MULTI-STORY CAR-PARK PROGRAM

Main Assignment – 10th May 2019

Abstract

This is the documentation for the design and implementation of my main assignment – a Multi-story Car-Parking Program. It explains how I designed, developed and tested my program so that it fully complied with the brief set out to me.

Ollie Thomas - 180001541

olt13@aber.ac.uk

Contents

Introduction

For this assignment, I was asked to write a 'Multi-Story Car-Parking Program' that had to deal with five different vehicles and zones. In this report, I will break down how I approached and designed the project, describe the testing I did while I was writing the code. I will also evaluate how my program performs compared with the original functional requirements. I believe that I have met all the requirements set out in the assignment brief.

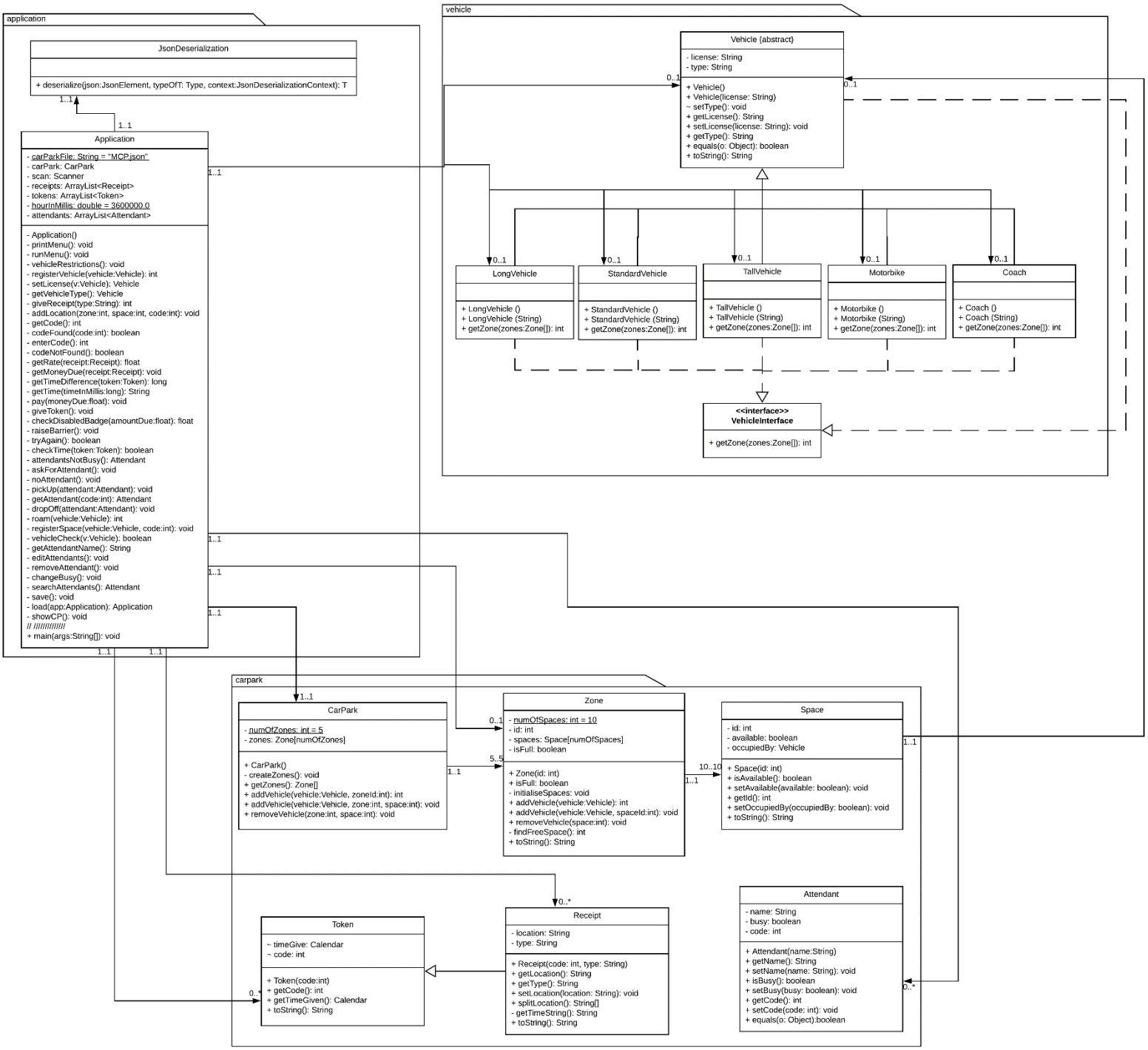
UML Case Diagram

The UML case diagram below presents the functionality that was required for my program.

Design

UML Class Diagram

The UML Class Diagram below presents all the classes (with fields and methods) that I implemented in my program and their relationships with one another. (See Appendix Diagram 1-3 for a close-up view of each package)



Class Descriptions

Application Class

The Application class is where the bulk of the program logic is written. This also includes the main method that runs the program; meaning this is where all the interaction with the user takes place. As such, it has relationships with most other classes and is the centre point for most of their interactions.

JsonDeserialization Class

This class is used when loading from the json file. Without registering the type adapter for the GsonBuilder as this, the program would not be able to distinguish between the different types of vehicle class due to the inheritance. This would cause an error, as the program would then attempt to create an object from just the Vehicle class, which is not possible as it is abstract. This class gets the type key word from the json file (e.g. "type": "vehicle.Coach") and uses that data to get the specific type of class. It only has a relationship with the Application class where it is called to aid with the loading from the file.

Vehicle Class

The Vehicle class is an abstract class (so there are no objects of it) and is the parent to all the different types of vehicle. It contains all the methods required for each of the vehicle types, which then inherit these for their own use.

LongVehicle, StandardVehicle, TallVehicle, Motorbike and Coach Class

These classes are used to make specific types of vehicle in the Application class (they can also be assigned to 'occupiedBy' in the Space class). They are also used to get which zones each type of vehicle are allowed in, and whether they are full or not. All these classes inherit from the Vehicle super class and implement VehicleInterface.

VehicleInterface Interface

The VehicleInterface interface ensures that each sub-type of vehicle that I have included has the 'getZone(zones: Zone[]): int' method. As it is fundamental to the running of the program, this also protects the program from future user interaction; forcing them to write this method for all their new vehicle types.

CarPark Class

The CarPark class builds the simulated car-park out of 5 Zone class objects and is called by the Application class to get information about, or change, the state of the car-park (such as add or remove a vehicle).

Zone Class

The Zone class builds a zone for the CarPark class. It consists of an array of 10 Space class objects. I chose 10 spaces as it will be easier to show how the program behaves when it is full. It also contains code that changes the state of the car-park. The Zone class is also used in the showCP() method in the Application class to loop through the zones in the car-park and display each one to the user.

Space Class

The Space class is used as a placeholder to store Vehicles. It has an id, a boolean to show if it is available, and an occupiedBy field of type Vehicle. It is used by the Zone class in an array to create the zone, and to tell the main program which spaces are free.

Token Class

This class holds information about the tokens that are given by the application to enable users to raise the barrier. It has a code (int) to identify it from other tokens and a timeGiven (Calendar) to enable to the program to see if the allowed "15 minutes to leave" have elapsed. The Token class is also the parent to the Receipt class, as the Receipt class also needed a code and a time stamp when it was given.

Receipt Class

The Receipt class holds information about the receipts that are given by the application. They enable the application to know how long each car has been parked for and to charge them appropriately. This class inherits from the Token class as they could share some instance variables and methods. On top of the inherited methods and fields, the Receipt class also stores a location (String) and the type of vehicle that is associated with the receipt.

Attendant Class

The Attendant class enables the attendant pool to be created in the Application class. Attendants have names and a boolean to show whether they are busy or not.

Complex Algorithm

For my most complex algorithm, I have chosen the payment system. I chose this one because there a many different aspects to it: calculate the time difference from the receipt, retrieve the rate for the specific zone, calculate the money due, check if the user is a blue badge holder, receive the payment; give change if overpayment made; ask for more if underpaid, and finally display where the user's vehicle is parked.

Pseudo-code

```
Function getMoneyDue (receipt)
        Calculate time difference with function
        Convert millis to hours and round up

        Get rate function
        Calculate money due: rate * time

        If vehicle not a coach (from receipt)
                Money due becomes result of check disabled badge
                function
        End

        Output time and money due
        Pay function
End

Function timeDifference (receipt)
        Get initial time from receipt
        Get current time
        Return current time – initial time in milliseconds
End
```

```
Function checkDisabledFunction (money due)
        If user has a blue badge
                If it is Sunday
                        Money due = 0
                Else
                        Money due halved
                End
        End
        Return money due
End

Function pay (money due)
        If money due not 0
                While not paid enough
                        Print out money due
                        Ask for money input
                        If money given = money due
                                Print thank you and break
                        Else if money given < money due
                                Ask for more money
                        Else if money given > money due
                                Give change and break
                        End
                End
        Give token to raise barrier
        End
End

Function getRate (receipt)
        Open prices file
        Get zone number from the receipt
        Retrieve rate for zone number from file

        Return rate
End
```

Testing

Test Table

ID	Requirement	Description	Inputs	Expected outputs	Pass/ Fail	Comments
1	FR1.	Adding a vehicle to an empty car-park	Type = Standard vehicle, License = OV66 MZU	Vehicle successfully added to zone 1 space 1. (See SS1)	P	
			Type = et	"Incorrect type" (See SS5)	P	
2	FR1.	Adding vehicle to a partially full car-park	Type = Standard vehicle, License = OV66 MZU	The vehicle will take the next available space. (See SS6)	P	
3	FR1.	Attempting to add a vehicle to a full zone	Type = Tall vehicle, License = FR63 ORO	The program will return an error saying no free space available.	F	I forgot to assign "isFull" to true. The program just went back to the menu.
4	FR2.	Give the user a receipt with a number code after they register their vehicle	Type = Standard vehicle, License = OV66 MZU	The program will print out the receipt: initial time, then code. (See SS1)	P	
5	FR3.	The user enters a receipt code, the program finds it in the system, displays how long they have been parked and calculates how much the user must pay	Code = 0, Blue Badge Holder = N, Payment = 1 unit	The program will find the receipt and print out how much the user needs to pay. The user will pay the correct amount. (See SS2)	P	
			Code = 2, Blue Badge Holder (BBH)	The program will find the receipt and	F	The maths didn't work, and the user

			= N, Payment = 3 unit	calculate the money due. The program should give 1 unit change. (See SS8)		wasn't given change (See SS7)
			Code = 1, BBH = N, Payment = 3 unit		P	
			Code = 1, BBH = N, Payment = 3 unit	The program will calculate the money due. It should ask for more money when the user underpays. (See SS9)	P	
			Code = 1, BBH = N, Payment = f unit	The program should give error message asking for numbers. The user can then continue entering the payment.	F	The program gives the error message, but still gives the user a token – even though they haven't paid. (See SS22)
					F	(See SS23) There is a problem with the loop
					P	(See SS24)
6	FR4.	If the user has a blue badge, they get a discount on the price of parking. The user has already registered their vehicle and has a receipt code.	BBH = Y	The program will halve the amount due for those with a blue badge. (See SS9)	F	The amount wasn't halved. (See SS10) This is because the amount due wasn't assigned to the new amount after the method was run.
					P	
			BBH = N	The amount due will stay the same. (See SS2)	P	
			BBH = YN	The program should print: "Please write y or n". (SS11)	P	

7	FR5.	The user uses the show car-park option. The car-park is semi-full.	None	The program should display a list of zones and their spaces. Each space should either have a license number or say empty. (See SS1)	F	NullPointerException was thrown (See SS14)
			None		P	
8	FR3.	The user is paying for their parking.	Code = 0	The program should report to the user where their car is parked.	F	NullPointerException as the location hadn't been saved properly; meaning it wasn't initiated.
					P	(See SS2)
9	FR6.	The user has paid for their parking, and now attempts to raise the barrier using the token code within the 15 minute time limit.	Code = fg	The program should report that the input needs to be an int. The program should report that the input needs to be an int.	F	The program produced an input mismatch exception (See SS16)
			Code = fg		P	The program successfully caught the exception (See SS15)
			Code = 0	The program should report that the barrier has been opened. (SS3)	P	
10	FR6.	The user has paid for their parking, and now attempts to raise the barrier using the token code after the 15 minute time limit.	Code = 0	The program should report that the barrier hasn't opened, and the user needs to seek assistance.	F	The barrier still opened – this is due to my number for 15 minutes in milliseconds being too big, so was allowing more time to leave.
			Code = 0		P	(See SS12)

11	FR6.	The user enters an invalid code.	Code = 4	The program should report that the code wasn't found, and that the user should try again.	P	(See SS13)
12	FR7.	The user wants to add an attendant to the pool.	Attendant name = John	The program takes in the name input and adds that attendant to the pool.	P	(See SS17)
13	FR7.	The user wants to remove an attendant from the pool.	Attendant name = Katie	The program will remove the attendant from the pool. (See SS18)	F	NullPointerException as "Katie" wasn't found in the pool, therefore a null was returned.
					P	Conditional statements were added to check for nulls that mean attendant not found.
14	FR8.1	The user wants to drop their vehicle off for an attendant to register and park it.	Vehicle = Motorbike	The program should refuse as motorbikes and coaches aren't allowed to be dropped off.	P	(See SS19)
			Vehicle = Standard	The program should let the attendant take over the registering process.	F	The program displayed the coach/motorbike error message
					P	(See SS20)
15	FR8.1.	The user has dropped their vehicle off – the attendant chooses to roam to find a space.	Vehicle = Tall	The program should wait for the attendant to find a space and register it in the system if it is free.	F	They are able to overwrite vehicles that are currently in the space.
					P	(See SS25)

16	FR8.2.	The user wants an attendant to pick up their parked vehicle.	Code = 2, BBH = N, Payment = 21	The user will have to pay for their parking, they will then give their token to the attendant who will get their car.	P	(See SS26)

Functional Requirement References

FR1. Register and park a vehicle

FR2. Give receipt with code and initial time

FR3. Enter the receipt code and pay

FR4. The blue badge holder discount

FR5. Displaying the car-park

FR6. Raise barrier if token under 15 minutes old, otherwise seek assistance

FR7. Edit the attendant pool (i.e. add or remove attendants, change whether they are busy)

FR8.1. Ask for attendant – drop off vehicle

FR8.2. Ask for attendant – pick up vehicle

Test Data Used

In order to fully test the program, I used many different types of input on all the variations that the program flow could possibly run through. This included inputting types of data into the program that it was not expecting. For example, (see Test 9: Code (int) = "fg") inputting a string when it is expecting an int; usually meaning the program would throw an InputMismatchException. Testing inputs like this meant I could make sure that errors thrown are both user-friendly and easy to understand – not like the automatic stack trace. Similarly, inputting a String of characters instead of just a character (see Test 6: BBH = "YN") can trip the program over if not handled correctly. Testing these inputs means we have fully checked how the program handles a range of different inputs. Thoroughly testing the program before it is released has also reduced the likelihood of exceptions such as a NullPointerException.

Evaluation

How I solved this assignment

My first step for completing this assignment was printing out and then annotating the brief set. This helped me to work out roughly which components would be required to complete the task. This also forced me to read the brief thoroughly and in-depth; meaning I knew what asked of me before I attempted to start programming.

Secondly, I used a flipchart to write out a series of flow diagrams each representing a different part of the functionality (see Appendix Picture 1). This enabled me to further visualise what class and methods would be required. It also helped me to break the task down into smaller segments, thus making the overall programming a much smoother task. Breaking down the functionality in this way made it much easier to create a Use Case diagram. Next, I started on a simple class diagram, as shown in Appendix Picture 2.

I then moved onto the actual programming; building on the class diagram I had drawn. I started with the basic functions – such as registering a vehicle – and then gradually expanding the program to include additional elements, until all the functions were in place. Throughout this phase, I was constantly running the code to test each little section that I had written; making sure each piece of logic worked, before I moved onto the next.

When I finished the program, I then asked a friend to test the code to ensure that there was no bias to their inputs. As they didn't know what the program was expecting, it was more likely for them to create the environment for small errors when dealing with the inputs.

What did I learn and what was difficult?

This assignment has helped me to develop my technical skills and the application of logical thought, whilst systematically tackling a task that needed to comply with a clear brief of functional requirements.

The main areas of the project that I found most difficult was at the beginning of the process, when I was first thinking about the project, deciding where to start with the brief and what route to follow. The steps outlined above (How I solved this Assignment) clearly demonstrate how I was able to overcome this challenge. A second aspect I found difficult was getting my head around the simulated nature of the program – there were a couple of times during this process when I was unsure what to display to the user as the action should have been a physical one.

What remains to be done?

I believe that I have fully met the requirements for this project, and I have shown this through the other sections of this report. I did attempt to create a graphical user interface; however, the program kept throwing exceptions that I couldn't fix when I tried to setup the JavaFX package on my laptop – so I was unable to continue with that. This was disappointing, as I should have been able to make the user interface easily; making the program more user friendly. Also, I would have liked to include a valet carwash option to the program, as this would have added a fitting extension to the assignment.

Conclusion

I believe that I have successfully fulfilled the brief to a high standard – and demonstrated this by the testing and evaluation of methods undertaken and shown in this report. I believe that I have also demonstrated an excellent understanding of the project through a well-structured report. As such, from looking at the marking matrix, I think my work deserves between 65-75%.

Appendix

Screenshots

```
What would you like to do? 1
Register Car
Vehicle Types:
  S - Standard sized (<2m tall, <5m long)
  L - Long vehicle (<3m tall, 5.1m - 6m long )
  T - Tall vehicle (<5m long, 2m - 3m tall)
  M - Motorbike
  C - Coach (<=15m long)
What type of vehicle do you have?  S
Standard sized
What is the vehicle registration number?  OV66 MZU
Receipt - Time given: 14:44:52
        Code: 0


Please park in zone 1 space 1
**Menu**
1: Register your car to park
2: Enter receipt code and pay
3: Show availability in the car-park
4: Ask for attendant
5: Raise barrier
6: Close application

Attendant options
7: Edit attendant pool
8: Change busy attendant
What would you like to do? 3
Show CP
Zone: 1, Occupied by: 1: OV66 MZU 2: Empty 3: Empty 4: Empty 5: Empty 6: Empty 7: Empty 8: Empty 9: Empty 10: Empty
Zone: 2, Occupied by: 1: Empty 2: Empty 3: Empty 4: Empty 5: Empty 6: Empty 7: Empty 8: Empty 9: Empty 10: Empty
Zone: 3, Occupied by: 1: Empty 2: Empty 3: Empty 4: Empty 5: Empty 6: Empty 7: Empty 8: Empty 9: Empty 10: Empty
Zone: 4, Occupied by: 1: Empty 2: Empty 3: Empty 4: Empty 5: Empty 6: Empty 7: Empty 8: Empty 9: Empty 10: Empty
Zone: 5, Occupied by: 1: Empty 2: Empty 3: Empty 4: Empty 5: Empty 6: Empty 7: Empty 8: Empty 9: Empty 10: Empty
```

SS1: shows a standard vehicle being added to the car-park, the user being given a receipt and being told where to park. It also shows the 'show car-park' function.

```
What would you like to do? 2
Enter receipt
Please enter your receipt code.
0
Your car is in Zone: 1 Space: 1
Are you a blue badge holder? (Y/N)
n
You have been here for: 00:09:10
1.0 units due
Please pay the amount due
1
Thank you
Here is your Token - Code: 0
You have 15 minutes to get your vehicle and use the barrier
```

SS2: shows the user entering their receipt code, paying and then receiving their exit token.

```
What would you like to do? 5
Raise barrier
Please enter your token code
0
The barrier is open, have a nice day!
```

SS3: shows the user using their token code to open the barrier within the allowed 15 minutes.

```
What would you like to do? 3
Show CP
Zone: 1, Occupied by: 1: Empty 2: Empty 3:
Zone: 2, Occupied by: 1: Empty 2: Empty 3:
Zone: 3, Occupied by: 1: Empty 2: Empty 3:
Zone: 4, Occupied by: 1: Empty 2: Empty 3:
Zone: 5, Occupied by: 1: Empty 2: Empty 3:
```

SS4: shows that the vehicle successfully left.

```
What would you like to do? 1
Register Car
Vehicle Types:
  S - Standard sized (<2m tall, <5m long)
  L - Long vehicle (<3m tall, 5.1m - 6m long )
  T - Tall vehicle (<5m long, 2m - 3m tall)
  M - Motorbike
  C - Coach (<=15m long)
What type of vehicle do you have?  et
Incorrect type
```

SS5: shows an invalid type "et" being entered and the subsequent error message. The program then asks for the type of vehicle again

```
Show CP
Zone: 1, Occupied by: 1: OV66 MZU 2: RO04 MVP 3: RF63 ORO 4: Empty
```

SS6: shows the vehicles being assigned to the next available space in the correct zone

```
What would you like to do? 2
Enter receipt
Please enter your receipt code.
2
Your car is in Zone: 1 Space: 3
Are you a blue badge holder? (Y/N)
n
You have been here for: 01:13:12
2.0 units due
Please pay the amount due
3
Here is your Token - Code: 1
You have 15 minutes to get your vehicle and use the barrier
```

SS7: shows an error in the program. The program isnt giving any change to the user when they overpay.

```
What would you like to do? 2
Enter receipt
Please enter your receipt code.
1
Your car is in Zone: 1 Space: 2
Are you a blue badge holder? (Y/N)
n
You have been here for: 01:19:09
2.0 units due
Please pay the amount due
3
Thank you, here is your change: 1.0 units
Here is your Token - Code: 2
You have 15 minutes to get your vehicle and use the barrier
```

SS8: shows the program behaving correctly when overpaid.

```
What would you like to do? 2
Enter receipt
Please enter your receipt code.
0
Your car is in Zone: 1 Space: 1
Are you a blue badge holder? (Y/N)
y
You have been here for: 03:51:32
2.0 units due
Please pay the amount due
1
Amount still left: 1.0 units
1
Thank you
Here is your Token - Code: 3
You have 15 minutes to get your vehicle and use the barrier
```

SS9: shows the user underpaying, the program updating the money given and asking for the remainder due. It also shows the discount given to disabled users, being half the amount due.

```
What would you like to do? 2
Enter receipt
Please enter your receipt code.
0
Your car is in Zone: 1 Space: 1
Are you a blue badge holder? (Y/N)
y
You have been here for: 00:00:09
1.0 units due
```

SS10: shows an error with the blue badge discount, as it hasn't halved the total cost.

```
What would you like to do? 2
Enter receipt
Please enter your receipt code.
0
Your car is in Zone: 1 Space: 1
Are you a blue badge holder? (Y/N)
yn
Pleas type y or n
Are you a blue badge holder? (Y/N)
```

SS11: shows what happens to an invalid entry

```
What would you like to do? 5
Raise barrier
Please enter your token code
1
The barrier has not opened as it has been longer than 15 minutes.
 Please seek assistance.
```

SS12: shows what happens when the 15 minute time limit to leave has elapsed.

```
What would you like to do? 5
Raise barrier
Please enter your token code
4
We don't recognise that code. Please check and try again.
```

SS13: shows what happens when the token code isn't recognised.

```
What would you like to do? 3
Show CP
Zone: 1, Occupied by: 1: null 2: null 3: null 4: null 5: null
Exception in thread "main" java.lang.NullPointerException
    at application.Application.showCP(Application.java:628)
    at application.Application.runMenu(Application.java:67)
    at application.Application.main(Application.java:637)
```

SS14: shows an example of a NullPointerException and the stack trace

```
Please enter your token code
fg
Please enter a number
```

SS15: shows the program catching an InputMismatchException

```
What would you like to do? 5
Raise barrier
Please enter your token code
fg
Exception in thread "main" java.util.InputMismatchException
    at java.base/java.util.Scanner.throwFor(Scanner.java:939)
    at java.base/java.util.Scanner.next(Scanner.java:1594)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2258)
    at java.base/java.util.Scanner.nextInt(Scanner.java:2212)
    at application.Application.raiseBarrier(Application.java:346)
    at application.Application.runMenu(Application.java:75)
    at application.Application.main(Application.java:636)
```

SS16: shows what happens without the try/catch around the int input.

```
What would you like to do? 7
Edit attendant pool
Add or remove an attendant from the pool? (A or R)
a
What is the attendants name?
John
```

SS17: shows the user adding an attendant

```
Add or remove an attendant from the pool? (A or R)
r
What is the attendants name?
Katie
That attendant wasn't found.
```

SS18: shows what happens if the user tries to remove an attendant that isn't there

```
What would you like to do? 4
Ask for attendant
Your attendant is JOHN
Would you like to Drop off or Pick up your vehicle? (D/P) Coaches and Motorbikes are not
d
Vehicle Types:
  S - Standard sized (<2m tall, <5m long)
  L - Long vehicle (<3m tall, 5.1m - 6m long )
  T - Tall vehicle (<5m long, 2m - 3m tall)
  M - Motorbike
  C - Coach (<=15m long)
What type of vehicle do you have? m
Motorbike
Coaches and motorbikes cannot be parked by an attendant. Please use the system yourself.
```

SS19: shows motorbike rider trying to drop off their motorbike with an attendant

```
What would you like to do? 4
Ask for attendant
Your attendant is JOHN
Would you like to Drop off or Pick up your vehicle? (D/P) Coaches and Motorbikes are not allowed to use this service
d
Vehicle Types:
  S - Standard sized (<2m tall, <5m long)
  L - Long vehicle (<3m tall, 5.1m - 6m long )
  T - Tall vehicle (<5m long, 2m - 3m tall)
  M - Motorbike
  C - Coach (<=15m long)
What type of vehicle do you have?  s
Standard sized
JOHN will now come register and park your vehicle for you.
Please wait for your receipt
## For Attendant Use ##
Would you like to use app or roam to find a space? R / A
a
What is the vehicle registration number?
```

SS20: shows the user asking to drop off their standard vehicle. The attendant then takes over and chooses to register the vehicle using the app. He then fills in the information using the "Register vehicle" option.

```
What would you like to do? 4
Ask for attendant
Unfortunately, there are no free attendants. Please park / collect your own car
```

SS21: shows the error message when there are no free attendants.

```
You have been here for: 00:00:04
1.0 units due
Please pay the amount due
f
Please enter a number
Here is your Token - Code: 2
You have 15 minutes to get your vehicle and use the barrier
```

SS22: shows an error with the payment system, it recognises they haven't used a number, but still gives a token.

```
Please enter a number
Please enter a number
Please enter a number
Please enter a number
Please enter a number
Please enter a number
Please enter a number
```

SS23: shows an error occuring with the loop when asking for payment

```
21.0 units due
Please pay the amount due
f
Please enter a number
21
Thank you
Here is your Token - Code: 1
```

SS24: shows the error catching in the payment system working correctly.

```
Would you like to use app or roam to find a space? R / A
r
Receipt - Time given: 12:27:45
        Code: 2

Give receipt to customer.

##Drive to find a space##
Which zone number is the space in?
1
What space number are you registering?
3
What is the vehicle registration number?  hg56 fhg
That space is taken. Please try another
Which zone number is the space in?
1
What space number are you registering?
4
What is the vehicle registration number?  hg56 fhg

Please park in zone 1 space 4
The space is successfully registered!
```

SS25: shows an attendant registering a free space through the roaming feature. The program doesn't allow them to register an occupied space.

```
Would you like to Drop off or Pick up your vehicle? (D/P) Coaches and Motorbikes are not allowed to use this service
p
Please use the program to pay and receive your token. Then give it to an attendant to retrieve your vehicle.
## Enter code ##
Please enter your receipt code.
2
Your car is in Zone: 1 Space: 3
Are you a blue badge holder? (Y/N)
n
You have been here for: 21:01:34
22.0 units due
Please pay the amount due
22
Thank you
Here is your Token - Code: 0
You have 15 minutes to get your vehicle and use the barrier
JOHN will now bring your vehicle so you can leave by using the raise barrier function.
```

SS26: shows the attendant pick up feature. The user pays, then the attendant collects the vehicle so the user can leave using the barrier function normally.

Figures

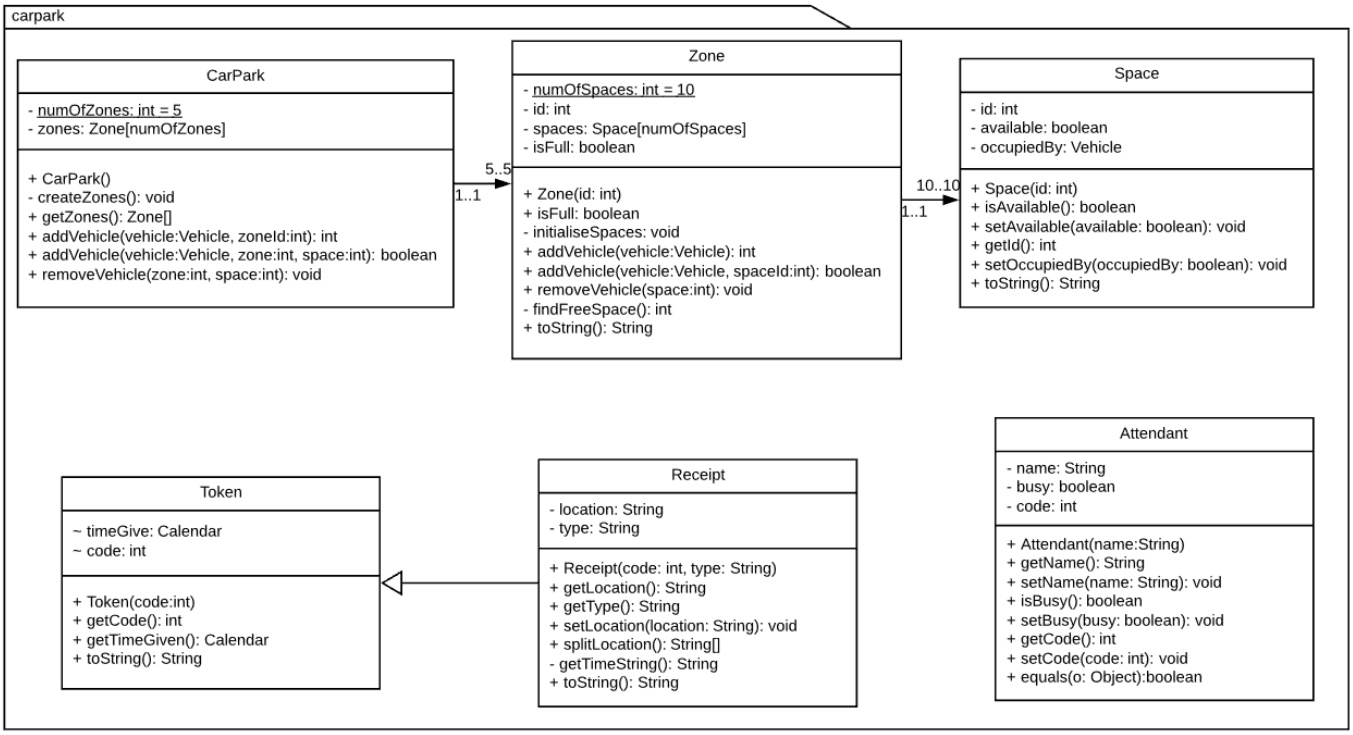


Diagram 1: shows the carpark package class diagram close-up

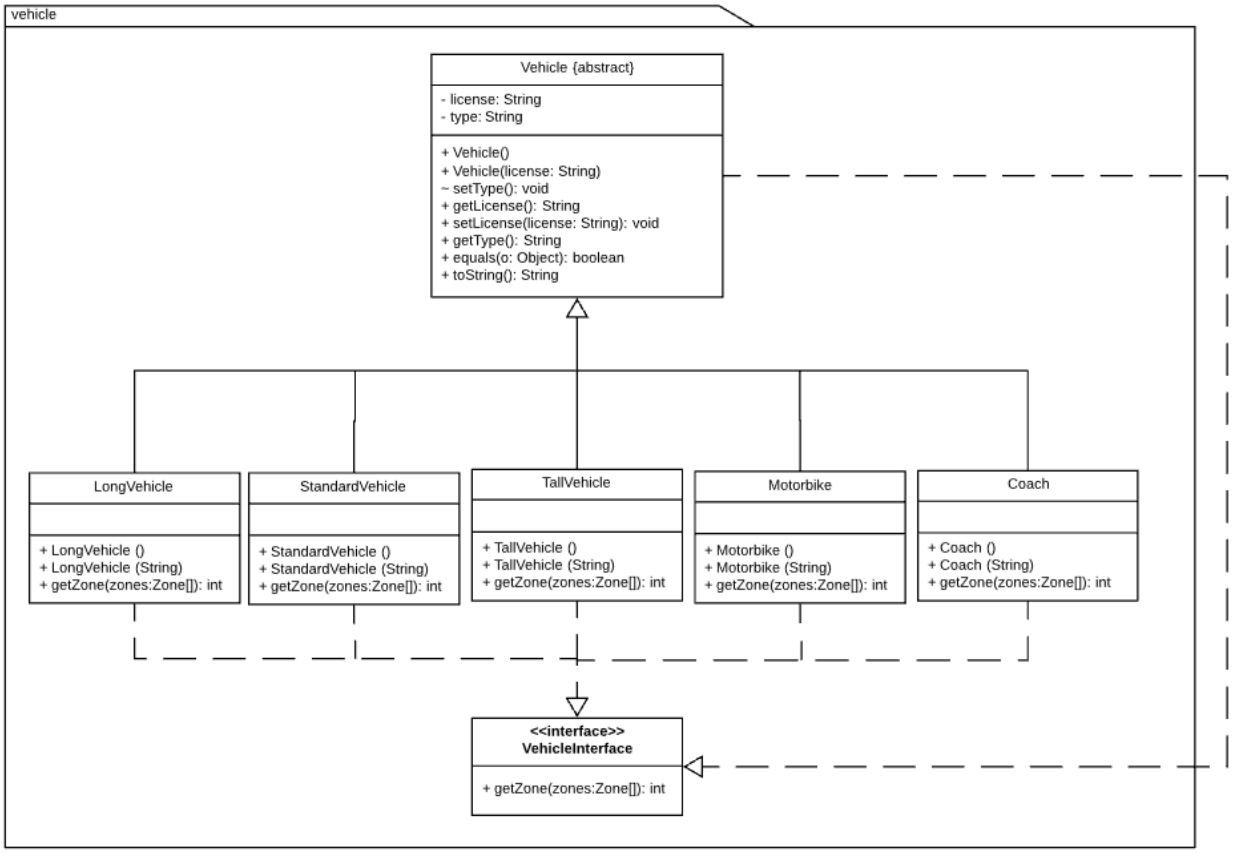


Diagram 2: shows the vehicle package class diagram close-up

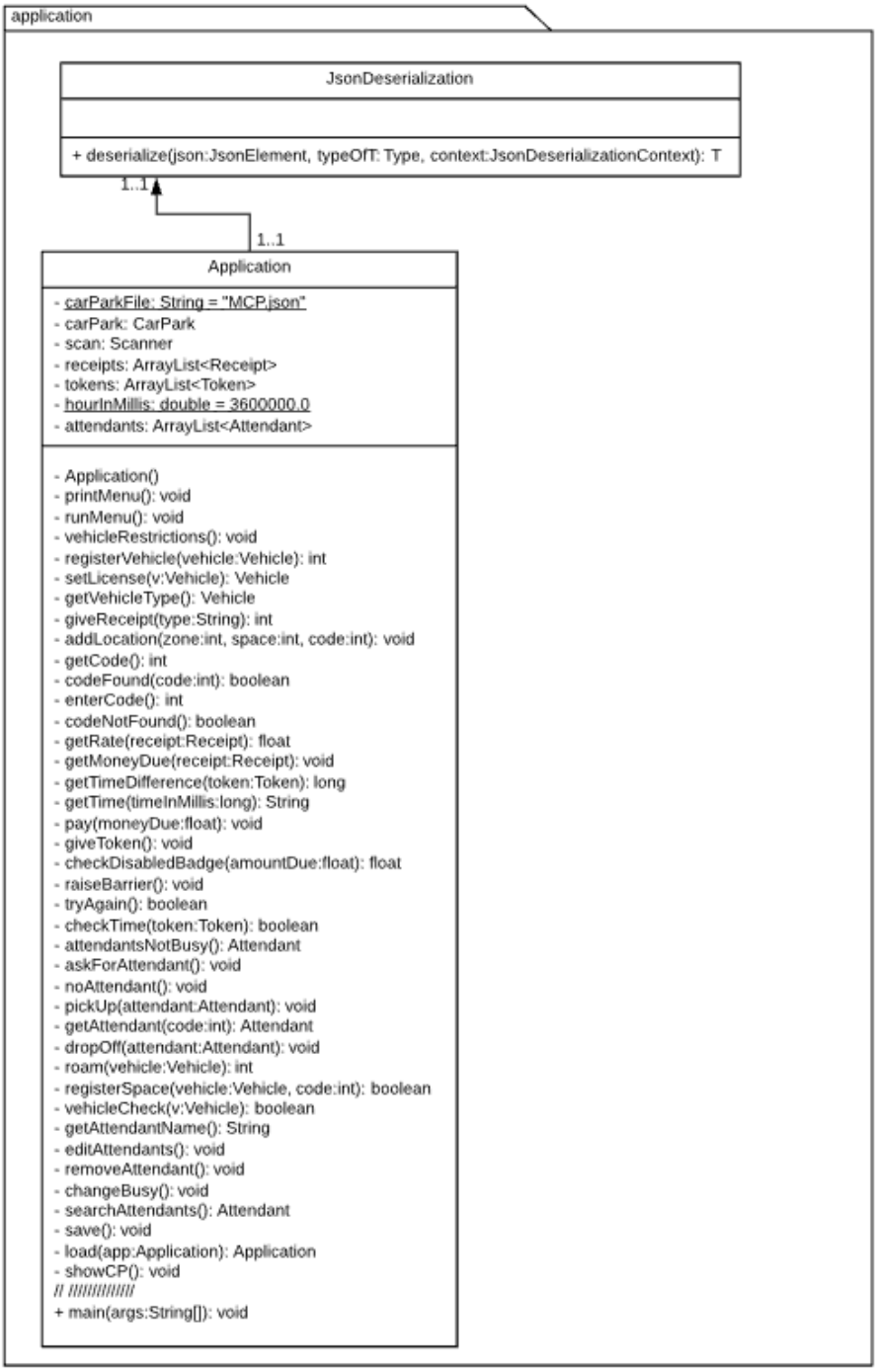
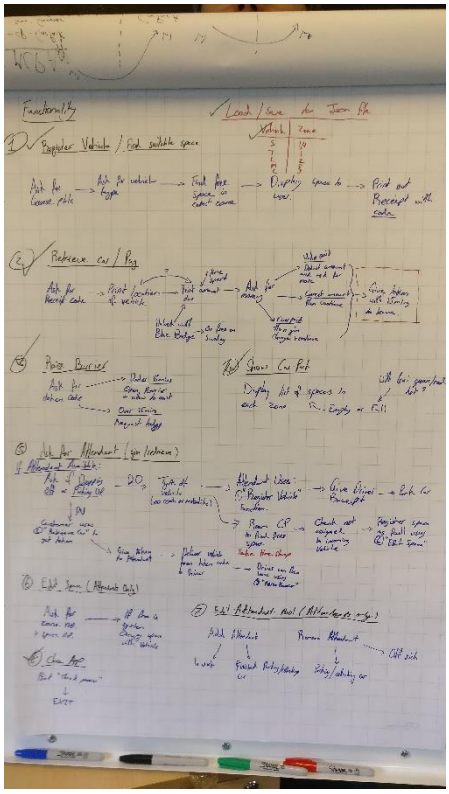
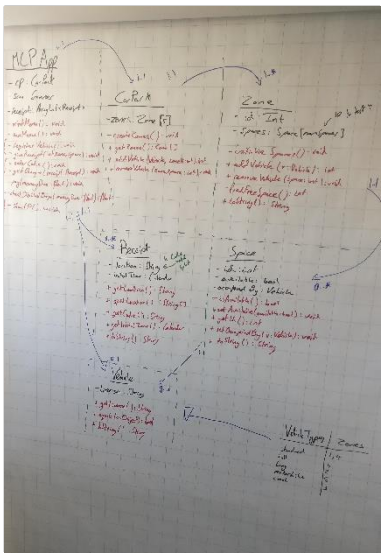


Diagram 3: shows the application package class diagram close-up



Picture 1: shows the flow diagrams I drew.



Picture 2: shows my first basic class diagram