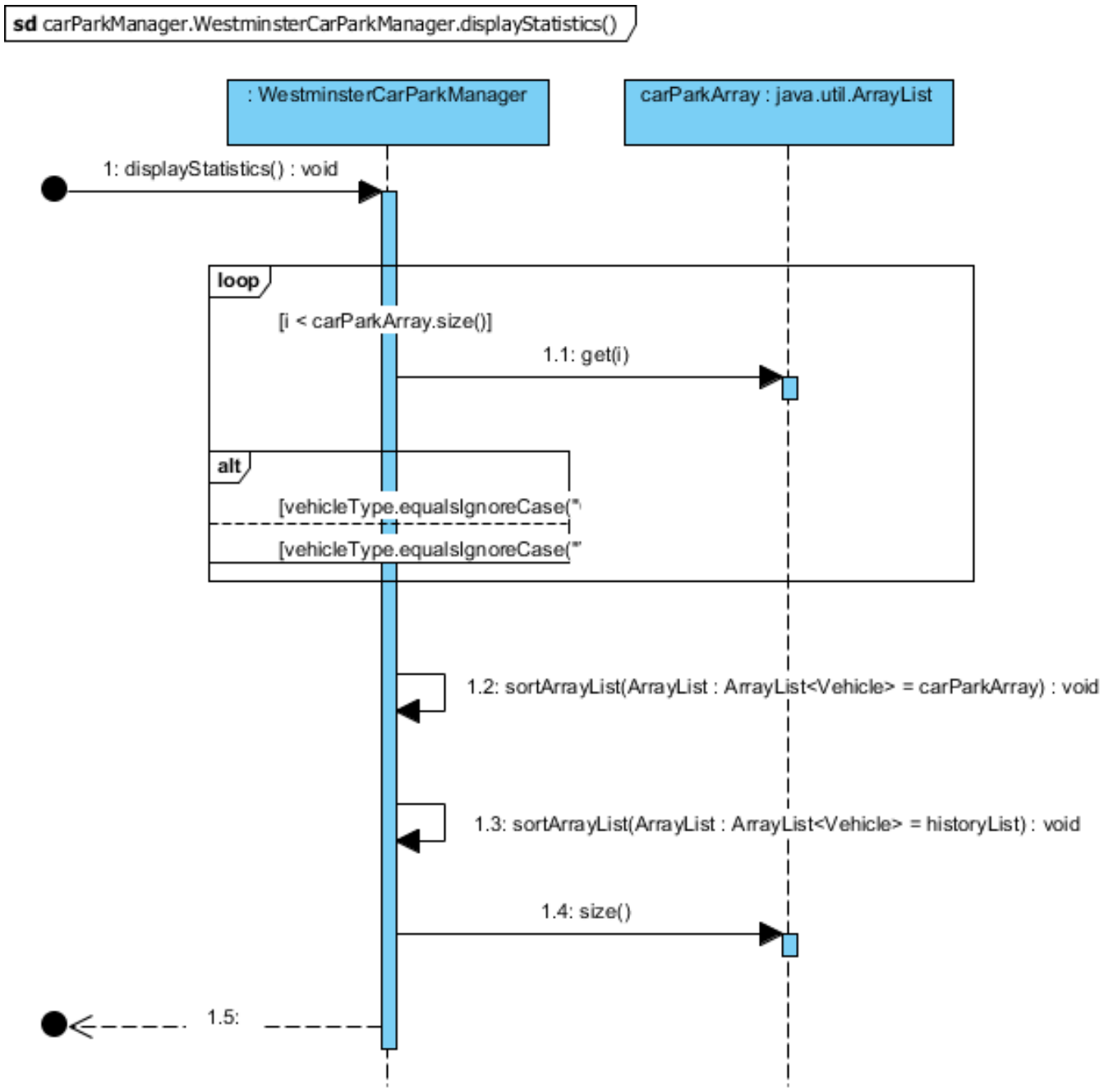

Figure 12

Testing

Black Box Testing

Test Case No.	INPUT	EXPECTED OUTPUT	ACTUAL OUTPUT	BUG
Main Menu All types of user inputs for main menu selections are tested in here				
01	6	"Invalid input"	"Invalid input"	No
02	^	"Invalid input"	"Invalid input"	No
03	Y	"Invalid input"	"Invalid input"	No
04	A	"------Add Vehicle--------"	"------Add Vehicle------"	No
05	A	"------Add Vehicle--------"	"------Add Vehicle-------"	No
06	V	"------Currently Parked Vehicle List--------"	"------Currently Parked Vehicle List------"	No
07	D	"------Delete Vehicle------"	"------Delete Vehicle--------"	No
08	C	"------Charges for currently parked vehicles------"	"------Charges for currently parked vehicles------"	No
09	S	"------Search by Date------"	"------Search by Date--------"	No
10	g	"------Statistical Data------"	"------Statistical Data------"	No
11	X	"Successfully saved data to the file"	"Successfully saved data to the file"	No
12	Enter fsdgdf as selection	"!!! Invalid Input. Input should contain only one character !!!"	"!!! Invalid Input. Input should contain only one character !!!"	No
13	Press ENTER button	"!!! Invalid Input. Input should contain only one character !!!"	"!!! Invalid Input. Input should contain only one character !!!"	No
In here all types of user inputs for "Add vehicle" function is tested All the date and time validations are also checked in here.				
01	Enter "c" as vehicle type	"Enter the entry date---------"	"Enter the entry date---------"	No
02	Enter "11" as vehicle type	"Invalid input"	"Invalid input"	No
03	Enter "0" as year	"Invalid year!"	"Invalid input"	No
04	Enter "2016" as year	"Month:"	"Month:"	No

05	Enter “fsdf” as year	“Invalid year!”	“Invalid year!”	No
06	Enter “0” as month	“Invalid month!”	“Invalid month!”	No
07	Enter “2” as month	“Day:”	“Day:”	No
08	Enter “fgg” as month	“Invalid date!”	“Invalid date!”	No
09	Enter “13” as month	“Invalid month!”	“Invalid month!”	
10	Enter “0” as date	“Invalid date!”	“Invalid date!”	No
11	Enter “3” as date	“Hour:”	“Hour:”	No
12	Enter “32” as date	“Invalid date!”	“Invalid date!”	No
13	Enter “hjk” as date	“Invalid date!”	“Invalid date!”	No
14	Enter 2 as month and 30 as date	“Invalid date!”	“Invalid date!”	No
15	Enter 4 as month and 31 as date	“Invalid date!”	“Invalid date!”	No
16	Enter 2000 as year and Enter 2 as month and 29 as date	“Hour:”	“Hour:”	No
17	Enter 2001 as year and Enter 2 as month and 29 as date	“Invalid date!”	“Invalid date!”	No
18	Enter “0” as hour	“Second:”	“Second:”	No
19	Enter “2016” as hour	“Invalid hour!”	“Invalid hour!”	No
20	Enter “fsdf” as hour	“Invalid hour!”	“Invalid hour!”	No
21	Enter “0” as minute	“Second:”	“Second:”	No
22	Enter “59” as minute	“Second:”	“Second:”	No
23	Enter “fgg” as minute	“Invalid minute!”	“Invalid minute!”	No
24	Enter “60” as minute	“Invalid minute!”	“Invalid minute!”	No
25	Enter “0” as second	"Please enter the vehicle Id plate number: "	"Please enter the vehicle Id plate number: "	No
26	Enter “3” as second	"Please enter the vehicle Id plate number: "	"Please enter the vehicle Id plate number: "	No
27	Enter “60” as second	“Invalid second!”	“Invalid second!”	No
28	Enter “hjk” as second	“Invalid second!”	“Invalid second!”	No

White Box Testing

I used "CONDITION TESTING" technique for white box testing. I gave inputs (for true and false states separately) for a particular condition and tested the condition.

No.	Test Case	Test Value	Expected Output	Actual Output	Bug
addVehicle() **IF conditions in addVehicle() function are tested**					
01	**if** (numOfAvailableSlots > 0)	numOfAvailableSlots=1	"Select the Vehicle type: [C]-Car [V]-Van [M]-Motor Bike :"	"Select the Vehicle type: [C]-Car [V]-Van [M]-Motor Bike :"	No
02	**if** (numOfAvailableSlots > 0)	numOfAvailableSlots=0	"Sorry. No available free slots in the car park"	"Sorry. No available free slots in the car park"	No
03	**if** (userInput.equalsIgnoreCase("v") && slotFinder(2) == -1)	slotFinder(2)=-1	"Sorry. No enough parking slots for a van"	"Sorry. No enough parking slots for a van"	No
04	**if** (userInput.equalsIgnoreCase("v") && slotFinder(2) == -1)	slotFinder(2)=3	"Enter the entry date---------"	"Enter the entry date---------"	No
05	**if** (userInput.equalsIgnoreCase("c"))	userInput=c	"Enter the vehicle color of Car: "	"Enter the vehicle color of Car: "	No
06	**if** (userInput.equalsIgnoreCase("c"))	userInput=d	Goes to else if part	Goes to else if part	No
07	**else if** (userInput.equalsIgnoreCase("v"))	userInput=v	"Cargo Volume of the Van"	"Cargo Volume of the Van"	No
08	**else if** (userInput.equalsIgnoreCase("v"))	userInput=d	Goes to else if part	Goes to else if part	No
09	**else if** (userInput.equalsIgnoreCase("m"))	userInput=m	"Engine size of the Motorbike"	"Engine size of the Motorbike"	No
10	**else if** (userInput.equalsIgnoreCase("m"))	userInput=d	End of the method	End of the method	No

deleteVehicle()
IF conditions in deleteVehicle() function are tested

01	`if (removeId == carParkArray.get(i).getVehiclePlateId())`	Enter a vehicle id which is currently in the car park	Remove the vehicle object from the array list	Remove the vehicle object from the array list	No
02	`if (removeId == carParkArray.get(i).getVehiclePlateId())`	Enter a vehicle id which is not in the car park	`"No vehicle found"`	`"No vehicle found"`	No

searchByDate()
IF conditions in displayQueue() function are tested

01	`if (tempObj.getYear() == object.getYear() && tempObj.getMonth() == object.getMonth() && tempObj.getDay() == object.getDay())`	Entered date has previously parked vehicles	Print related vehicle details	Print related vehicle details	No
02	`if (tempObj.getYear() == object.getYear() && tempObj.getMonth() == object.getMonth() && tempObj.getDay() == object.getDay())`	Entered date has no previously parked vehicles	`"No vehicle data found!"`	`"No vehicle data found!"`	No

displayStatistics()
IF conditions in displayStatistics() function are tested

01	`if (vehicleType.equalsIgnoreCase("Car"))`	`vehicleType=”car”`	Increase carCount variable	Increase carCount variable	No
02	`if (vehicleType.equalsIgnoreCase("Car"))`	`vehicleType=”van”`	Goes to else if part”	Goes to else if part	No

03	**else if** (vehicleType.equalsIgnoreCase("Van"))	vehicleType=”van”	Increase vanCount variable	Increase vanCount variable	No
04	**else if** (vehicleType.equalsIgnoreCase("Van"))	vehicleType=”motorbike”	Goes to else if part	Goes to else if part	No
05	**else if** (vehicleType.equalsIgnoreCase("MotorBike"))	vehicleType=”motorbike”	Increase bikeCount variable	Increase bikeCount variable	No