CS12320 Individual Assignment: Multi-story Car Parking System, MCP

Veronika Karsanova, vek1@aber.ac.uk

9th of May 2019

Contents

1 Introduction

1.1 Overview

This documentation details the implementation of the MCP, Multi-Story Car Park System, outlined in the assignment brief[1].

Use-case diagram of the program, design implementation overview, and a write-up, that covers the work evaluation, accompanied the document.

Majority of the requirements were fulfilled, including some basic flair attempts. However, a number of errors and bugs was still encountered during testing.

1.2 Use-Case Diagram

See figure 1

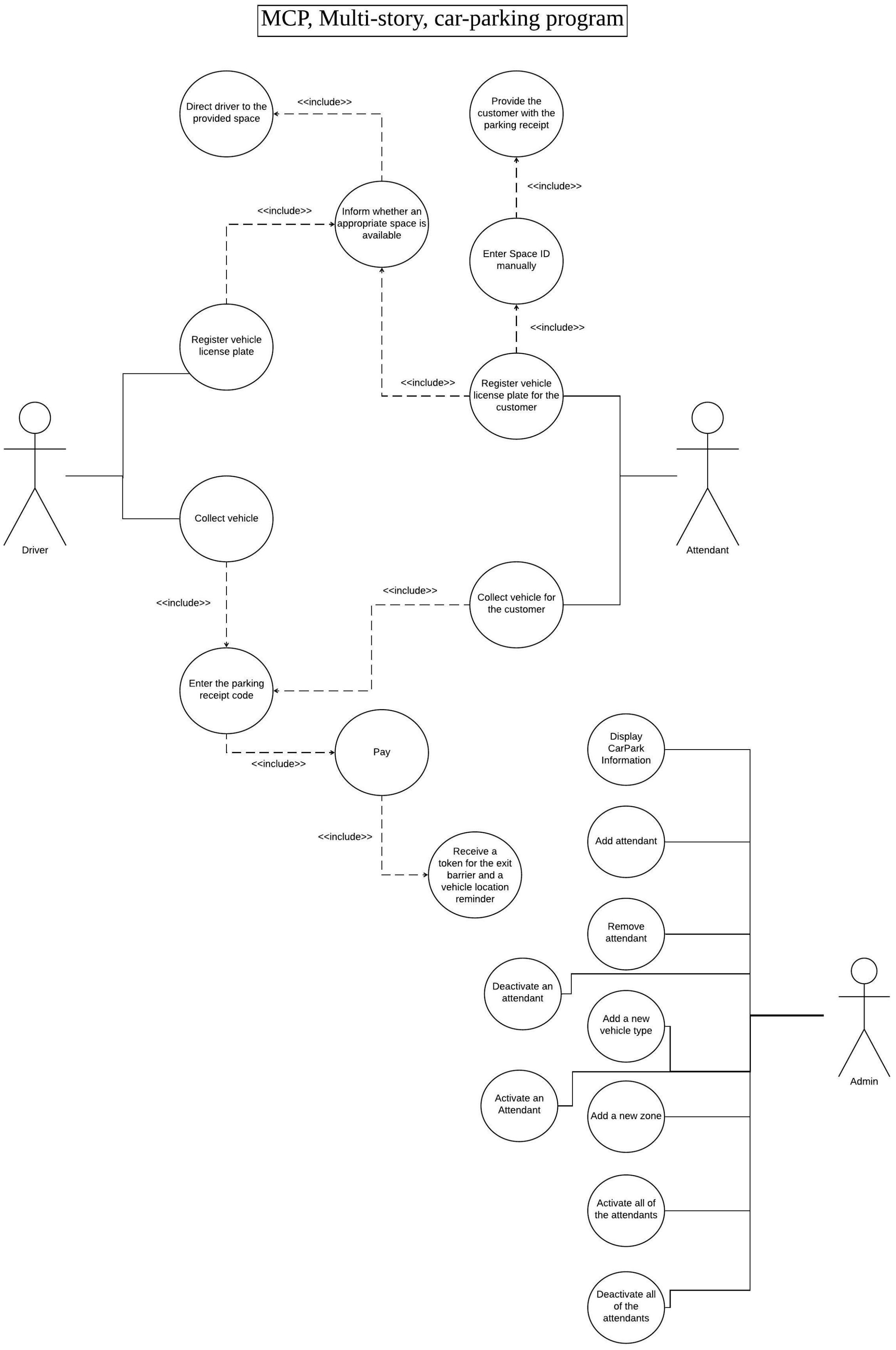

MCP, Multi-story, car-parking program
Direct driver to the provided space
<<include>>
Provide the customer with the parking receipt
<<include>>
Inform whether an appropriate space is available
<<include>>
Enter Space ID manually
<<include>>
Register vehicle license plate
<<include>>
Register vehicle license plate for the customer
Driver
Attendant
Collect vehicle
<<include>>
<<include>>
Collect vehicle for the customer
Enter the parking receipt code
Pay
<<include>>
<<include>>
Receive a token for the exit barrier and a vehicle location reminder
Display CarPark Information
Add attendant
Remove attendant
Deactivate an attendant
Add a new vehicle type
Admin
Activate an Attendant
Add a new zone
Activate all of the attendants
Deactivate all of the attendants

Figure 1: Use-Case Diagram

2 Design

2.1 UML Class Diagram

See figure 2.

2.2 Description of classes and their relationships

The vehicles package was one of the first and main things created, since putting all the vehicle types in one cohesive group made sense from the point of the program organization. Somehow, I want to point out, that I would do it differently if it was not for the sake of having inherited vehicle type classes (e.g. StandardSized) and the possibility of creating objects of any other vehicle types if there is such need. I tend to believe that enums would be more efficient, based on the brief context, since current subclasses do not have any unique instance variables and differ only by the default values of height and length.

Vehicle class – this class is determined as the superclass for all of the supported vehicle types and stores the license plate number in a String as its instance variable. In terms of methods and its use, it is fairly basic with the default and normal constructors, a number of getters, setters and a toString() method for printing the license plate number.

StandardSized class – one of the subclasses of vehicle class, represents the vehicles up to 2 meters in height and 4.9 meters in length.

HigherSized class – one of the subclasses of vehicle class, represents the vehicles over 2 meters, up to 3 meters in height and 5 meters in length.

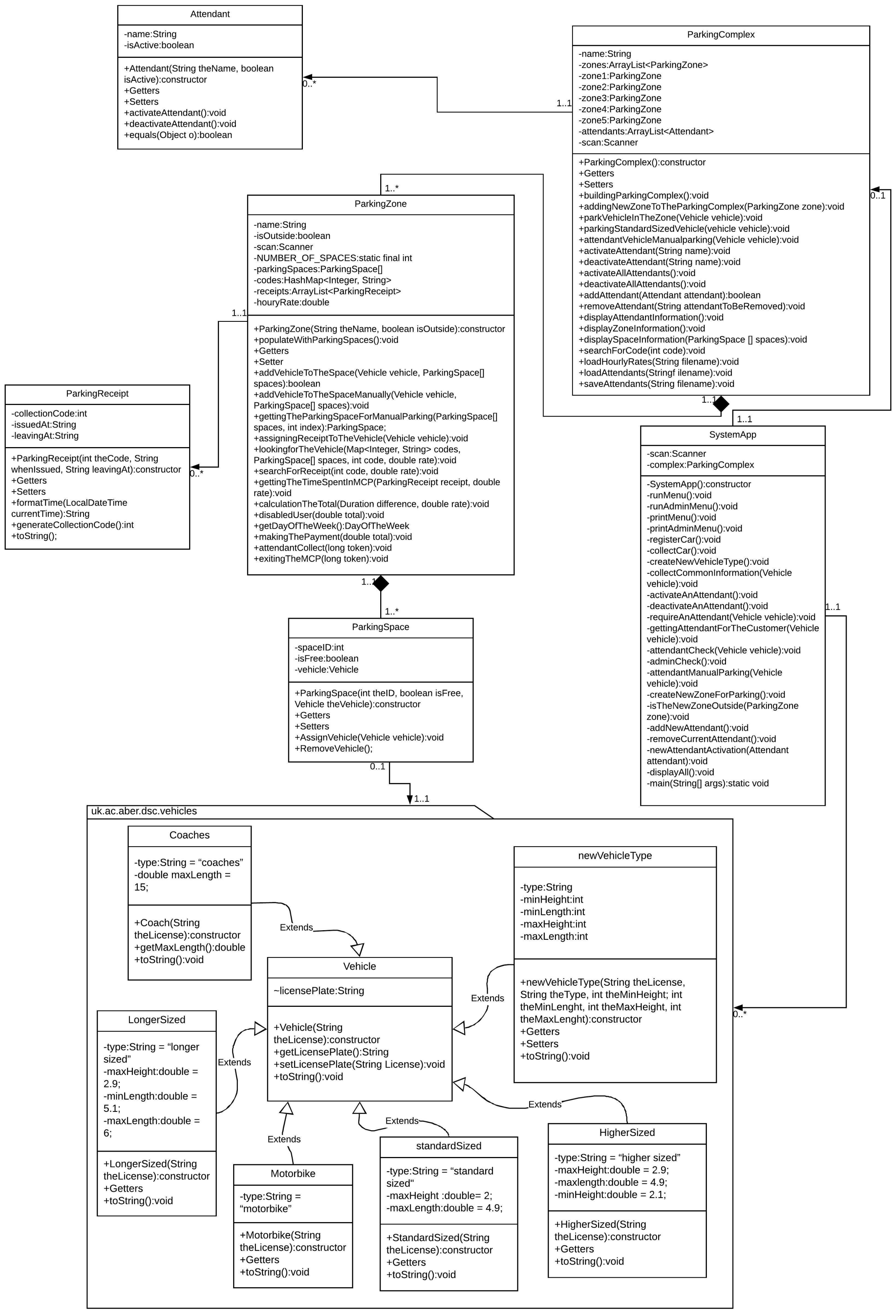
LongerSized class – one of the subclasses of vehicle class, represents the vehicles up to 2.9 meters in height and 5.1 meters to 6 meters in length.

Coach class – one of the subclasses of vehicle class, represents the coaches, vehicles up to 15 meters in length and any height.

Motorbike class – one of the subclasses of vehicle class, represents the motorbikes.

NewVehicleType class – one of the subclasses of vehicle class, used to create new vehicles if those are required. Stores the minimum and maximum height and length in the instance variables, as well as the name. Getters and Setters implemented.

ParkingSpace class – the lowest class in the parking complex hierarchy. Stores the information about its ID as an integer, availability as a boolean and the vehicle parked at the space. Apart from standard getters and setters has a method assignVehicle(), which assigns a vehicle to space and changes the availability to "false", and removeVehicle(), which does the exact opposite.



Attendant
-name:String
-isActive:boolean
+Attendant(String theName, boolean isActive):constructor
+Getters
+Setters
+activateAttendant():void
+deactivateAttendant():void
+equals(Object o):boolean
0..*
1..1
ParkingComplex
-name:String
-zones:ArrayList<ParkingZone>
-zone1:ParkingZone
-zone2:ParkingZone
-zone3:ParkingZone
-zone4:ParkingZone
-zone5:ParkingZone
-attendants:ArrayList<Attendant>
-scan:Scanner
+ParkingComplex():constructor
+Getters
+Setters
+buildingParkingComplex():void
+addingNewZoneToTheParkingComplex(ParkingZone zone):void
+parkVehicleInTheZone(Vehicle vehicle):void
+parkingStandardSizedVehicle(vehicle vehicle):void
+attendantVehicleManualparking(Vehicle vehicle):void
+activateAttendant(String name):void
+deactivateAttendant(String name):void
+activateAllAttendants():void
+deactivateAllAttendants():void
+addAttendant(Attendant attendant):boolean
+removeAttendant(String attendantToBeRemoved):void
+displayAttendantInformation():void
+displayZoneInformation():void
+displaySpaceInformation(ParkingSpace [] spaces):void
+searchForCode(int code):void
+loadHourlyRates(String filename):void
+loadAttendants(Stringf ilename):void
+saveAttendants(String filename):void
0..1
1..1
1..1
1..*
ParkingZone
-name:String
-isOutside:boolean
-scan:Scanner
-NUMBER_OF_SPACES:static final int
-parkingSpaces:ParkingSpace[]
-codes:HashMap<Integer, String>
-receipts:ArrayList<ParkingReceipt>
-houryRate:double
+ParkingZone(String theName, boolean isOutside):constructor
+populateWithParkingSpaces():void
+Getters
+Setter
+addVehicleToTheSpace(Vehicle vehicle, ParkingSpace[] spaces):boolean
+addVehicleToTheSpaceManually(Vehicle vehicle, ParkingSpace[] spaces):void
+gettingTheParkingSpaceForManualParking(ParkingSpace[] spaces, int index):ParkingSpace;
+assigningReceiptToTheVehicle(Vehicle vehicle):void
+lookingforTheVehicle(Map<Integer, String> codes, ParkingSpace[] spaces, int code, double rate):void
+searchForReceipt(int code, double rate):void
+gettingTheTimeSpentInMCP(ParkingReceipt receipt, double rate):void
+calculationTheTotal(Duration difference, double rate):void
+disabledUser(double total):void
+getDayOfTheWeek():DayOfTheWeek
+makingThePayment(double total):void
+attendantCollect(long token):void
+exitingTheMCP(long token):void
1..1
0..*
ParkingReceipt
-collectionCode:int
-issuedAt:String
-leavingAt:String
+ParkingReceipt(int theCode, String whenIssued, String leavingAt):constructor
+Getters
+Setters
+formatTime(LocalDateTime currentTime):String
+generateCollectionCode():int
+toString();
SystemApp
-scan:Scanner
-complex:ParkingComplex
-SystemApp():constructor
-runMenu():void
-runAdminMenu():void
-printMenu():void
-printAdminMenu():void
-registerCar():void
-collectCar():void
-createNewVehicleType():void
-collectCommonInformation(Vehicle vehicle):void
-activateAnAttendant():void
-deactivateAnAttendant():void
-requireAnAttendant(Vehicle vehicle):void
-gettingAttendantForTheCustomer(Vehicle vehicle):void
-attendantCheck(Vehicle vehicle):void
-adminCheck():void
-attendantManualParking(Vehicle vehicle):void
-createNewZoneForParking():void
-isTheNewZoneOutside(ParkingZone zone):void
-addNewAttendant():void
-removeCurrentAttendant():void
-newAttendantActivation(Attendant attendant):void
-displayAll():void
-main(String[] args):static void
1..1
1..1
1..*
ParkingSpace
-spaceID:int
-isFree:boolean
-vehicle:Vehicle
+ParkingSpace(int theID, boolean isFree, Vehicle theVehicle):constructor
+Getters
+Setters
+AssignVehicle(Vehicle vehicle):void
+RemoveVehicle();
0..1
1..1
0..*
uk.ac.aber.dsc.vehicles
Coaches
-type:String = “coaches”
-double maxLength = 15;
+Coach(String theLicense):constructor
+getMaxLength():double
+toString():void
Extends
newVehicleType
-type:String
-minHeight:int
-minLength:int
-maxHeight:int
-maxLength:int
+newVehicleType(String theLicense, String theType, int theMinHeight; int theMinLenght, int theMaxHeight, int theMaxLenght):constructor
+Getters
+Setters
+toString():void
Extends
Vehicle
~licensePlate:String
+Vehicle(String theLicense):constructor
+getLicensePlate():String
+setLicensePlate(String License):void
+toString():void
LongerSized
-type:String = “longer sized”
-maxHeight:double = 2.9;
-minLength:double = 5.1;
-maxLength:double = 6;
+LongerSized(String theLicense):constructor
+Getters
+toString():void
Extends
Extends
Extends
Extends
Motorbike
-type:String = “motorbike”
+Motorbike(String theLicense):constructor
+Getters
+toString():void
standardSized
-type:String = “standard sized”
-maxHeight :double= 2;
-maxLength:double = 4.9;
+StandardSized(String theLicense):constructor
+Getters
+toString():void
HigherSized
-type:String = “higher sized”
-maxHeight:double = 2.9;
-maxlength:double = 4.9;
-minHeight:double = 2.1;
+HigherSized(String theLicense):constructor
+Getters
+toString():void

[!htb]

Figure 2: UML Class Diagram

ParkingZone class – the medium class in the parking complex hierarchy. Stores the information about its name (e.g. Zone 1), whether it is placed outside, a fixed array of ParkingSpaces with a constant for their number, a HaspMap of collectionCode – licensePlateNumber pairs, an ArrayList of receipts and a double to represent the hourly rate. Apart from standard getters and setters has a method addVehicleToTheSpace(), which takes a vehicle and an array of spaces, uses the for loop to go through the array and assigns the vehicle to the first available space. Within this method, we are calling another one, assigningReceiptToTheVehicle(), which handles the storing of the collection code and the license number. Method addVehicleToTheSpaceManually() is operated by the attendants who want to park the vehicle by roaming through the car park. This is implemented by first allowing them to choose the zone in the Parking Complex class and then the ID of the space they wish to park the vehicle. In order to perform collection, we are lookingForTheVehicle(), which goes through the Code – License pairs to find a match with the code inputted by the user. We then search for the receipt in the ArrayList of receipts, comparing the time when the vehicle was parked to the one now, the time it is being collected, and then passing the difference to calculatingTheTotal(), which calculates the amount customer will need to pay. Methods disabledUser() and getDayOfTheWeek() are in charge of the special cases, such as a halved fee for disabled users. Once the total is passed to makingThePayment() and the payments are processed, the user gets to choose whether they require an attendant to collect the vehicle or they will do it themselves. The last and final method, exitingTheMCP(), checks whether the token is still valid and, if it is so, removes the vehicle from its space; otherwise – prints an error message.

ParkingComplex class – the highest class in the parking complex hierarchy. Has a name and unites the pre-created parking zones in a separate ArrayList. Apart from standard getters and setters has a method parkVehicleInTheZone(), which takes a vehicle as a parameter and operates a switch case based on its class. Depending on the vehicle type, it will then call the addVehicleToTheSpace() method in order to assign it to one of the created zones. In addition to that, there is a parkingStandardSizedVehicle(), which allows the user to choose whether they want to be parked outside, in Zone 1, or inside, in Zone 4. Method attendantVehicleManualParking() does the same thing, but allows the attendant to pick the zone for a vehicle to be parked – we assume that attendants know their job well, and will not mess with the vehicle restrictions per each zone. A number of methods, such as activateAnAttendant(), deactivateAnAttendant(), activateAllAttendants(), deactivateAllAttendants(), addAttendant(), removeAttendant(), dislayAttendantsInformation, displayZoneInformation(), displaySpaceInformation(), searchForCode() are part of this class. Load of the for hourly rates happens here, as well as saving and loading of the attendants.

ParkingReceipt class – represents the parking receipt which is generated once the vehicle is parked. Contains the collectionCode, which will be used for the vehicle release, when the driver is back. Also uses a class from the java.time package, LocalDateTime, for variables which store the type of arrival and departure of the vehicle.

Attendant class – represent the car-park employee, garage attendant. Contains the name of the attendant and whether they are active or not. Apart from standard getters and setter had the activateAttendant() and deactivateAttendant() method, as well as equals for the check of uniqueness, when the admin adds a new attendant to the pool.

SystemApp class – the main class of the program, the application class. Handles the requests from users by navigating through a number of methods, which are called from the switch case.

2.3 Pseudocode example

Calculating the total using amount of time spend in the car park and hourly rate of the zone

```
seconds = timeSpent in seconds;
hours = seconds / 3600;
seconds -= hours * 3600;
long minutes = seconds / 60;
seconds -= (minutes * 60);
double total; // total that user needs to pay
if dayOfTheWeek == SUNDAY then
    print "Parking is free today!";
    token = millis();
    exitMCP();
else
    if minutes or seconds ¿ 0 then
        roundup:
        total = (hours + 1) * rateOfTheZone;
        CheckForDisabledUser(token);
    else {minutes or seconds == 0}
        total = hours * rate;
        CheckForDisabledUser(token);
    end if
end if
```

3 Testing

3.1 Test table

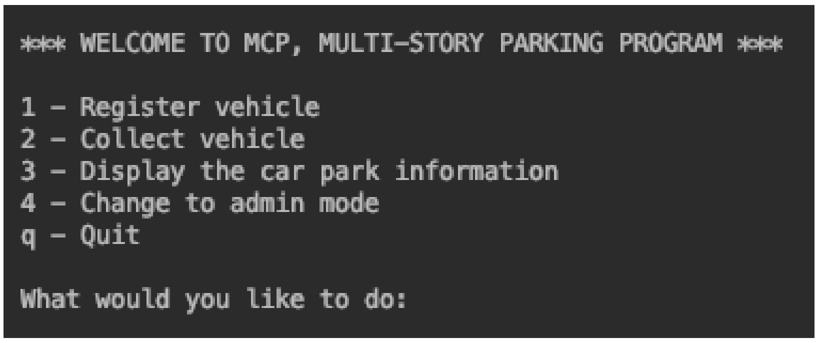
ID	Requirement	Description	Inputs	Expected outputs	Pass / Fail	Comments
A1.1	FR1	The work of the command-line menu: option input.	1 (option 1: register vehicle)	Program offers to choose the type of vehicle being registered.	P	
			Q	Program exit.	P	
			0	Prints out a default message from the switch and then the menu.	P	
A1.2	FR2	Register vehicle menu option: choosing vehicle type.	Enter a true option: 1 (a car or a small van)	Asks if I require an attendant.	P	
			Enter non-option: 0	Prints out a default message and then the menu.	P	
			Enter char: “A”	Prints out an error message for the InputMismatchException.	P	
A1.3	FR3	Y/N answer: Do you require an attendant?	Enter “N” for no;	Asks for the license plate number of the vehicle being registered.	P	
			Enter “O” for abnormal input.	Prints out an error message: “answer cannot be processed”	P	
			For “Y” aka yes see A1.6			
A1.4	FR4	The license plate number of the vehicle input and its assignment to the suitable zone (according to the brief).	Enter a string for a car or small van: “JQDWIE”	License plate number excepted. Program asks whether the user wants to be parked outside or not, since StandardSized vehicles can be parked in zone1 and zone4.	P	

			No abnormal input, since a String			
A1.5	FR5	Y/N: Do we want to be parked outside?	Enter "Y" for yes as a normal input;	Message about where the vehicle was parked (zone and space id) and the receipt with time of issue and collection code.	P	
			Enter "0" for abnormal input.	Prints out an error message: "answer cannot be processed"	P	
			Enter "N" for no as a normal input;	Message about where the vehicle was parked (zone and space id) and the receipt with time of issue and collection code.	P	
A1.6	FR6	Y/N answer: Do you require an attendant?	Enter "Y" for yes	Display message that there are not attendants available if attendants pool is empty.	P	
			Enter "Y" for yes	Security check. Offers to enter password for further actions.	P	
A1.7	FR7	Security check: input password.	Enter: correct password	Enter license plate: this time from the attendant's side (see A1.4)	P	
			Enter: incorrect password	Error message: "incorrect password" and offer to try again.	P	
A1.8	FR8	Y/N: Attendant is choosing the spot manually?	Enter: "N" for "no".	Automatically parks it, the same way shown in A1.5	P	
			Enter "Y" for "yes":	Lists the zone info and offers to choose zone to attendant (we assume that he will respect the vehicle type regulations)	P	

A1.9	FR9	Manual zone input for the attendants.	Enter int number in a range 1-3: normal input	Offers to enter the space ID next.	P	
			Enter int out of range: 0	Default message: answer cannot be "processed"	P	
			Enter char: A	Error message: "please, input valid zone number".	P	
A1.10	FR10	Choosing space ID manually.	Enter the space ID of the range and free space:	See A1.5 for normal input.	P	
			Enter int out of range: 0	Default message: answer cannot be "processed"	P	
			Enter char: A	Error message: "please, input valid zone number".	P	
B1.1	FR11	Option menu: collect vehicle. Input collection code.	Normal input: given collection code.	Asks whether user is a disabled user, halves the fee if yes.		
			Abnormal input: non existing collection code	Ignores the value, prints the menu again.	P	
			Abnormal input: string	Error message: "please, input valid zone number".	P	
B1.2	FR12	Y/N: Disabled user?	"N" for "No":	Pay the total amount for the stay in the car park.	P	
			"Y" for "Yes":	Same, but total amount is halved.	P	
			Abnormal inputs:	Message: "Input valid answer"	P	

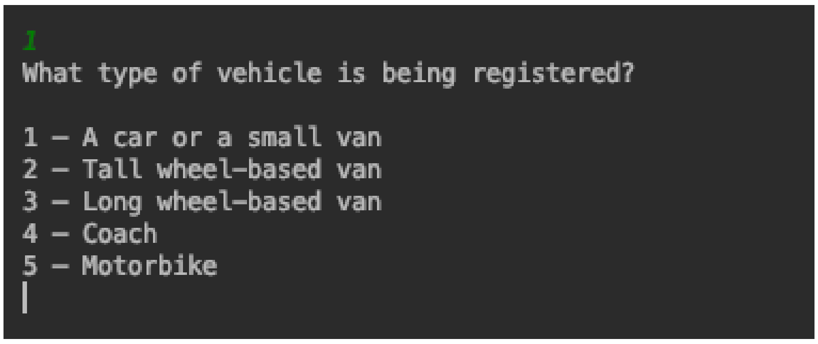
B1.3	FR13	Paying the total.	Less than required	Message: “less than total, payment was not processed, try again” – runs method again	P	
			Exact amount	Gives out a token and send to the method, which offers attendant collection.	P	
			More than required	Gives change, gives out token, asks whether the attendant is needed.	P	
B1.4	FR14	Y/N: Attendant needed?	Yes	Should remove attendant from the free pool and help customer to exit the MCP.	F	/not properly implemented/ Simply prints out message for attendant and moves to the exit.
			No	Moves to the exit	P	
			Abnormal inputs:	Message: “Input valid answer”	P	
B1.5	FR15	Exit barrier: do you wish to input the token?	Yes	If token was on hands for less than 15 minutes – vehicle released, customer gone.		
			No	Go back to the menu.	F	Asks again
C1.1	FR16	Displaying car park information	Menu option 3.	Displays information about the car park	P	
C1.2	FR17	Changing to admin mode	See A1.7	See A1.7	P	

3.2 Running program



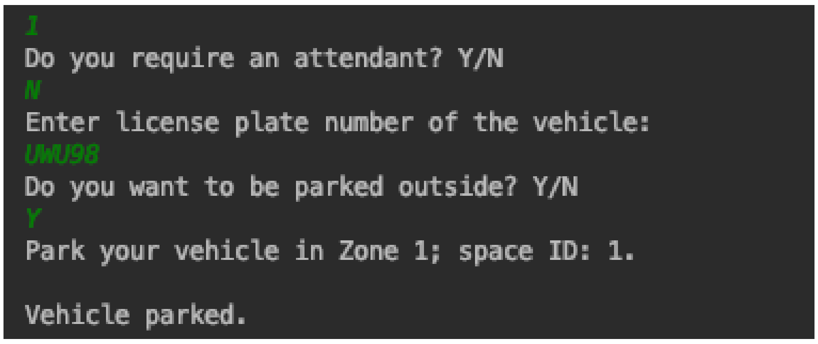
```
*** WELCOME TO MCP, MULTI-STORY PARKING PROGRAM ***

1 - Register vehicle
2 - Collect vehicle
3 - Display the car park information
4 - Change to admin mode
q - Quit

What would you like to do:
```

Figure 3: User Menu with options



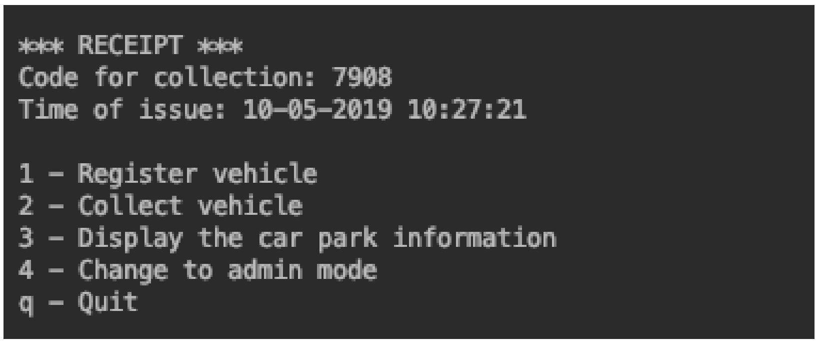
```
1
What type of vehicle is being registered?

1 - A car or a small van
2 - Tall wheel-based van
3 - Long wheel-based van
4 - Coach
5 - Motorbike
|
```

Figure 4: Registering vehicle: option 1



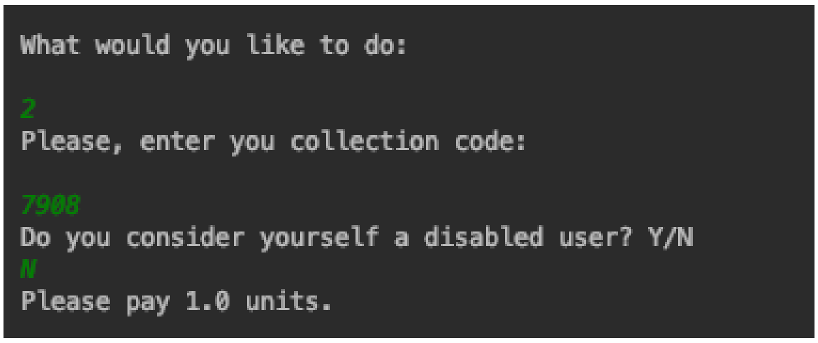
```
1
Do you require an attendant? Y/N
N
Enter license plate number of the vehicle:
UWU98
Do you want to be parked outside? Y/N
Y
Park your vehicle in Zone 1; space ID: 1.

Vehicle parked.
```

Figure 5: Answering follow-up questions and getting the space ID



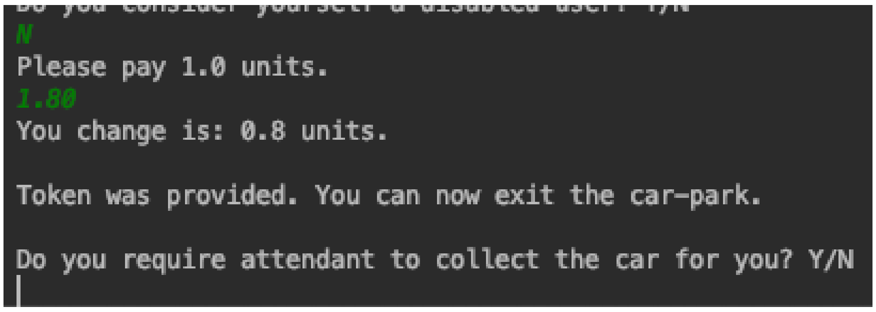
```
*** RECEIPT ***
Code for collection: 7908
Time of issue: 10-05-2019 10:27:21

1 - Register vehicle
2 - Collect vehicle
3 - Display the car park information
4 - Change to admin mode
q - Quit
```

Figure 6: The receipt, which contains collection code and time of issue



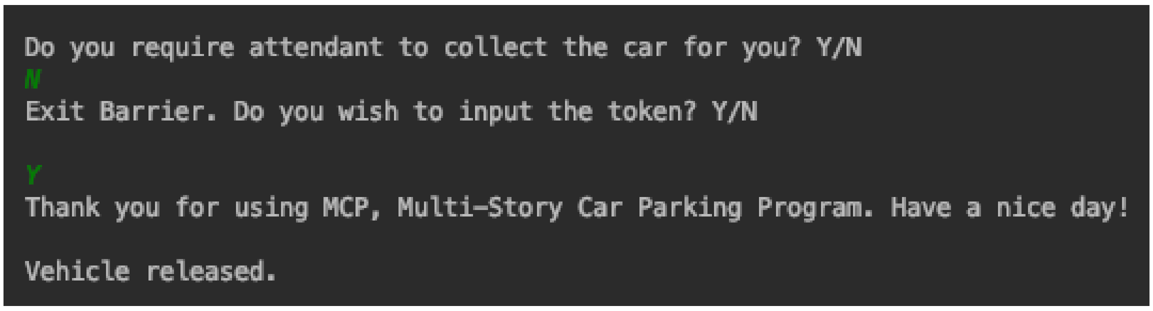
```
What would you like to do:

2
Please, enter you collection code:

7908
Do you consider yourself a disabled user? Y/N
N
Please pay 1.0 units.
```

Figure 7: Collection vehicle: option 2 and follow-up question to calculate the total



```
N
Please pay 1.0 units.
1.80
You change is: 0.8 units.

Token was provided. You can now exit the car-park.

Do you require attendant to collect the car for you? Y/N
|
```

Figure 8: Receiving change and the token, being asked whether we need an attendant



```
Do you require attendant to collect the car for you? Y/N
N
Exit Barrier. Do you wish to input the token? Y/N

Y
Thank you for using MCP, Multi-Story Car Parking Program. Have a nice day!

Vehicle released.
```

Figure 9: Another set of questions and successful release of the car

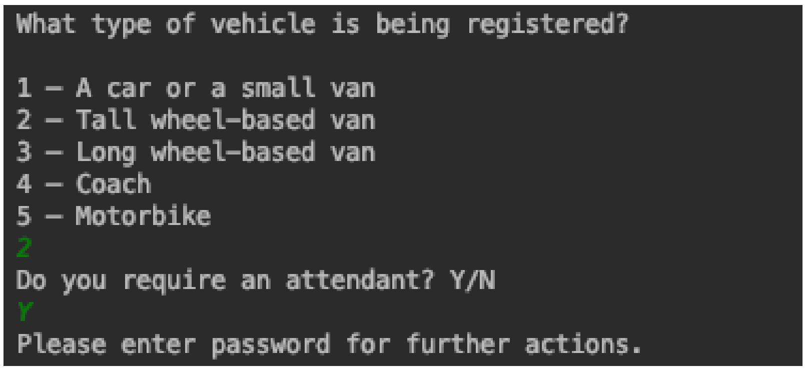

```
What type of vehicle is being registered?

1 – A car or a small van
2 – Tall wheel-based van
3 – Long wheel-based van
4 – Coach
5 – Motorbike
2
Do you require an attendant? Y/N
Y
Please enter password for further actions.
```

Figure 10: This time, we require an attendant

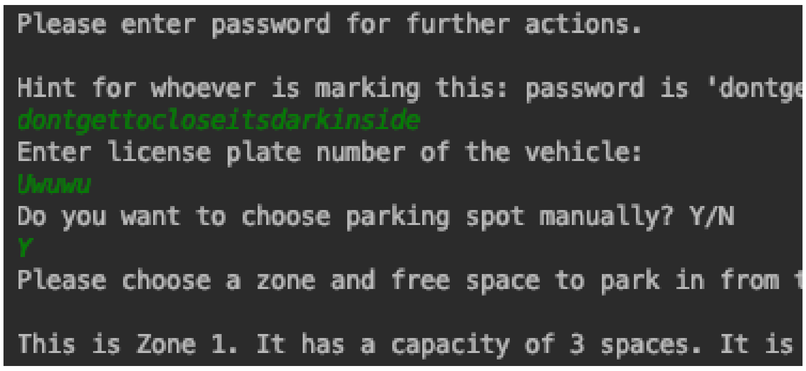

```
Please enter password for further actions.

Hint for whoever is marking this: password is 'dontge
dontgettocloseitsdarkinside
Enter license plate number of the vehicle:
Uwuwu
Do you want to choose parking spot manually? Y/N
Y
Please choose a zone and free space to park in from t

This is Zone 1. It has a capacity of 3 spaces. It is
```

Figure 11: Security Check and Manually Parking

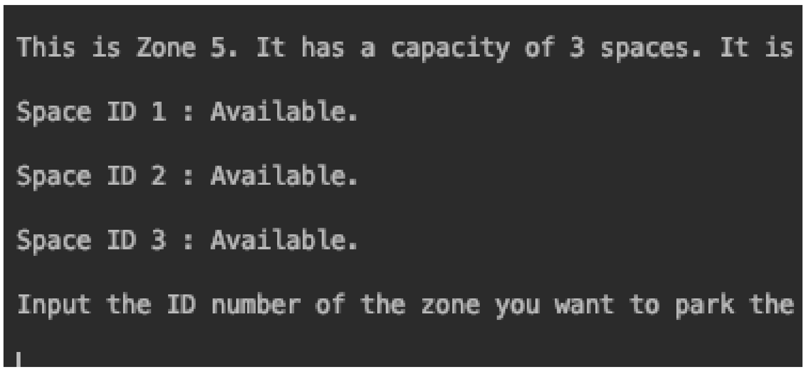

```
This is Zone 5. It has a capacity of 3 spaces. It is

Space ID 1 : Available.

Space ID 2 : Available.

Space ID 3 : Available.

Input the ID number of the zone you want to park the

|
```

Figure 12: Inputted 1 for Zone 1

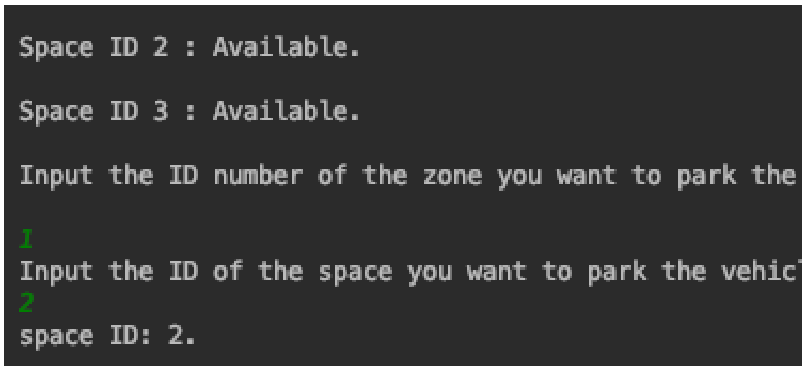
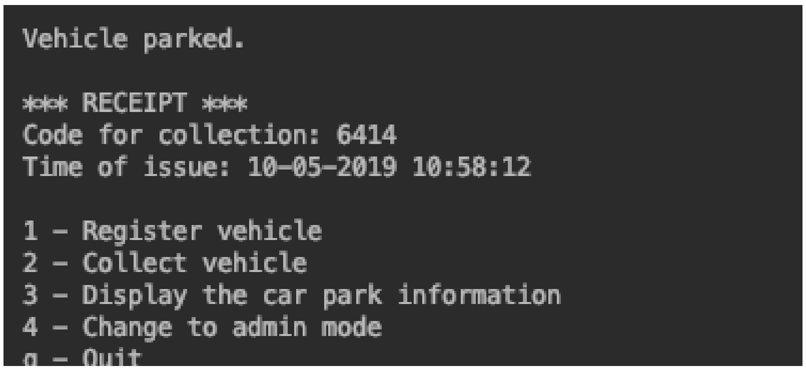
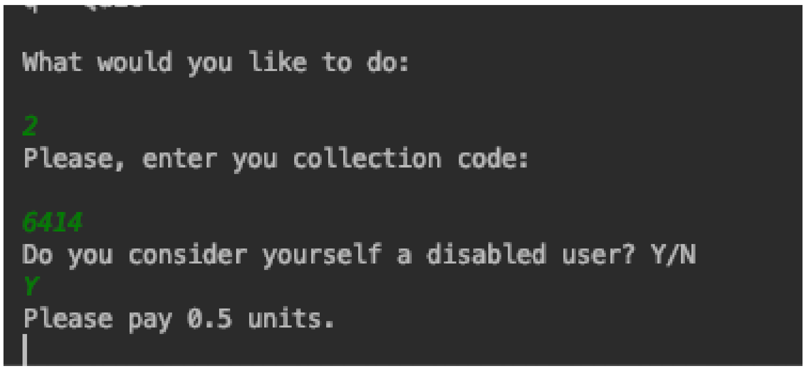

```
Space ID 2 : Available.

Space ID 3 : Available.

Input the ID number of the zone you want to park the

1
Input the ID of the space you want to park the vehic
2
space ID: 2.
```

Figure 13: And Space ID 2, which was available



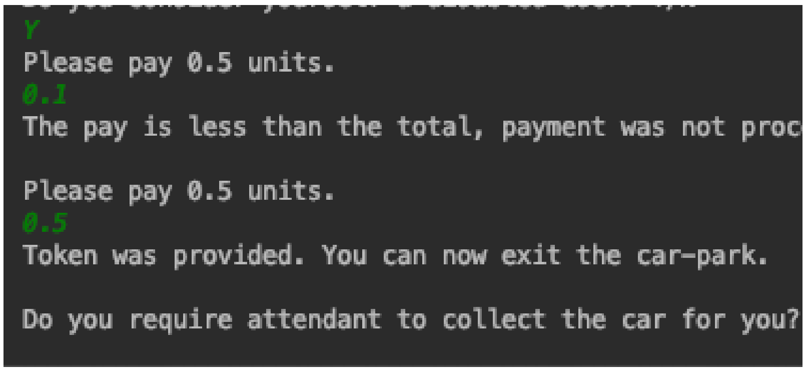
```
Vehicle parked.

*** RECEIPT ***
Code for collection: 6414
Time of issue: 10-05-2019 10:58:12

1 - Register vehicle
2 - Collect vehicle
3 - Display the car park information
4 - Change to admin mode
q - Quit
```

Figure 14: Collection code for this parking



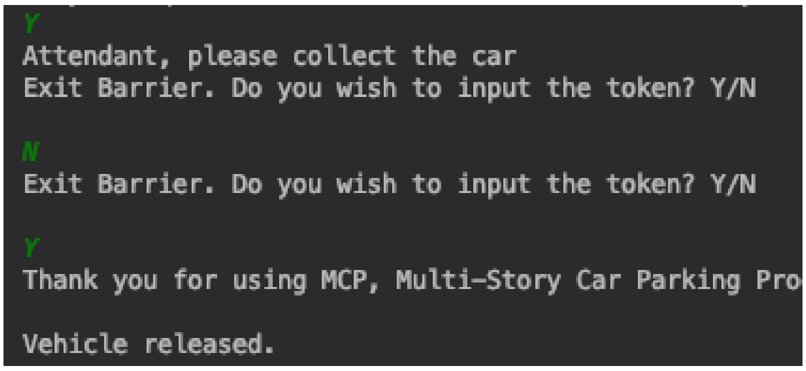
```
What would you like to do:

2
Please, enter you collection code:

6414
Do you consider yourself a disabled user? Y/N
Y
Please pay 0.5 units.
```

Figure 15: Entering collection code and choosing a disabled user



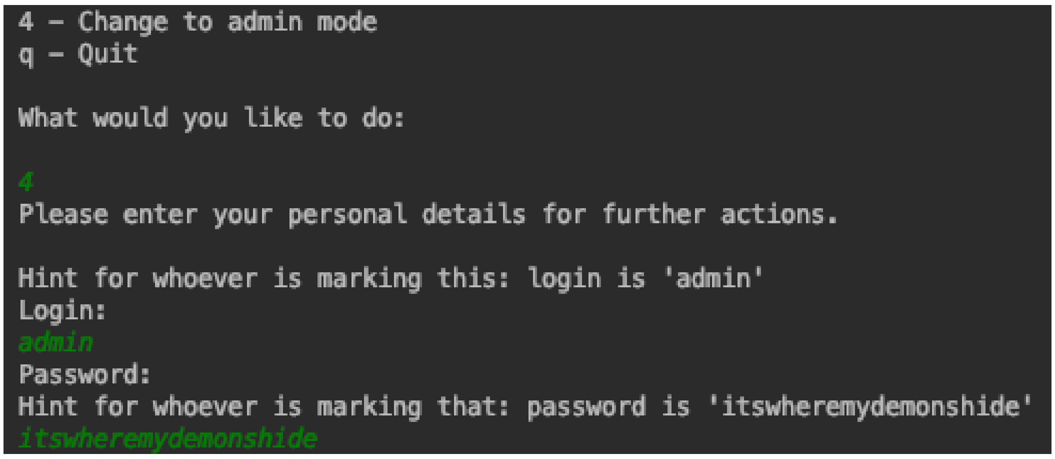
```
Y
Please pay 0.5 units.
0.1
The pay is less than the total, payment was not proc

Please pay 0.5 units.
0.5
Token was provided. You can now exit the car-park.

Do you require attendant to collect the car for you?
```

Figure 16: Checking how payment works



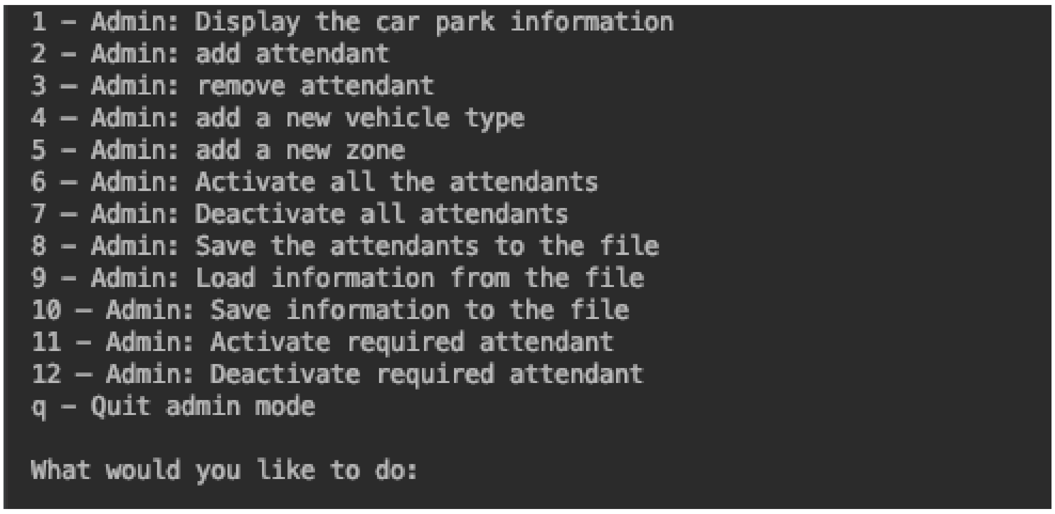
```
Y
Attendant, please collect the car
Exit Barrier. Do you wish to input the token? Y/N

N
Exit Barrier. Do you wish to input the token? Y/N

Y
Thank you for using MCP, Multi-Story Car Parking Pro

Vehicle released.
```

Figure 17: Calling attendant over and answering ”no” to the token input



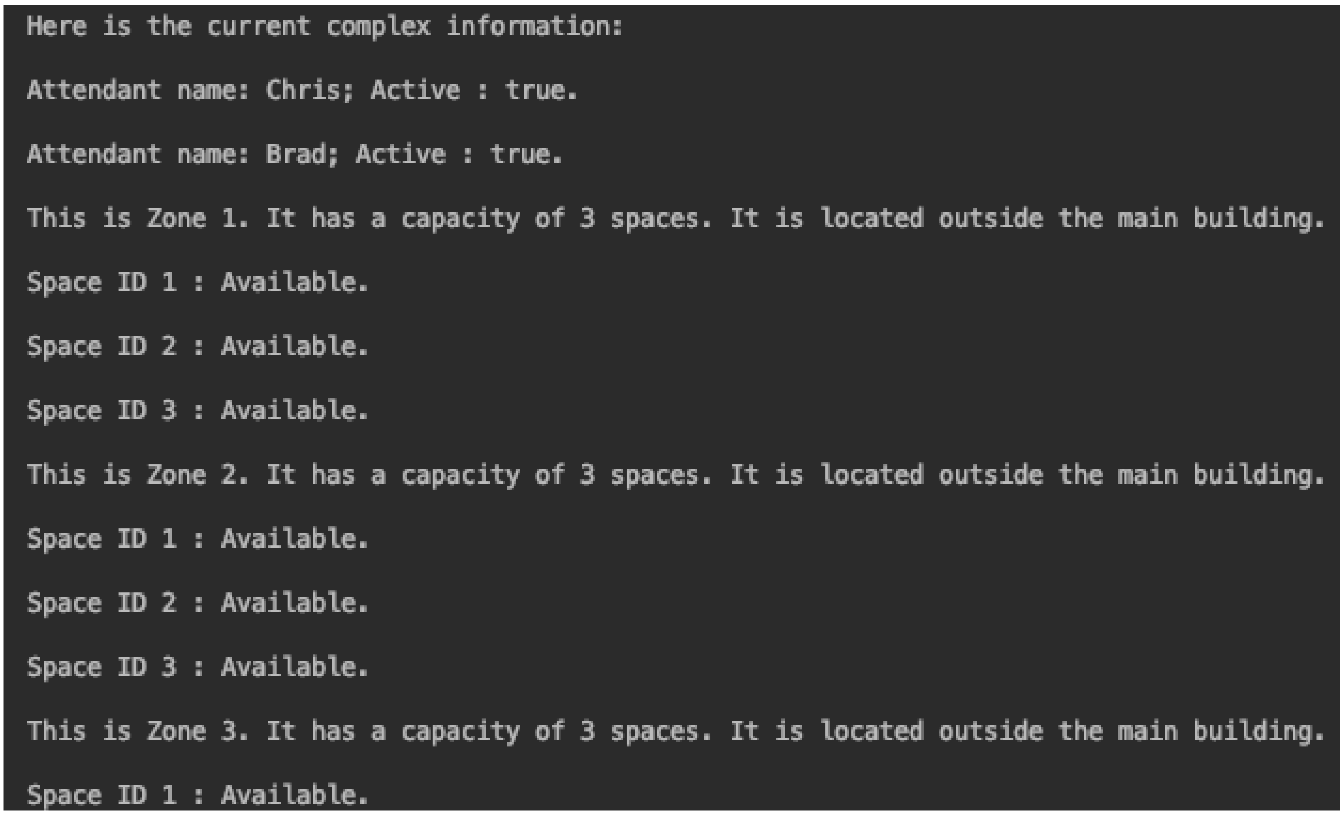
```
4 - Change to admin mode
q - Quit

What would you like to do:

4
Please enter your personal details for further actions.

Hint for whoever is marking this: login is 'admin'
Login:
admin
Password:
Hint for whoever is marking that: password is 'itswheremydemonshide'
itswheremydemonshide
```

Figure 18: Time to switch to admin mode, but first let's input the login and the password



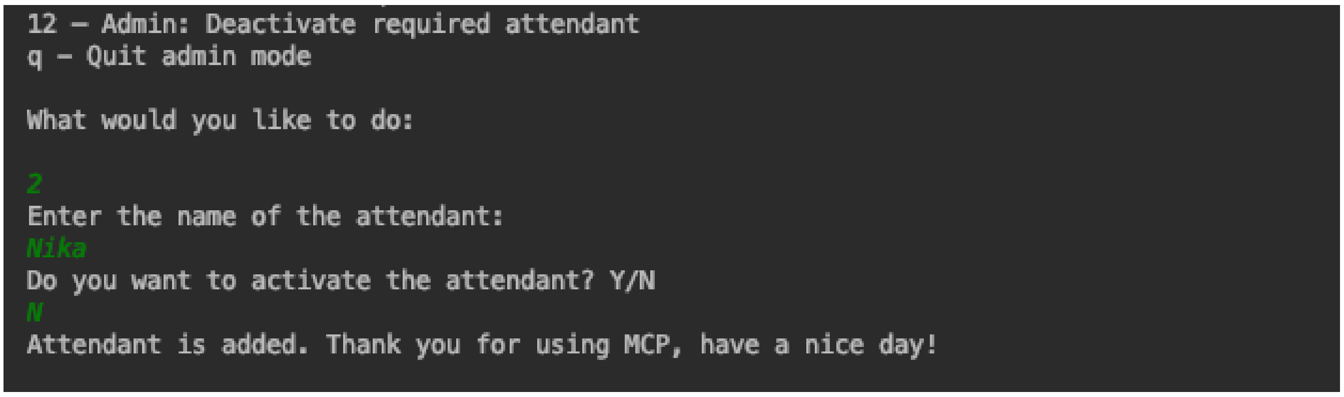
```
1 - Admin: Display the car park information
2 - Admin: add attendant
3 - Admin: remove attendant
4 - Admin: add a new vehicle type
5 - Admin: add a new zone
6 - Admin: Activate all the attendants
7 - Admin: Deactivate all attendants
8 - Admin: Save the attendants to the file
9 - Admin: Load information from the file
10 - Admin: Save information to the file
11 - Admin: Activate required attendant
12 - Admin: Deactivate required attendant
q - Quit admin mode

What would you like to do:
```

Figure 19: Admin menu



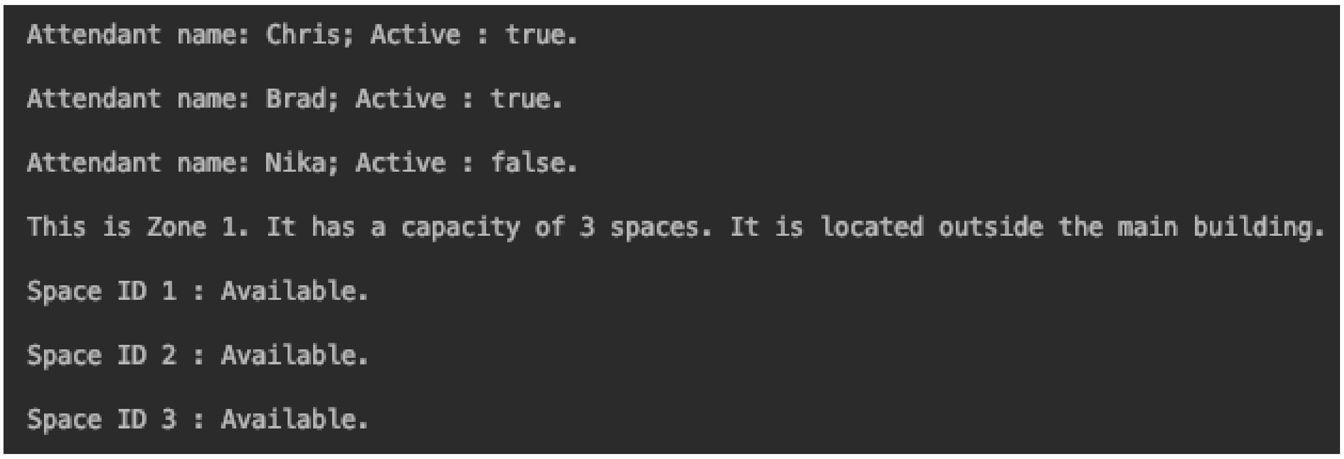
```
Here is the current complex information:

Attendant name: Chris; Active : true.

Attendant name: Brad; Active : true.

This is Zone 1. It has a capacity of 3 spaces. It is located outside the main building.

Space ID 1 : Available.

Space ID 2 : Available.

Space ID 3 : Available.

This is Zone 2. It has a capacity of 3 spaces. It is located outside the main building.

Space ID 1 : Available.

Space ID 2 : Available.

Space ID 3 : Available.

This is Zone 3. It has a capacity of 3 spaces. It is located outside the main building.

Space ID 1 : Available.
```

Figure 20: Option 1: Display Information about the Car Park



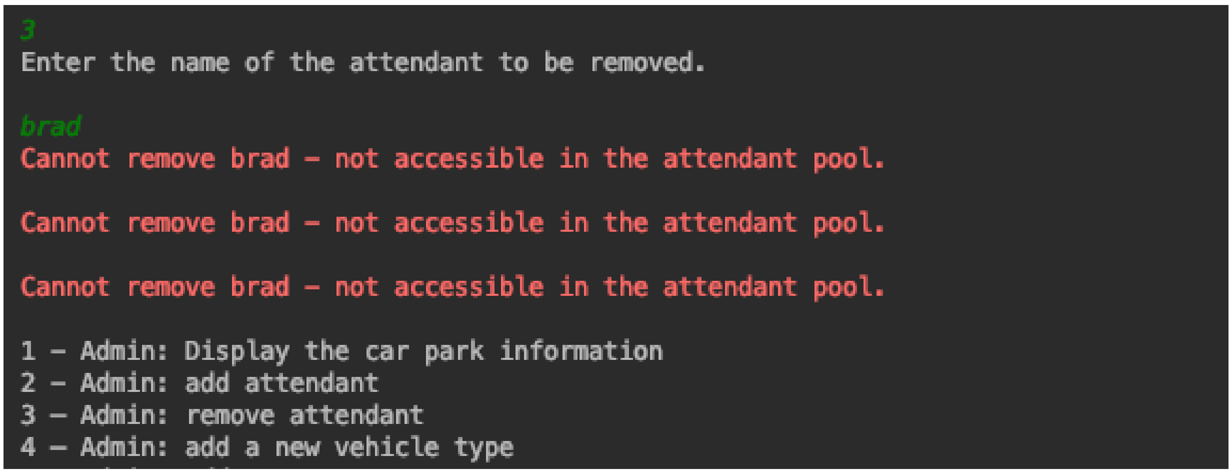
```
12 - Admin: Deactivate required attendant
q - Quit admin mode

What would you like to do:

2
Enter the name of the attendant:
Nika
Do you want to activate the attendant? Y/N
N
Attendant is added. Thank you for using MCP, have a nice day!
```

Figure 21: Option 2: Add attendant



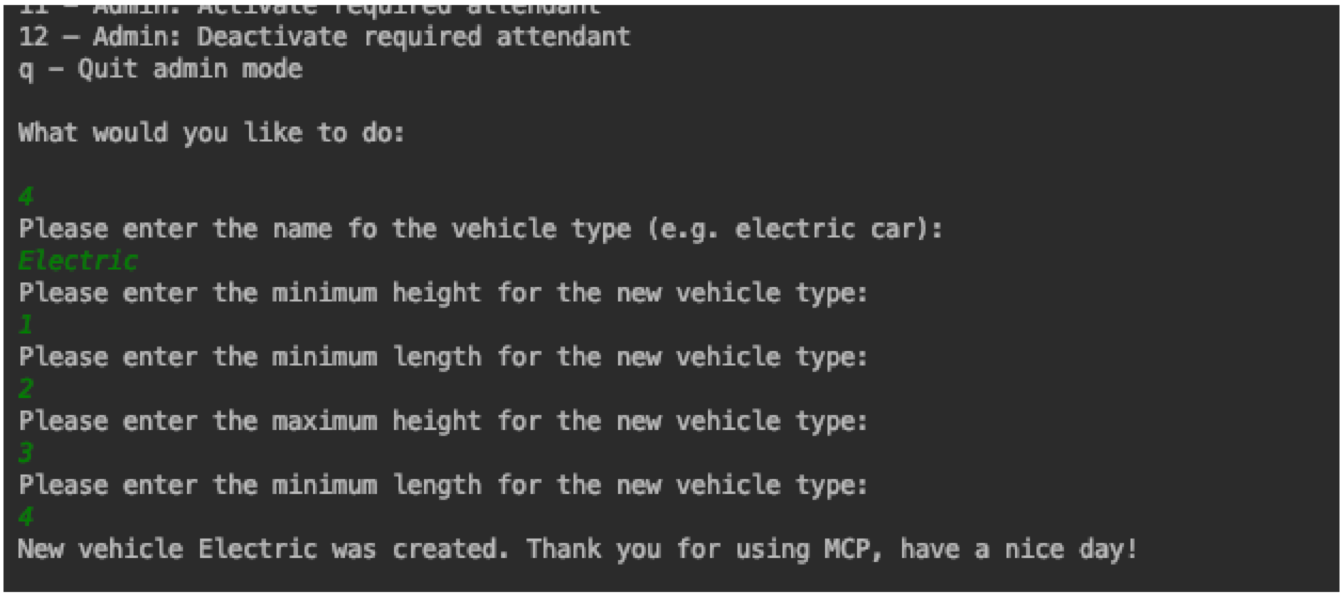
```
Attendant name: Chris; Active : true.

Attendant name: Brad; Active : true.

Attendant name: Nika; Active : false.

This is Zone 1. It has a capacity of 3 spaces. It is located outside the main building.

Space ID 1 : Available.

Space ID 2 : Available.

Space ID 3 : Available.
```

Figure 22: Attendant added



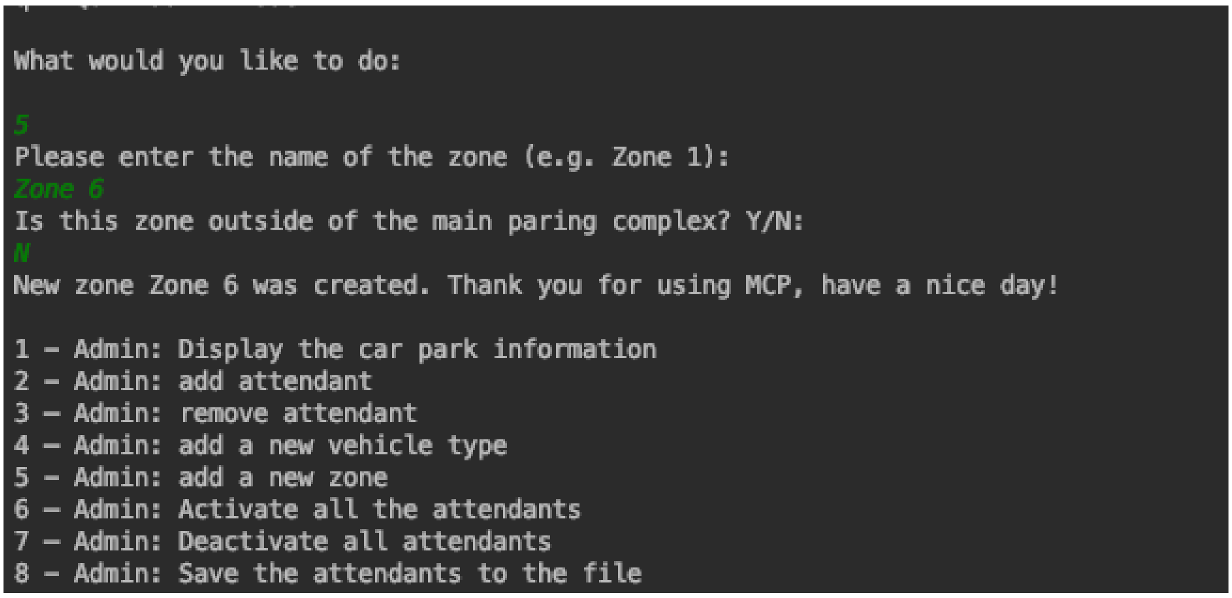
```
3
Enter the name of the attendant to be removed.

brad
Cannot remove brad - not accessible in the attendant pool.

Cannot remove brad - not accessible in the attendant pool.

Cannot remove brad - not accessible in the attendant pool.

1 - Admin: Display the car park information
2 - Admin: add attendant
3 - Admin: remove attendant
4 - Admin: add a new vehicle type
```

Figure 23: Oops! Error occured when removing the attendant.



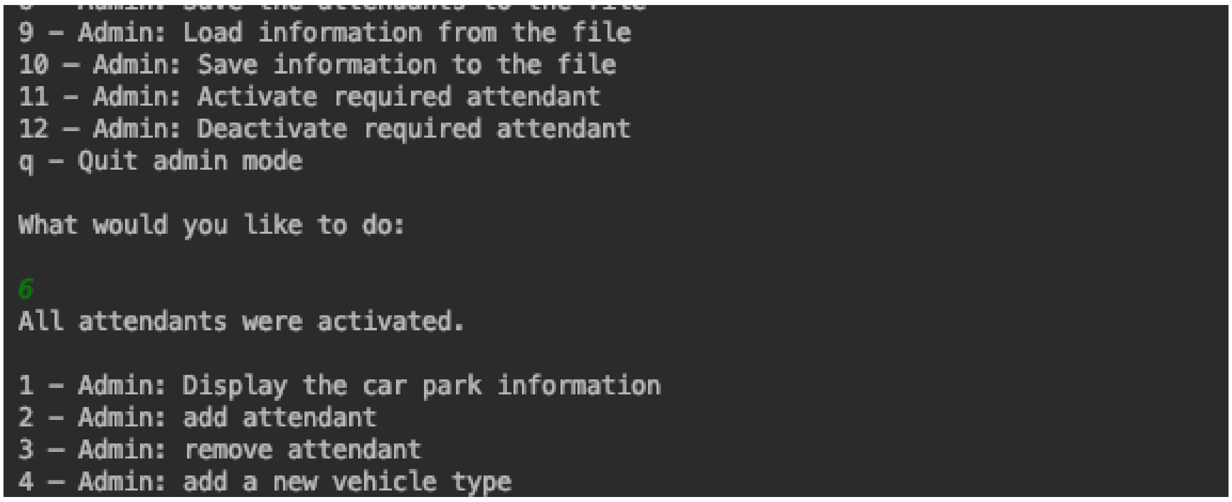
```
11 – Admin: Activate required attendant
12 – Admin: Deactivate required attendant
q – Quit admin mode

What would you like to do:

4
Please enter the name fo the vehicle type (e.g. electric car):
Electric
Please enter the minimum height for the new vehicle type:
1
Please enter the minimum length for the new vehicle type:
2
Please enter the maximum height for the new vehicle type:
3
Please enter the minimum length for the new vehicle type:
4
New vehicle Electric was created. Thank you for using MCP, have a nice day!
```

Figure 24: Creating new vehicle type



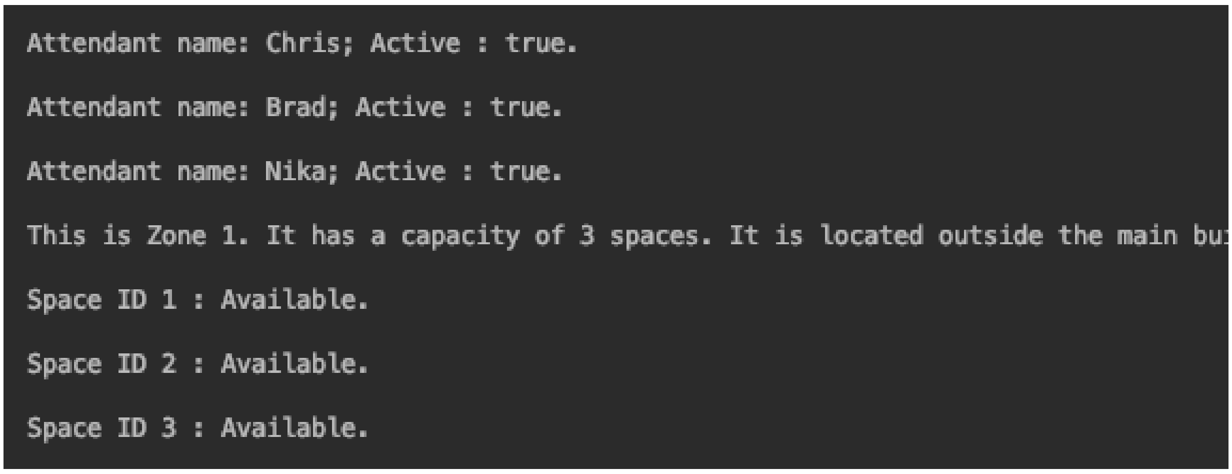
```
What would you like to do:

5
Please enter the name of the zone (e.g. Zone 1):
Zone 6
Is this zone outside of the main paring complex? Y/N:
N
New zone Zone 6 was created. Thank you for using MCP, have a nice day!

1 – Admin: Display the car park information
2 – Admin: add attendant
3 – Admin: remove attendant
4 – Admin: add a new vehicle type
5 – Admin: add a new zone
6 – Admin: Activate all the attendants
7 – Admin: Deactivate all attendants
8 – Admin: Save the attendants to the file
```

Figure 25: Creating a new zone



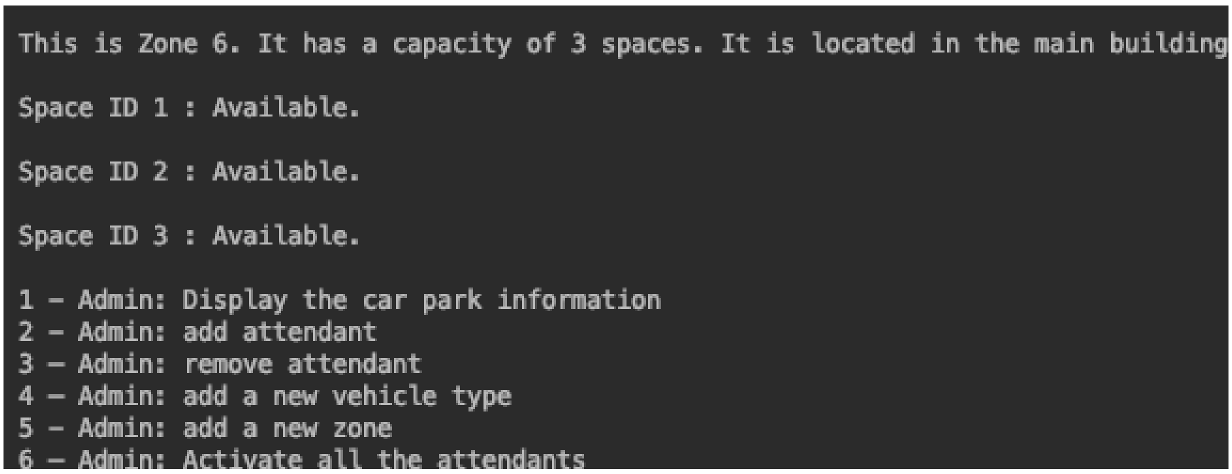
```
9 - Admin: Load information from the file
10 - Admin: Save information to the file
11 - Admin: Activate required attendant
12 - Admin: Deactivate required attendant
q - Quit admin mode

What would you like to do:

6
All attendants were activated.

1 - Admin: Display the car park information
2 - Admin: add attendant
3 - Admin: remove attendant
4 - Admin: add a new vehicle type
```

Figure 26: Activating all of the attendants.



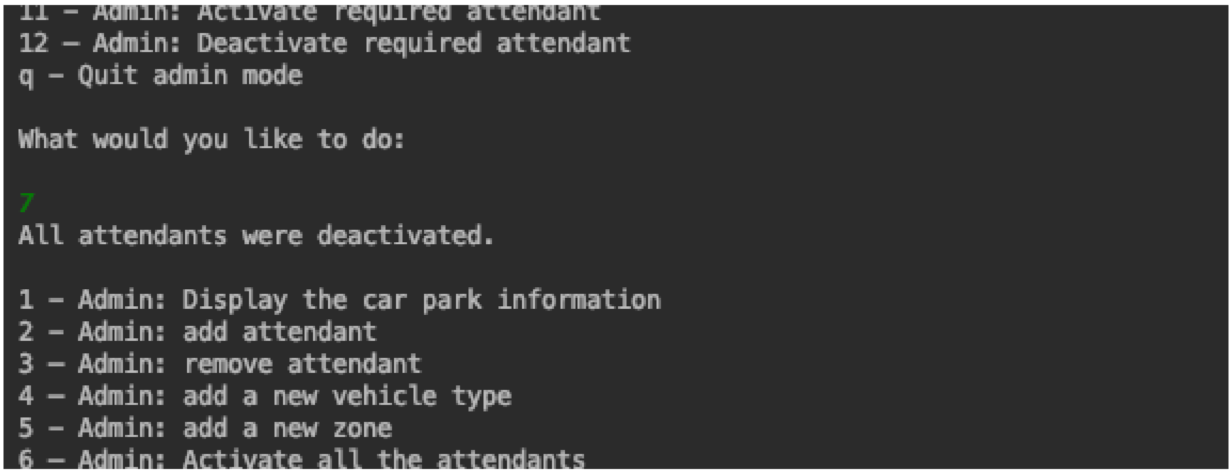
```
Attendant name: Chris; Active : true.

Attendant name: Brad; Active : true.

Attendant name: Nika; Active : true.

This is Zone 1. It has a capacity of 3 spaces. It is located outside the main bu

Space ID 1 : Available.

Space ID 2 : Available.

Space ID 3 : Available.
```

Figure 27: Proof of previous figure



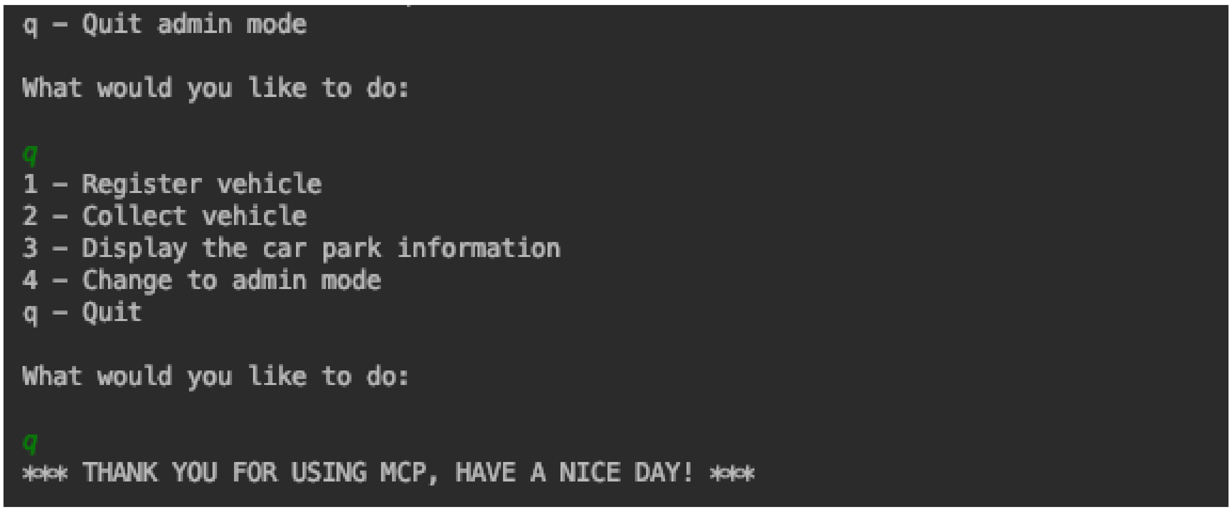
```
This is Zone 6. It has a capacity of 3 spaces. It is located in the main building

Space ID 1 : Available.

Space ID 2 : Available.

Space ID 3 : Available.

1 - Admin: Display the car park information
2 - Admin: add attendant
3 - Admin: remove attendant
4 - Admin: add a new vehicle type
5 - Admin: add a new zone
6 - Admin: Activate all the attendants
```

Figure 28: Zone we created earlier



```
11 - Admin: Activate required attendant
12 - Admin: Deactivate required attendant
q - Quit admin mode

What would you like to do:

7
All attendants were deactivated.

1 - Admin: Display the car park information
2 - Admin: add attendant
3 - Admin: remove attendant
4 - Admin: add a new vehicle type
5 - Admin: add a new zone
6 - Admin: Activate all the attendants
```

Figure 29: Deactivating all the attendants



```
q - Quit admin mode

What would you like to do:

q
1 - Register vehicle
2 - Collect vehicle
3 - Display the car park information
4 - Change to admin mode
q - Quit

What would you like to do:

q
*** THANK YOU FOR USING MCP, HAVE A NICE DAY! ***
```

Figure 30: And that is roughly it!

3.3 Test data discussion

Not much to say about that. I was just trying to check how easy it is to break the program, by inputting unexpected inputs and checking how they were handled. That means if an int is expected – try to input a char.

As far as I noticed, quite a few if not all methods that use ints as valid inputs have the try-catch implemented in order to prevent the crashing.

4 Evaluation

4.1 The process of solving the assignment

It all started with a very precise reading of the brief and underlining of any potential classes, example objects or attributes. Since the statement asks for quite a few features, the use-case diagram was not only a logical step to take but also the recommended one. Worth mentioning, that at the beginning I did not include the "admin" part of the diagram (see 1), but added it later, closer to the actual end of the coding itself.

Designing of the brief class diagram came in next, but I gave up on it surprisingly fast. Somehow, it was hard and demotivating for me to try and imagine the processes happening in the code, without actually seeing the code in front of my eyes. It seemed like I hit a block of a kind, and could not move further than the most obvious Vehicle Class.

That is why I decided to take a step back, which, I assumed, in a long run, would push me forward, and started the implementation of the Vehicle superclass and all of its subclasses. Only once I did that, I managed to come back to the left-out class diagram draft and continue it. While doing so, I was getting some inspiration from the educative website[4] and slowly started to make some progress.

Or it seemed like it, since the next two or three weeks I, again, hit the block. Did not want to try things out, which was stupid of me, because once I made myself do that – everything started to slowly unravel.

During the coding process, I was trying to use those things that we learned over the academic year. Periodically, was taking a look at the code from the mini-assignment and worksheets.

The code implementation itself turned out to be quiet interesting this time. Normally, I would just break everything into chunks and code piece by piece, but this time it felt more like adding the layers. First – make the vehicle be added to space, second – add the types each zone supports, then figure out receipts, attendants and so on.

I can think of a number of things I would improve and make more functional, if I would have given myself more time to try various concepts, rather than not willing to start in the first place.

4.2 Flair

As a good addition to the current functionality, I did add a fairly simple but quite useful security check, that makes sure that the customer does not have access to the method aimed for the attendants and the admin. Before showing any methods dedicated to the car-park employees, the program asks for the password (for attendants) or login with a password (for admins). Following that, admin can not only add a new vehicle type if required but also new zones, activate and deactivate all the attendants in the free attendant pool (see figures starting from figure 3.2).

Before starting the assignment I have actually considered using JavaFX in order to implement a map design for the occupied and non-occupied spaces, but the laptop I currently use would start to glitch and overheat, so I decided that it would be neither wise or efficient decision to try and do that.

4.3 Encountered difficulties

One of the very first problems was to actually figure out the design of the future program. I did spend a decent time on trying out different class structures on paper and thinking about the best way of incorporating zones, which at first were represented as ArrayLists, and vehicle types, which primarily were supposed to be enums. I was in between storing the receipt in the parking space class, the vehicle itself or even zones.

I was not sure what will be better, so I was pretty much afraid of starting the work on the assignment because I did not want to make a stupid mistake that would pull over many others – even though I knew that it was a mistake of its own.

If we are talking about the functional requirements, then the hardest one turned out to be the payment system. Because I was using java.time.LocalDateTime rather than currentMillis() to track the time when the vehicle was assigned to the Parking Space, I was really confused with the implementation of the payment process, which then led to problems with the implementation of the charging system.

Unfortunately, there are still some errors and incomplete bits left: removeAttendant() method in the admin menu does not do what asked (see figure 3.2), loading and saving for the entire program are not implemented – you can only save the ArrayList of attendants; rather than actually helping with collection – attendant is mentioned only in a print statement.

These and other small bugs certainly affect the quality of the program, and they would not be there if it was not for me constantly overthinking my next stop, rather then actually trying to do something.

4.4 What was learnt

I have profonded my overall knowledge of Java, since big projects like this one always challenge you to look at things under different perspectives and think about various ways of accomplishing the result you need.

AExperimented with the ways of error-checking implementation, as I did not pay enough attention to it during the mini-assignment earlier in the semester. With this project, it is not the best either, but I am trying to at least prevent the code from crushing by catching any InputMismatchException errors and specifying what I require the if statements to do in details.

Also, while doing my research on possible solutions for the problem, I have studied the use of HashMaps and LinkedLists in the context of parking lot design. Even though they have not become the center of my solution, I do use a HashMap to store the "collectionCode – licensePlate" key-value pairs. Not to mention that I went further with the ways you can store and manipulate with time in Java, learned about the java.time library and what it is capable of doing.

4.5 Recommended mark

Based on the version 1.2 of Assessment Criteria for Development, even though my work most certainly is not flawless or exceptional in any way, I think that I have managed to show a somewhat good understanding of the problem and was able to present it as such. As always, if I would have put more effort into this assignment, I would have shown an overall better performance, but at this point, I believe that I have earned myself around 65 percent of the total mark.

References

[1] *Individual Assignment: Multi-story Car-Parking Program (MCP) – CS12320/CC12320*,
Loftus, Chris,
Access link
Reviewed: 10/05/2019,
Note: Restricted Access.

[2] *Enum DayOfWeek*,
Oracle,
Acces Link
Reviewed: 10/05/2019.

[3] *How to calculate time difference between two dates using LocalDateTime in Java 8?*,
Harpeet, User,
Acces Link
Reviewed: 10/05/2019.

[4] *Design a Parking Lot*,
Design Gurus,
Acces Link
Reviewed: 10/05/2019.