

# Table of Contents

Introduction .....	2
1. Requirement Specification.....	2
2. Functional Specification .....	3
2.1 Data Flow on Context Level .....	3
2.2 Dynamic Message Passing: Control Flow in General.....	3
2.3 UML (Unified Modeling Language) .....	4
3. Product Specification .....	5
3.1 Main Methods.....	6
3.2 View Car Rental Details.....	7
3.3 Add New Car Rental Record.....	9
3.4 Borrow a Car .....	12
3.5 Modify Car Rental Details .....	14
3.6 Delete Car Rental Details .....	18
3.7 Search car rental details .....	19
3.8 List Available Cars for Rent .....	21
3.9 List All Borrowed Cars .....	22
3.10 Exit.....	23
4. Introduction to Java programming language .....	23
Conclusion.....	26
Bibliography .....	27

## Introduction

In this project, we build up a car rental system for Western Sydney University with programming language Java. It is built based on object-oriented programming (OOP) principles by inheritance, encapsulation, dynamic message passing, object composition, etc. Those include but not limited to implementing interface, encapsulating variable and methods within classes, manipulating arrays of objects, selecting the procedural code to execute in response to a method call, realizing object composition by representing “has-a” relationships within a class and “is-a-type-of” relationship between class and object.

This project focuses on system design phase in System Development Life Cycle (SDLC), illustrating the methods used to implement programming logic with OOP principles. Therefore, for now, we choose command prompt as the user interface and display the main menu and submenus. Once the system starts to run, users can access the information in the car rental system without or with minimal technical knowledge on the backend logic and deployment. The instructions to guide users to use this system has been included in this system along with any choice the users make while using this system. After testing phase, in the future, we shall move on to the next phase of SDLC, the implementation phase, and integrate a graphical user interface into this project.

For the data storage, as this is a scalable project and the focus for now is to design a set of feasible programming logics, the data amount is only 10 car rental records. Ideally, if this program can run 10 records, it can run 100 or 10000 records as well. As a result, we choose text file as the database to store and retrieve data.

Regarding the logical and physical design for car records, firstly, we create an array of car objects in CarReadWrite class, and retrieve car records from the input text file (the primary database) with Scanner library. For each car record, the system automatically scans every attribute value to create a new car object. By doing this repeatedly, all the car records in the CarInput text files are transformed into car objects stored in the car list. During the program execution, if there is any need to store the modified car records to the user, the writeOutput method in the CarReadWrite will be called. Administrators can find the updated data in the output text file.

## 1. Requirement Specification

The objective of this project is to design and implement a Java OOP program that runs in command prompt and loads existing car rental records from our database. If any modification takes place, it stores the update data in the output file.

In addition, here are the required functions for this project,

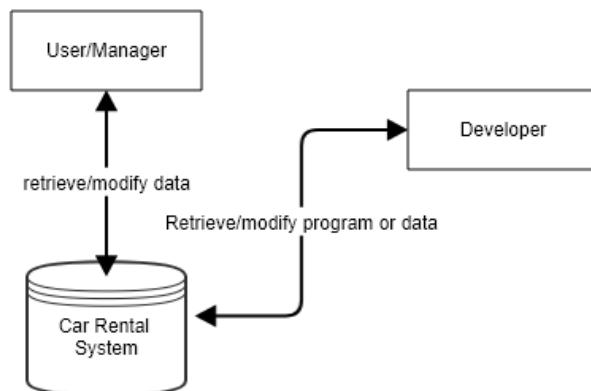
1. View Car rental records
2. Add new car rental record

3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental record
7. List of available cars for rent
8. List all borrowed cars
9. Exit

## 2. Functional Specification

### 2.1 Data Flow on Context Level

For data flow between the system and the environment, on the context level, there are two types of users to the car rental system. One is the manager or other users who can retrieve and modify data. The other is developer who can not only retrieve and modify data, but also modify the metadata and the program itself.



*Figure 1 Data Flow on Context Level*

### 2.2 Dynamic Message Passing: Control Flow in General

According to the requirement specification, there will be nine functionalities within the car rental system. The control flow of this system starts from the main method, which resides in the `ManageCar` class. In the main method, the system displays main menu to the user and allow users to choose a submenu/function to run. After the execution of every submenu, the system will direct the control flow back to the main menu automatically and asks the user to choose a submenu again.

The figure below shows all the functionalities except the exit function in the main menu.

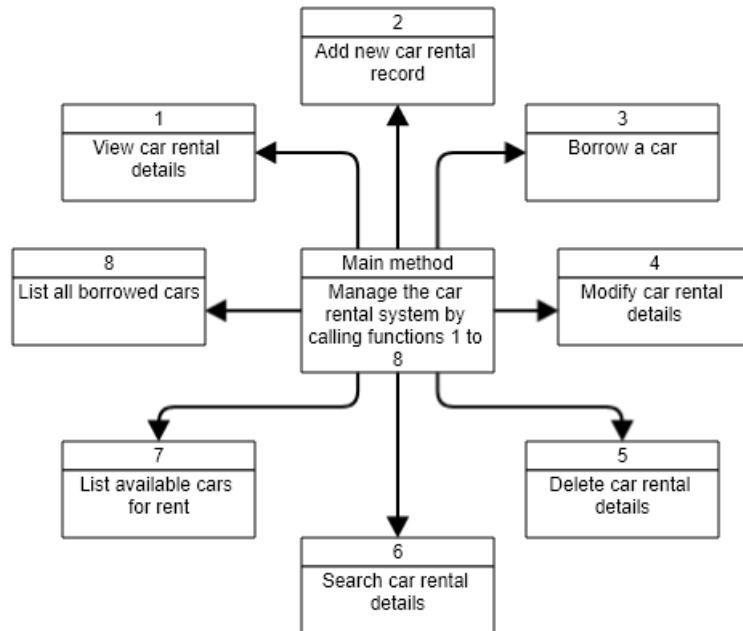


Figure 2 Control Flow in General

## 2.3 UML (Unified Modeling Language)

For each class in the car rental system, the model of the structure, attributes, and behavior are diagrammed as Figure 3.

### Interface and its implementation

Vehicle interface defines the basic setter, getter, and `toString` methods for all vehicle classes. As a type of vehicle, Car class implements the vehicle interface and define the methods inherited.

### Data Flow Control

CarReadWrite class controls most of data flow within this system for both data input and output purposes. It supplies data and for other classes to execute their functions properly. In addition, there is a copy of all car rental data stored in the Car class and displayed by the View class. This is an experiment trying to show the other way to declare, initialize, store, and display objects within the same class. This is the only case to manipulate car rental records data without using the database (CarInput and output text files).

### Main Method in ManageCar class

Inside the ManageCar class, the main method can call View class, CarReadWrite class, Borrow class, Modify class, Search class. Those classes contain diverse functions that can be called according to user's choice.

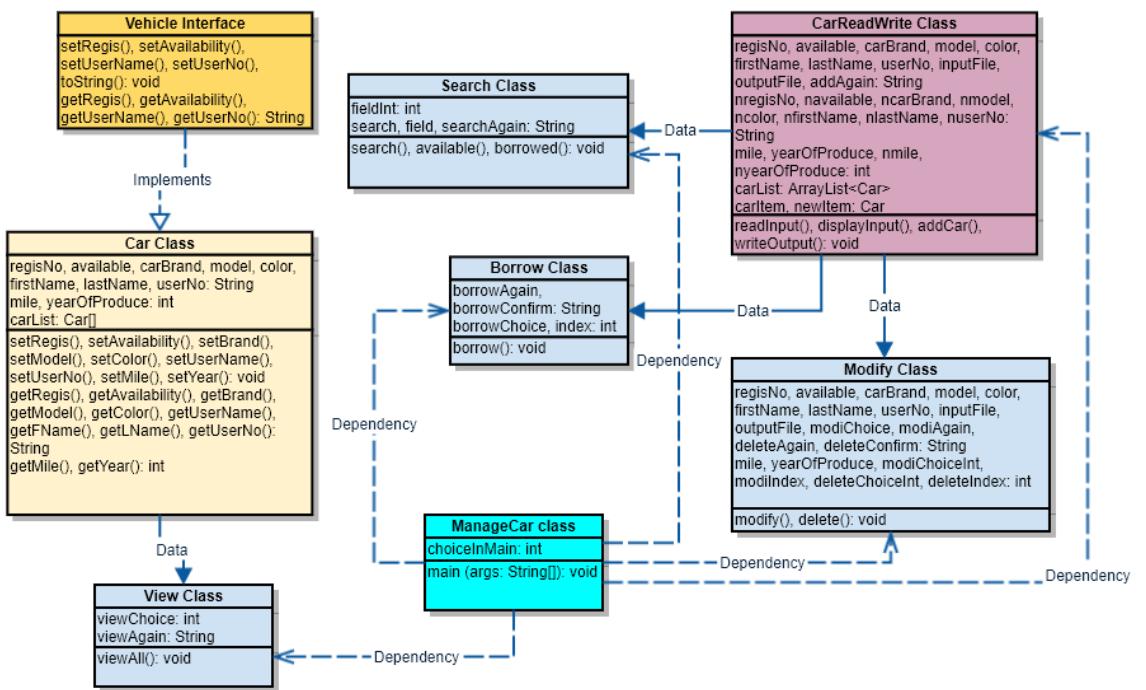


Figure 3 UML of the Car Rental System

### 3. Product Specification

Regarding the design for car records, firstly, we create an array of car objects in **CarReadWrite** class, and retrieve car records from the input text file (the primary database) with Scanner library. For each car record, the system automatically scans every attribute value to create a new car object. By doing this repeatedly, all the car records in the input text files are transformed into car objects stored in the car list.

```

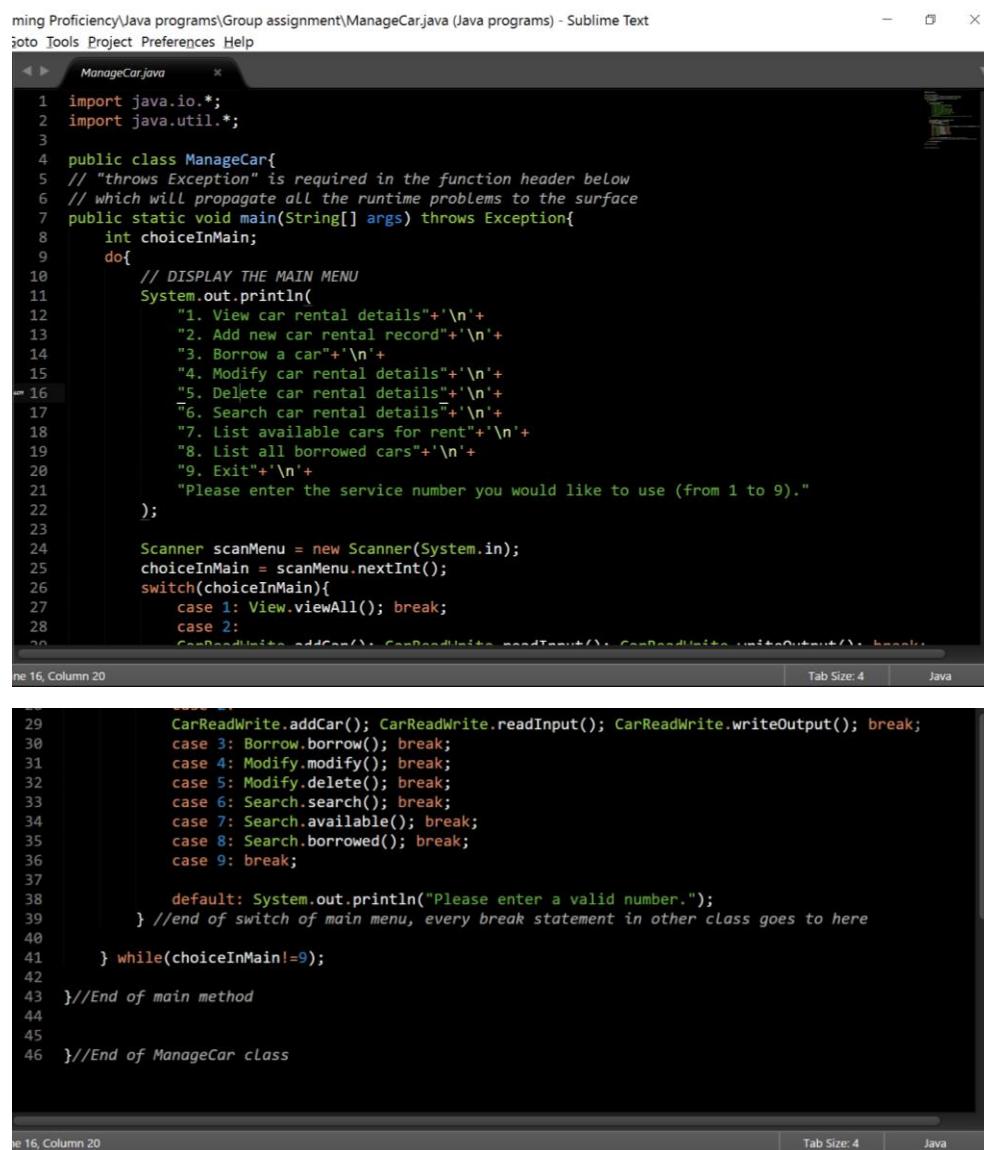
15
16     Scanner scanFile = new Scanner(new File(inputFile));
17
18     // Scan info from inputFile, add them into the carList
19     while (scanFile.hasNextLine()){
20         regisNo = scanFile.nextInt();
21         available = scanFile.next();
22         carBrand = scanFile.next();
23         model = scanFile.next();
24         color = scanFile.next();
25         while (scanFile.hasNextInt()){
26             mile = scanFile.nextInt();
27             yearOfProduce = scanFile.nextInt();
28         }
29         if (scanFile.hasNext()){
30             firstName = scanFile.next();
31         }
32         if (scanFile.hasNext()){
33             lastName = scanFile.next();
34             userNo = scanFile.next();
35         }
36         // add scanned info into the carList
37         Car carItem = new Car(regisNo,available,carBrand,model,color,mile,
38             yearOfProduce,firstName,lastName,userNo);
39         carList.add(carItem);
40     }
41     scanFile.close();
42 } //End of readInput method
43

```

### 3.1 Main Methods

Main methods resides the driving class of this program, ManageCar class, which is written with do-while loop and switch statement. The do-while loop makes sure every time after the user uses one of the services (option ranges from 1 to 8), the system will direct the user back to the main menu to choose a service to use again.

```
1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).
|
```



The screenshot shows the Java code for the ManageCar class in Sublime Text. The code includes a main menu loop with a switch statement for options 1 through 8, and a break statement for option 9 to exit the loop. The code is annotated with comments explaining the purpose of each section.

```
1 import java.io.*;
2 import java.util.*;
3
4 public class ManageCar{
5 // "throws Exception" is required in the function header below
6 // which will propagate all the runtime problems to the surface
7 public static void main(String[] args) throws Exception{
8     int choiceInMain;
9     do{
10         // DISPLAY THE MAIN MENU
11         System.out.println(
12             "1. View car rental details\n"+
13             "2. Add new car rental record\n"+
14             "3. Borrow a car\n"+
15             "4. Modify car rental details\n"+
16             "5. Delete car rental details\n"+
17             "6. Search car rental details\n"+
18             "7. List available cars for rent\n"+
19             "8. List all borrowed cars\n"+
20             "9. Exit\n"+
21             "Please enter the service number you would like to use (from 1 to 9)."
22     );
23
24     Scanner scanMenu = new Scanner(System.in);
25     choiceInMain = scanMenu.nextInt();
26     switch(choiceInMain){
27         case 1: View.viewAll(); break;
28         case 2: CarReadWrite.addCar(); CarReadWrite.readInput(); CarReadWrite.writeOutput(); break;
29         case 3: Borrow.borrow(); break;
30         case 4: Modify.modify(); break;
31         case 5: Modify.delete(); break;
32         case 6: Search.search(); break;
33         case 7: Search.available(); break;
34         case 8: Search.borrowed(); break;
35         case 9: break;
36
37         default: System.out.println("Please enter a valid number.");
38     } //end of switch of main menu, every break statement in other class goes to here
39 } //end of main method
40
41 } //End of ManageCar class
```

### 3.2 View Car Rental Details

Once the user starts to run this car rental system, firstly the system displays the main menu and asks user's choice by number. If user enter 1, it calls View class and shows a list of cars and asks the user to choose a car to see details.

For example, if the user enter 2, the detailed car rental record of Car No. 2 will be display. Then the system will ask if user wants to see another car records.

If the user enter "yes" (both Yes or yes will do, the system can take both), then the system will ask the user to enter a car number to check the car details; if the user enter "no", the system will take the user go back to the main menu.

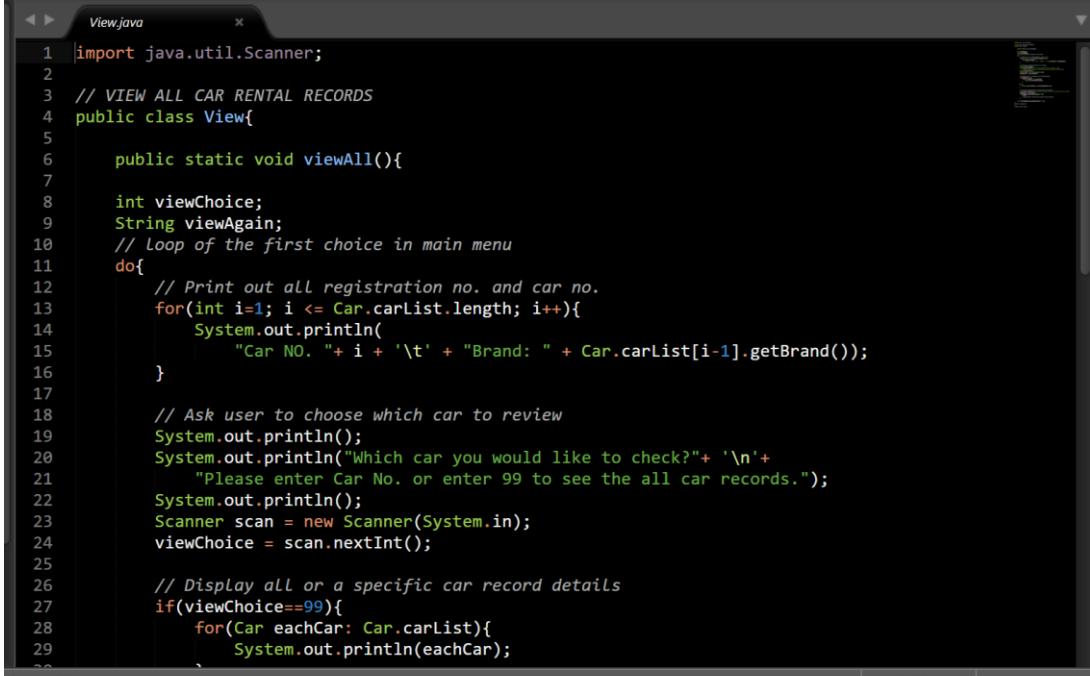
```
1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).
1
Car NO. 1   CarModel Benz01
Car NO. 2   CarModel Nissan01
Car NO. 3   CarModel Toyota01
Car NO. 4   CarModel Audi01
Car NO. 5   CarModel Ferrari01
Car NO. 6   CarModel Mazda01
Car NO. 7   CarModel Ford01
Car NO. 8   CarModel Volvo01
Car NO. 9   CarModel Peugeot01
Car NO. 10  CarModel Subaru01

Which car you would like to check?
Please enter Car No. or enter 99 to see the all car records.

2
Car registration number: C2
Availability: Yes
Brand: Nissan
Model: Nissan01
Color: black
Mileage: 12000
Year of Produce: 2010
User Name: null null
User Number: null

Would you like to check other car records? (yes or no)
no
1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).|
```

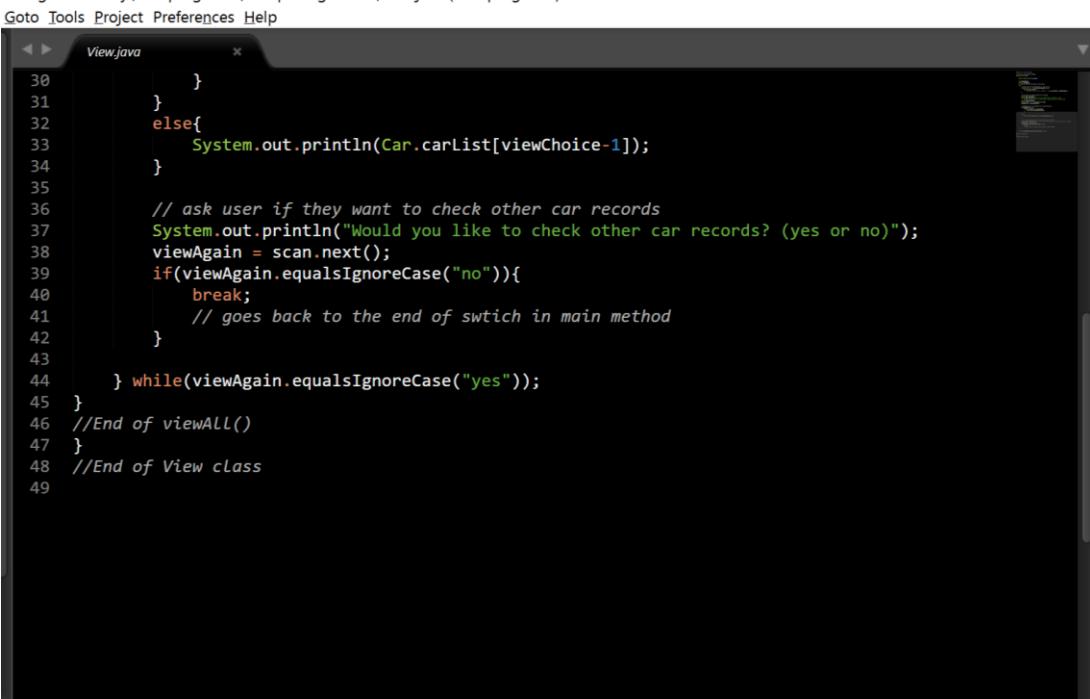
Source code of viewAll method in View class is shown as below. In viewAll method, the system starts with a do-while loop, then enter a for loop to display all registration no. and car no., ask user to choose which car to review, then display all or a specific car record details according to the user's choice. In the end, it asks the user whether he/she wants to check other car records. If the answer is yes, then the control flow will start the do-while loop again. If the answer is no, the system will go back to the main menu.



```

View.java * 
1 import java.util.Scanner;
2
3 // VIEW ALL CAR RENTAL RECORDS
4 public class View{
5
6     public static void viewAll(){
7
8         int viewChoice;
9         String viewAgain;
10        // Loop of the first choice in main menu
11        do{
12            // Print out all registration no. and car no.
13            for(int i=1; i <= Car.carList.length; i++){
14                System.out.println(
15                    "Car NO. " + i + '\t' + "Brand: " + Car.carList[i-1].getBrand());
16            }
17
18            // Ask user to choose which car to review
19            System.out.println();
20            System.out.println("Which car you would like to check?" + '\n' +
21                "Please enter Car No. or enter 99 to see the all car records.");
22            System.out.println();
23            Scanner scan = new Scanner(System.in);
24            viewChoice = scan.nextInt();
25
26            // Display all or a specific car record details
27            if(viewChoice==99){
28                for(Car eachCar: Car.carList){
29                    System.out.println(eachCar);
30                }
31            } else{
32                System.out.println(Car.carList[viewChoice-1]);
33            }
34
35            // ask user if they want to check other car records
36            System.out.println("Would you like to check other car records? (yes or no)");
37            viewAgain = scan.next();
38            if(viewAgain.equalsIgnoreCase("no")){
39                break;
40                // goes back to the end of switch in main method
41            }
42        }
43
44    } while(viewAgain.equalsIgnoreCase("yes"));
45 }
46 //End of viewAll()
47 }
48 //End of View class
49

```



```

Line 1, Column 1 | Tab Size: 4 | Java

```

### 3.3 Add New Car Rental Record

If the user chooses to add a new car, the system will ask the user to enter information about the new car including registration number, availability, car brand, car model, color, mileage, year of production, user name, and user phone number.

```
2
Please enter new car registration no.:
Or enter "exit" to exit.
C11
Please enter availability (yes or no).
no
Please enter car brand.
Ford
Please enter car model.
FordHatchBack
Please enter car color.
Red
Please enter mileage (number).
25000
Please enter the year of produce.
2011
Please enter the first name of the user.
null
Please enter the last name of the user.
null
Please enter the phone number of the user.
null
You have added a new car record:

Car registration number: C11
Availability: no
Brand: Ford
Model: FordHatchBack
Color: Red
Mileage: 25000
Year of Produce: 2011
User Name: null null
User Number: null

Would you like to add more?
```

After adding the requested fields, the system will confirm the correctness of this new record. At this moment, the administrator or developer can find the added new car record inside the output text file.

About the backend technology, in the CarReadWrite class, firstly, to execute this function, the main method will call addCar method to ask the user enter related values. Secondly, the system calls the readInput method to retrieve existing car records from database (CarInput text file). Thirdly, the system will call writeOutput method to write the existing car records and newly added car records into the output

text file all together. Source code of addCar method, readInput method, writeOutput method are shown as below.

### addCar method:

```

mming Proficiency\Java programs\Group assignment\CarReadWrite.java • (Java programs) - Sublime Text
Goto Tools Project Preferences Help
CarReadWrite.java ManageCar.java
53 // ASK USERS TO ADD NEW CAR RECORDS
54 public static void addCar() throws Exception{
55     Scanner newScan = new Scanner(System.in);
56     String addAgain;
57     do{
58         String nregisNo="", navailable="", ncarBrand="", nmodel="", ncolor="", nfirstName="";
59         int nmile=0, nyearOfProduce=0;
60         System.out.println("Please enter new car registration no.: \n"+
61             "Or enter \"exit\" to exit.");
62         nregisNo = newScan.nextLine();
63         // If statement HAS TO put before next to the first newScan!
64         if (nregisNo.equalsIgnoreCase("exit")){
65             break;
66
67         System.out.println("Please enter availability (yes or no).");
68         navailable = newScan.nextLine();
69         System.out.println("Please enter car brand.");
70         ncarBrand = newScan.nextLine();
71         System.out.println("Please enter car model.");
72         nmodel = newScan.nextLine();
73         System.out.println("Please enter car color.");
74         ncolor = newScan.nextLine();
75         System.out.println("Please enter mileage (number).");
76         nmile = newScan.nextInt();
77         System.out.println("Please enter the year of produce.");
78         nyearOfProduce = newScan.nextInt();
79         System.out.println("Please enter the first name of the user.");
80         nfirstName = newScan.nextLine();
81         System.out.println("Please enter the last name of the user.");
82         nlastName = newScan.nextLine();
83         System.out.println("Please enter the phone number of the user.");
84         nuserNo = newScan.nextInt();
85
86         // Add the new car item into carList
87         Car newItem = new Car(nregisNo,navailable,ncarBrand,nmodel,ncolor,nmile,
88             nyearOfProduce,nfirstName,nlastName,nuserNo);
89         carList.add(newItem);
90
91         //Check for entering next loop
92         System.out.println("You have added a new car record: "+ "\n\n" + newItem + "\n");
93         System.out.println("Would you like to add more?");
94         addAgain = newScan.nextLine();
95         if (addAgain.equalsIgnoreCase("no")){
96             break;
97         }
98
99     } while(addAgain.equalsIgnoreCase("yes"));
100
101     newScan.close();
102 } // End of addCar method
103
Line 54, Column 47
Tab Size: 4
Java

```

```

ning Proficiency\Java programs\Group assignment\CarReadWrite.java • (Java programs) - Sublime Text
Goto Tools Project Preferences Help
CarReadWrite.java ManageCar.java
75     System.out.println("Please enter mileage (number).");
76     nmile = newScan.nextInt();
77     System.out.println("Please enter the year of produce.");
78     nyearOfProduce = newScan.nextInt();
79     System.out.println("Please enter the first name of the user.");
80     nfirstName = newScan.nextLine();
81     System.out.println("Please enter the last name of the user.");
82     nlastName = newScan.nextLine();
83     System.out.println("Please enter the phone number of the user.");
84     nuserNo = newScan.nextInt();
85
86     // Add the new car item into carList
87     Car newItem = new Car(nregisNo,navailable,ncarBrand,nmodel,ncolor,nmile,
88         nyearOfProduce,nfirstName,nlastName,nuserNo);
89     carList.add(newItem);
90
91     //Check for entering next loop
92     System.out.println("You have added a new car record: "+ "\n\n" + newItem + "\n");
93     System.out.println("Would you like to add more?");
94     addAgain = newScan.nextLine();
95     if (addAgain.equalsIgnoreCase("no")){
96         break;
97     }
98
99 } while(addAgain.equalsIgnoreCase("yes"));
100
101 newScan.close();
102 } // End of addCar method
103
Line 54, Column 47
Tab Size: 4
Java

```

### readInput method:

```
ning Proficiency\Java programs\Group assignment\CarReadWrite.java • (Java programs) - Sublime Text
File Tools Project Preferences Help
CarReadWrite.java ManageCar.java
1 import java.util.*;
2 import java.io.*;
3
4 public class CarReadWrite {
5
6     static private String regisNo="", available="", carBrand="", model="", color"",
7         firstName="", lastName="", userNo="";
8     static private int mile=0, yearOfProduce=0;
9     static private String inputFile="carInput.txt", outputFile="output.txt";
10    static ArrayList<Car> carList = new ArrayList<Car>();
11
12    // The existing car rental records
13    // Must put "throws Exception" in method header
14    public static void readInput() throws Exception{
15
16        Scanner scanFile = new Scanner(new File(inputFile));
17
18        // Scan info from inputFile, add them into the carlist
19        while (scanFile.hasNextLine()){
20            regisNo = scanFile.next();
21            available = scanFile.next();
22            carBrand = scanFile.next();
23            model = scanFile.next();
24            color = scanFile.next();
25            while (scanFile.hasNextInt()){
26                mile = scanFile.nextInt();
27                yearOfProduce = scanFile.nextInt();
28            }
29        }
30    }
31
32    if (scanFile.hasNext()){
33        firstName = scanFile.next();
34    }
35    if (scanFile.hasNext()){
36        lastName = scanFile.next();
37        userNo = scanFile.next();
38    }
39    // add scanned info into the carlist
40    Car carItem = new Car(regisNo,available,carBrand,model,color,mile,
41        yearOfProduce,firstName,lastName,userNo);
42    carList.add(carItem);
43
44    scanFile.close();
45 } //End of readInput method
46
47 // Display the content in the array carList
48 public static void displayInput(){
49     int idx = 0;
50     while(idx < carList.size()){
51         System.out.println(carList.get(idx));
52         idx++;
53     }
54 } //end of displayInput method
55
56 // ASK USERS TO ADD NEW CAR RECORDS
57 public static void addCar() throws Exception{
58     Scanner newScan = new Scanner(System.in);
59     String addAgain;
60
61     do{
62
63         System.out.print("Enter Regis No: ");
64         regisNo = newScan.nextLine();
65
66         System.out.print("Enter Available: ");
67         available = newScan.nextLine();
68
69         System.out.print("Enter Car Brand: ");
70         carBrand = newScan.nextLine();
71
72         System.out.print("Enter Model: ");
73         model = newScan.nextLine();
74
75         System.out.print("Enter Color: ");
76         color = newScan.nextLine();
77
78         System.out.print("Enter Mileage: ");
79         mile = newScan.nextInt();
80
81         System.out.print("Enter Year Of Produce: ");
82         yearOfProduce = newScan.nextInt();
83
84         Car carItem = new Car(regisNo,available,carBrand,model,color,mile,
85             yearOfProduce,firstName,lastName,userNo);
86
87         carList.add(carItem);
88
89         System.out.print("Do you want to add again? (y/n): ");
90         addAgain = newScan.nextLine();
91
92         if (addAgain.equalsIgnoreCase("n")){
93             break;
94         }
95     }while(true);
96
97 }
98
99 }
```

```
ning Proficiency\Java programs\Group assignment\CarReadWrite.java • (Java programs) - Sublime Text
File Tools Project Preferences Help
CarReadWrite.java ManageCar.java
1 import java.util.*;
2 import java.io.*;
3
4 public class CarReadWrite {
5
6     static private String regisNo="", available="", carBrand="", model="", color"",
7         firstName="", lastName="", userNo="";
8     static private int mile=0, yearOfProduce=0;
9     static private String inputFile="carInput.txt", outputFile="output.txt";
10    static ArrayList<Car> carList = new ArrayList<Car>();
11
12    // The existing car rental records
13    // Must put "throws Exception" in method header
14    public static void readInput() throws Exception{
15
16        Scanner scanFile = new Scanner(new File(inputFile));
17
18        // Scan info from inputFile, add them into the carlist
19        while (scanFile.hasNextLine()){
20            regisNo = scanFile.next();
21            available = scanFile.next();
22            carBrand = scanFile.next();
23            model = scanFile.next();
24            color = scanFile.next();
25            while (scanFile.hasNextInt()){
26                mile = scanFile.nextInt();
27                yearOfProduce = scanFile.nextInt();
28            }
29        }
30    }
31
32    if (scanFile.hasNext()){
33        firstName = scanFile.next();
34    }
35    if (scanFile.hasNext()){
36        lastName = scanFile.next();
37        userNo = scanFile.next();
38    }
39    // add scanned info into the carlist
40    Car carItem = new Car(regisNo,available,carBrand,model,color,mile,
41        yearOfProduce,firstName,lastName,userNo);
42    carList.add(carItem);
43
44    scanFile.close();
45 } //End of readInput method
46
47 // Display the content in the array carList
48 public static void displayInput(){
49     int idx = 0;
50     while(idx < carList.size()){
51         System.out.println(carList.get(idx));
52         idx++;
53     }
54 } //end of displayInput method
55
56 // ASK USERS TO ADD NEW CAR RECORDS
57 public static void addCar() throws Exception{
58     Scanner newScan = new Scanner(System.in);
59     String addAgain;
60
61     do{
62
63         System.out.print("Enter Regis No: ");
64         regisNo = newScan.nextLine();
65
66         System.out.print("Enter Available: ");
67         available = newScan.nextLine();
68
69         System.out.print("Enter Car Brand: ");
70         carBrand = newScan.nextLine();
71
72         System.out.print("Enter Model: ");
73         model = newScan.nextLine();
74
75         System.out.print("Enter Color: ");
76         color = newScan.nextLine();
77
78         System.out.print("Enter Mileage: ");
79         mile = newScan.nextInt();
80
81         System.out.print("Enter Year Of Produce: ");
82         yearOfProduce = newScan.nextInt();
83
84         Car carItem = new Car(regisNo,available,carBrand,model,color,mile,
85             yearOfProduce,firstName,lastName,userNo);
86
87         carList.add(carItem);
88
89         System.out.print("Do you want to add again? (y/n): ");
90         addAgain = newScan.nextLine();
91
92         if (addAgain.equalsIgnoreCase("n")){
93             break;
94         }
95     }while(true);
96
97 }
98
99 }
```

### writeOutput method:

```
ning Proficiency\Java programs\Group assignment\CarReadWrite.java • (Java programs) - Sublime Text
File Tools Project Preferences Help
CarReadWrite.java ManageCar.java
89     carList.add(newItem);
90
91     //Check for entering next Loop
92     System.out.println("You have added a new car record: "+ "\n\n" + newItem + "\n");
93     System.out.println("Would you like to add more?");
94     addAgain = newScan.next();
95     if (addAgain.equalsIgnoreCase("no")){
96         break;
97     }
98
99 } while(addAgain.equalsIgnoreCase("yes"));
100
101 newScan.close();
102 } // End of addCar method
103
104 // SET UP THE WRITING OF THE OUTPUT FILE
105 public static void writeOutput() throws Exception{
106     PrintWriter output = new PrintWriter(new FileWriter(outputFile));
107     // Write data from carList including newly added and input file, to output file
108     int idx = 0;
109     while(idx < carList.size()){
110         output.println(carList.get(idx));
111         idx++;
112     }
113     // finish writing info to the output file
114     output.close();
115
116 } // End of writeOutput method
117
```

### 3.4 Borrow a Car

To borrow a car, the user needs to choose a car to borrow, and the program will prompt confirmation message to confirm whether the user wants to rent or not. If yes, the user can add the user name and phone number to borrow it. Once a car is borrowed, in the database, its availability status changes from yes to no.

```
Windows Powershell
Which car you would like to borrow?
Please enter Car No. or enter 99 to see all car records.

2
Car registration number: C2
Availability: Yes
Brand: Nissan
Model: Nissan01
Color: black
Mileage: 12000
Year of Produce: 2010
User Name: null null
User Number: null

Are you sure to borrow this car? (yes or no)
yes
Please enter your first name.
Ada
Please enter your last name.
Wu
Please enter your phone number.
0405836245
This is the car you borrowed.

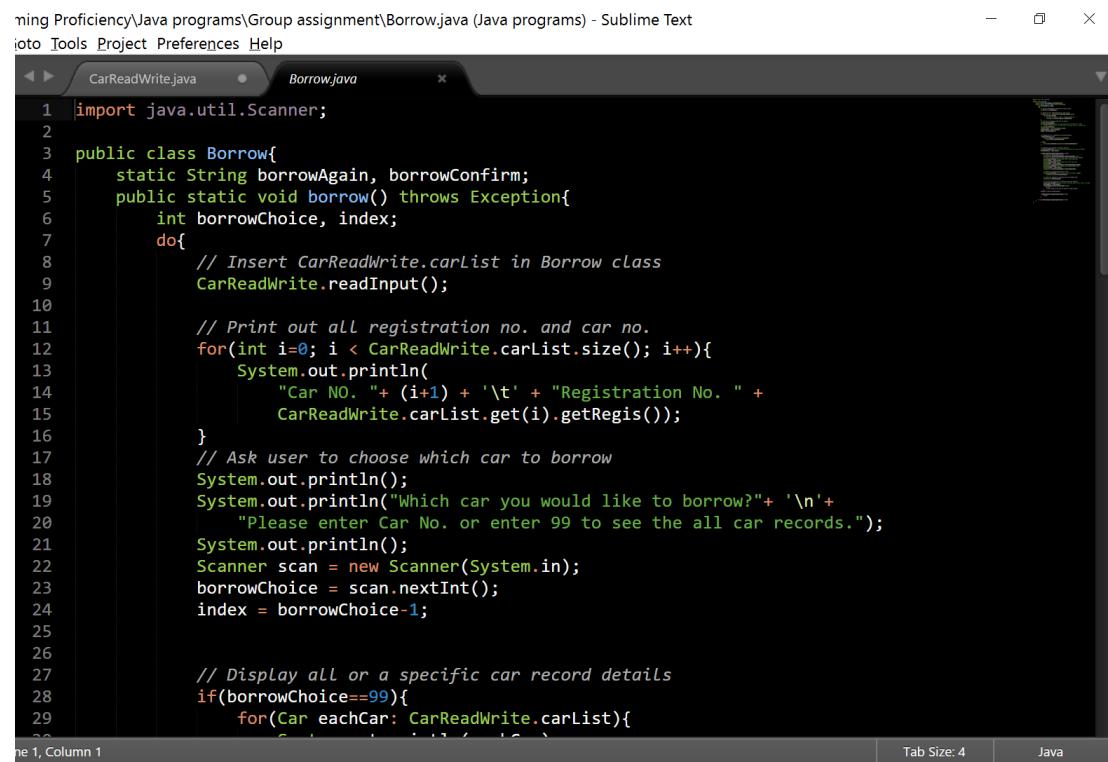
Car registration number: C2
Availability: no
Brand: Nissan
Model: Nissan01
Color: black
Mileage: 12000
Year of Produce: 2010
User Name: Ada Wu
User Number: 0405836245

Would you like to borrow other cars? (yes or no)
```

Regarding the backend part, first we insert carList from CarReadWrite class into Borrow class, then display all registration no. and car no. to the user by using for loop. Then we ask the user to choose a car to borrow, scan user's choice, and display the detailed car rental information to the user to confirm whether the user wants to borrow it or not.

If the user confirms to borrow a specific car, the system will modify the value of availability field of this car from yes to no. And ask the user to enter user name (first name as well as last name) and user phone number. Then use setters of this car object to add the user information and write them into the carList in the database. Since CarReadWrite class is where we access the database, we use static method and value to access carList by CarReadWrite.carList.get(index).setUserNo(userNo).

The source code of the borrow method in the Borrow class is displayed as below.



The screenshot shows a Sublime Text editor window with the file 'Borrow.java' open. The code implements a 'borrow' method in the 'Borrow' class. It starts by importing Scanner and reading input from CarReadWrite. It then prints all car records and asks the user to choose one. If the user chooses to see all records, it loops through the list. Otherwise, it sets the chosen car's availability to false and displays its details.

```
1 import java.util.Scanner;
2
3 public class Borrow{
4     static String borrowAgain, borrowConfirm;
5     public static void borrow() throws Exception{
6         int borrowChoice, index;
7         do{
8             // Insert CarReadWrite.carList in Borrow class
9             CarReadWrite.readInput();
10
11            // Print out all registration no. and car no.
12            for(int i=0; i < CarReadWrite.carList.size(); i++){
13                System.out.println(
14                    "Car NO. "+ (i+1) + '\t' + "Registration No. " +
15                    CarReadWrite.carList.get(i).getRegis());
16            }
17            // Ask user to choose which car to borrow
18            System.out.println();
19            System.out.println("Which car you would like to borrow?"+ '\n'+
20                "Please enter Car No. or enter 99 to see the all car records.");
21            System.out.println();
22            Scanner scan = new Scanner(System.in);
23            borrowChoice = scan.nextInt();
24            index = borrowChoice-1;
25
26
27            // Display all or a specific car record details
28            if(borrowChoice==99){
29                for(Car eachCar: CarReadWrite.carList){
30                    System.out.println(eachCar);
31                }
32            }
33        }while(borrowAgain.equals("yes"));
34    }
35 }
```

The screenshot shows a Sublime Text window with two tabs: 'CarReadWrite.java' and 'Borrow.java'. The 'Borrow.java' tab is active, displaying the following Java code:

```
30         System.out.println(eachCar);
31     }
32     } else{
33         System.out.println(CarReadWrite.carList.get(index));
34     }
35
36     // confirm user's choice to borrow this car
37     System.out.println("Are you sure to borrow this car? (yes or no)");
38     borrowConfirm = scan.next();
39
40     if(borrowConfirm.equalsIgnoreCase("yes")){
41         // reset the availability as no
42         CarReadWrite.carList.get(index).setAvailability("no");
43         // reset the first name and last name in borrowed car object
44         System.out.println("Please enter your first name.");
45         String fName = scan.next();
46         System.out.println("Please enter your last name.");
47         String lName = scan.next();
48         CarReadWrite.carList.get(index).setUserName(fName, lName);
49         System.out.println("Please enter your phone number.");
50         String userNo = scan.next();
51         CarReadWrite.carList.get(index).setUserNo(userNo);
52
53         // print the current borrowed car info
54         System.out.println("This is the car you borrowed."+"\n\n"+
55             CarReadWrite.carList.get(index));
56
57         // Write all objects in carList into the output file
58         CarReadWrite.writeOutput();
59
60         // ask user if they want to borrow other car records
61         System.out.println("Would you like to borrow other cars? (yes or no)");
62         borrowAgain = scan.next();
63         if(borrowAgain.equalsIgnoreCase("no")){
64             break;
65             // goes back to the end of switch in main method
66         }
67     }//End of borrow confirmation
68
69     if(borrowConfirm.equalsIgnoreCase("no")){
70         break;
71     }
72
73 } while(borrowAgain.equalsIgnoreCase("yes"));
74
75 }
```

The screenshot shows a Sublime Text window with two tabs: 'CarReadWrite.java' and 'Borrow.java'. The 'Borrow.java' tab is active, displaying the following Java code:

```
59
60     // ask user if they want to borrow other car records
61     System.out.println("Would you like to borrow other cars? (yes or no)");
62     borrowAgain = scan.next();
63     if(borrowAgain.equalsIgnoreCase("no")){
64         break;
65         // goes back to the end of switch in main method
66     }
67 //End of borrow confirmation
68
69     if(borrowConfirm.equalsIgnoreCase("no")){
70         break;
71     }
72
73 } while(borrowAgain.equalsIgnoreCase("yes"));
74
75 }
```

### 3.5 Modify Car Rental Details

To modify a car rental record, first the user need to choose which car to modify. The user can enter a car no. or enter 99 to see all car records. Then the system will ask which field the user want to modify. The user can enter a field name according the options provided by the system, and the system is not case sensitive so it accepts title case, upper case, and lower case.

```

Which car you would like to modify?
Please enter Car No. or enter 99 to see the all car records.

5
Car registration number: C5
Availability: No
Brand: Ferrari
Model: Ferrari01
Color: silver
Mileage: 8222
Year of Produce: 1998
User Name: Tina Zhao
User Number: 0403827463

Please enter a field to modify
(regisNo/availability/carBrand/model/color/userName/userNo/mile/yearOfProduce).
Or enter "exit" to exit.
resigNo
Please enter registration number (C1 to C10).
C11

This is the modified result.

```

```

Car registration number: C11
Availability: No
Brand: Ferrari
Model: Ferrari01
Color: silver
Mileage: 8222
Year of Produce: 1998
User Name: Tina Zhao
User Number: 0403827463

Would you like to modify more?
yes
Car NO. 1 Registration No. C1
Car NO. 2 Registration No. C2
Car NO. 3 Registration No. C3
Car NO. 4 Registration No. C4
Car NO. 5 Registration No. C11
Car NO. 6 Registration No. C6
Car NO. 7 Registration No. C7
Car NO. 8 Registration No. C8
Car NO. 9 Registration No. C9
Car NO. 10 Registration No. C10

Which car you would like to modify?
Please enter Car No. or enter 99 to see the all car records.

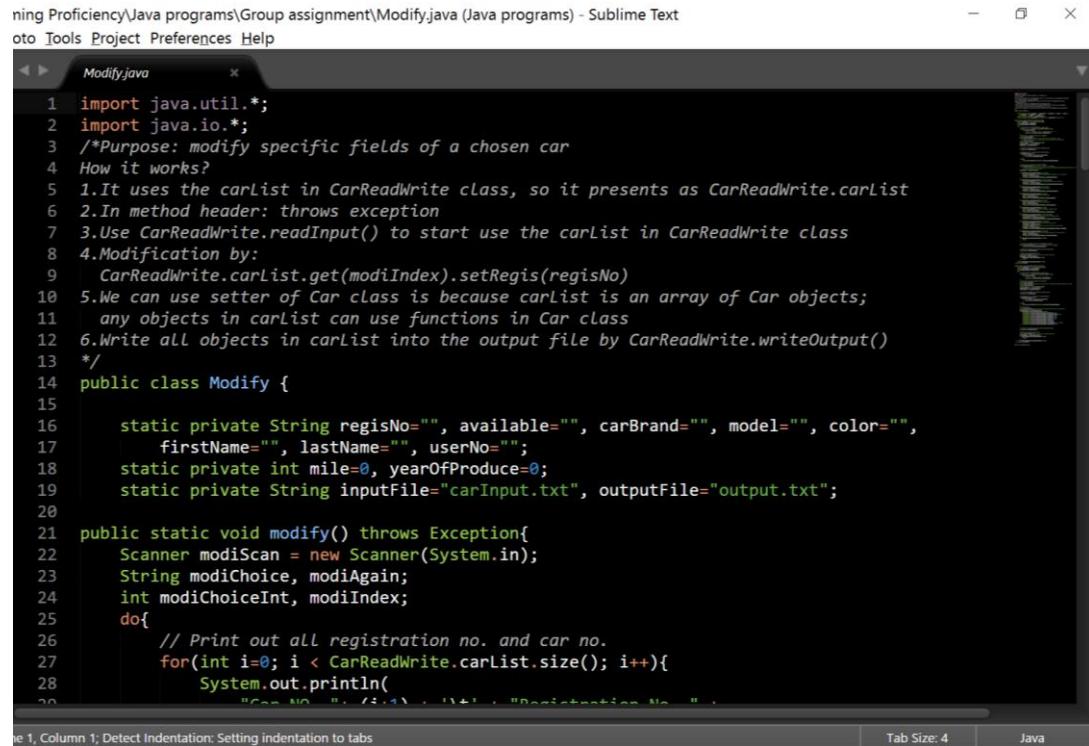
5
Car registration number: C11
Availability: No
Brand: Ferrari
Model: Ferrari01
Color: silver
Mileage: 8222
Year of Produce: 1998

```

For the backend technology of the modify method in Modify class, there are a few basic steps for design. Firstly, the system uses the carList in CarReadWrite class, so it presents as CarReadWrite.carList. And although it uses static variable from CarReadWrite class, it still need to declare *throws exception* in method header to make the system run. Secondly, it uses CarReadWrite.readInput() to start use the carList in CarReadWrite class. Thirdly, the modification is done by

`CarReadWrite.carList.get(modIndex).setRegis(regisNo)`. Fourth, we can use setter of Car class is because carList is an array of Car objects, and any objects in carList can use functions in Car class. Fifth, write all objects in carList into the output file by `CarReadWrite.writeOutput()`.

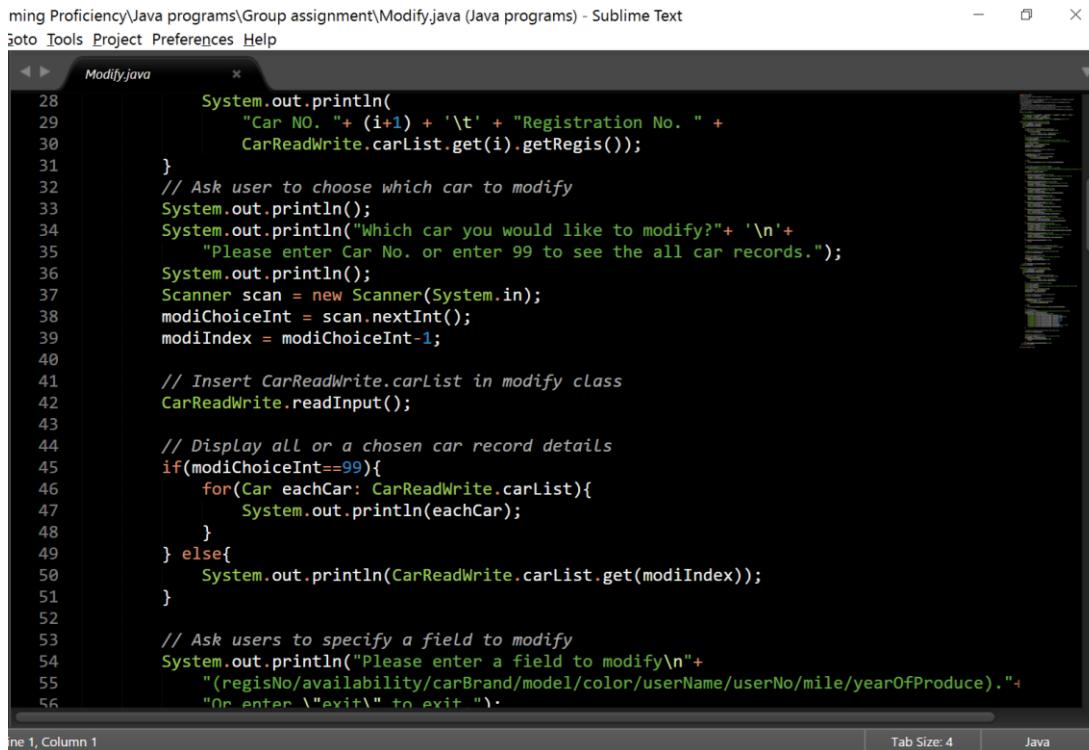
Source code of modify method in Modify class:



```

1 import java.util.*;
2 import java.io.*;
3 /*Purpose: modify specific fields of a chosen car
4 How it works?
5 1.It uses the carList in CarReadWrite class, so it presents as CarReadWrite.carList
6 2.In method header: throws exception
7 3.Use CarReadWrite.readInput() to start use the carList in CarReadWrite class
8 4.Modification by:
9     CarReadWrite.carList.get(modIndex).setRegis(regisNo)
10 5.We can use setter of Car class is because carList is an array of Car objects;
11     any objects in carList can use functions in Car class
12 6.Write all objects in carList into the output file by CarReadWrite.writeOutput()
13 */
14 public class Modify {
15
16     static private String regisNo="", available="", carBrand="", model="", color"",
17         firstName="", lastName="", userNo="";
18     static private int mile=0, yearOfProduce=0;
19     static private String inputFile="carInput.txt", outputFile="output.txt";
20
21     public static void modify() throws Exception{
22         Scanner modiScan = new Scanner(System.in);
23         String modiChoice, modiAgain;
24         int modiChoiceInt, modiIndex;
25         do{
26             // Print out all registration no. and car no.
27             for(int i=0; i < CarReadWrite.carList.size(); i++){
28                 System.out.println(
29                     "Car NO. " + (i+1) + '\t' + "Registration No. " +
30                     CarReadWrite.carList.get(i).getRegis());
31             }
32             // Ask user to choose which car to modify
33             System.out.println();
34             System.out.println("Which car you would like to modify?" + '\n' +
35                 "Please enter Car No. or enter 99 to see the all car records.");
36             System.out.println();
37             Scanner scan = new Scanner(System.in);
38             modiChoiceInt = scan.nextInt();
39             modiIndex = modiChoiceInt-1;
40
41             // Insert CarReadWrite.carList in modify class
42             CarReadWrite.readInput();
43
44             // Display all or a chosen car record details
45             if(modiChoiceInt==99){
46                 for(Car eachCar: CarReadWrite.carList){
47                     System.out.println(eachCar);
48                 }
49             } else{
50                 System.out.println(CarReadWrite.carList.get(modiIndex));
51             }
52
53             // Ask users to specify a field to modify
54             System.out.println("Please enter a field to modify\n" +
55                 "(regisNo/availability/carBrand/model/color/userName/userNo/mile/yearOfProduce).")
56                 "Or enter \"exit\" to exit ");

```



```

28         System.out.println(
29             "Car NO. " + (i+1) + '\t' + "Registration No. " +
30             CarReadWrite.carList.get(i).getRegis());
31     }
32     // Ask user to choose which car to modify
33     System.out.println();
34     System.out.println("Which car you would like to modify?" + '\n' +
35         "Please enter Car No. or enter 99 to see the all car records.");
36     System.out.println();
37     Scanner scan = new Scanner(System.in);
38     modiChoiceInt = scan.nextInt();
39     modiIndex = modiChoiceInt-1;
40
41     // Insert CarReadWrite.carList in modify class
42     CarReadWrite.readInput();
43
44     // Display all or a chosen car record details
45     if(modiChoiceInt==99){
46         for(Car eachCar: CarReadWrite.carList){
47             System.out.println(eachCar);
48         }
49     } else{
50         System.out.println(CarReadWrite.carList.get(modiIndex));
51     }
52
53     // Ask users to specify a field to modify
54     System.out.println("Please enter a field to modify\n" +
55         "(regisNo/availability/carBrand/model/color/userName/userNo/mile/yearOfProduce).")
56         "Or enter \"exit\" to exit ");

```

mming Proficiency\Java programs\Group assignment\Modify.java (Java programs) - Sublime Text

Goto Tools Project Preferences Help

```

56         "Or enter \"exit\" to exit.");
57         modiChoice = modiScan.next();
58
59     if (modiChoice.equalsIgnoreCase("resigNo")){
60         System.out.println("Please enter registration number (C1 to C10).");
61         regisNo = modiScan.next();
62         CarReadWrite.carList.get(modiIndex).setRegis(regisNo);
63     }
64     if (modiChoice.equalsIgnoreCase("availability")){
65         System.out.println("Please enter availability (yes or no).");
66         available = modiScan.next();
67         CarReadWrite.carList.get(modiIndex).setAvailability(available);
68     }
69     if (modiChoice.equalsIgnoreCase("carBrand")){
70         System.out.println("Please enter brand.");
71         carBrand = modiScan.next();
72         CarReadWrite.carList.get(modiIndex).setBrand(carBrand);
73     }
74     if (modiChoice.equalsIgnoreCase("model")){
75         System.out.println("Please enter model.");
76         model = modiScan.next();
77         CarReadWrite.carList.get(modiIndex).setModel(model);
78     }
79     if (modiChoice.equalsIgnoreCase("color")){
80         System.out.println("Please enter color.");
81         color = modiScan.next();
82         CarReadWrite.carList.get(modiIndex).setColor(color);
83     }
84     if (modiChoice.equalsIgnoreCase("mile")){
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125

```

Line 1, Column 1 Tab Size: 4 Java

ning Proficiency\Java programs\Group assignment\Modify.java (Java programs) - Sublime Text

Goto Tools Project Preferences Help

```

84     if (modiChoice.equalsIgnoreCase("mile")){
85         System.out.println("Please enter mileage (number).");
86         mile = modiScan.nextInt();
87         CarReadWrite.carList.get(modiIndex).setMile(mile);
88     }
89     if (modiChoice.equalsIgnoreCase("yearOfProduce")){
90         System.out.println("Please enter the year of produce.");
91         yearOfProduce = modiScan.nextInt();
92         CarReadWrite.carList.get(modiIndex).setYear(yearOfProduce);
93     }
94     if (modiChoice.equalsIgnoreCase("userName")){
95         System.out.println("Please enter the first name.");
96         firstName = modiScan.next();
97         System.out.println("Please enter the last name of the user.");
98         lastName = modiScan.next();
99         CarReadWrite.carList.get(modiIndex).setUserName(firstName,lastName);
100    }
101    if (modiChoice.equalsIgnoreCase("userNo")){
102        System.out.println("Please enter the phone number of the user.");
103        userNo = modiScan.next();
104        CarReadWrite.carList.get(modiIndex).setUserNo(userNo);
105    }
106    if (modiChoice.equalsIgnoreCase("exit")){
107        break;
108    }
109
110 // Print the modified item in CarReadWrite.carList
111 System.out.println("\nThis is the modified result."+"\n\n"+
112
113
114
115
116
117
118
119
120
121
122
123
124
125

```

Line 1, Column 1 Tab Size: 4 Java

```

112
113
114
115
116
117
118
119
120
121
122
123
124
125

```

### 3.6 Delete Car Rental Details

If the user wants to delete a car record, it can be executed simply by entering the car number. For the backend, the system will set all attribute values in a car object into null.

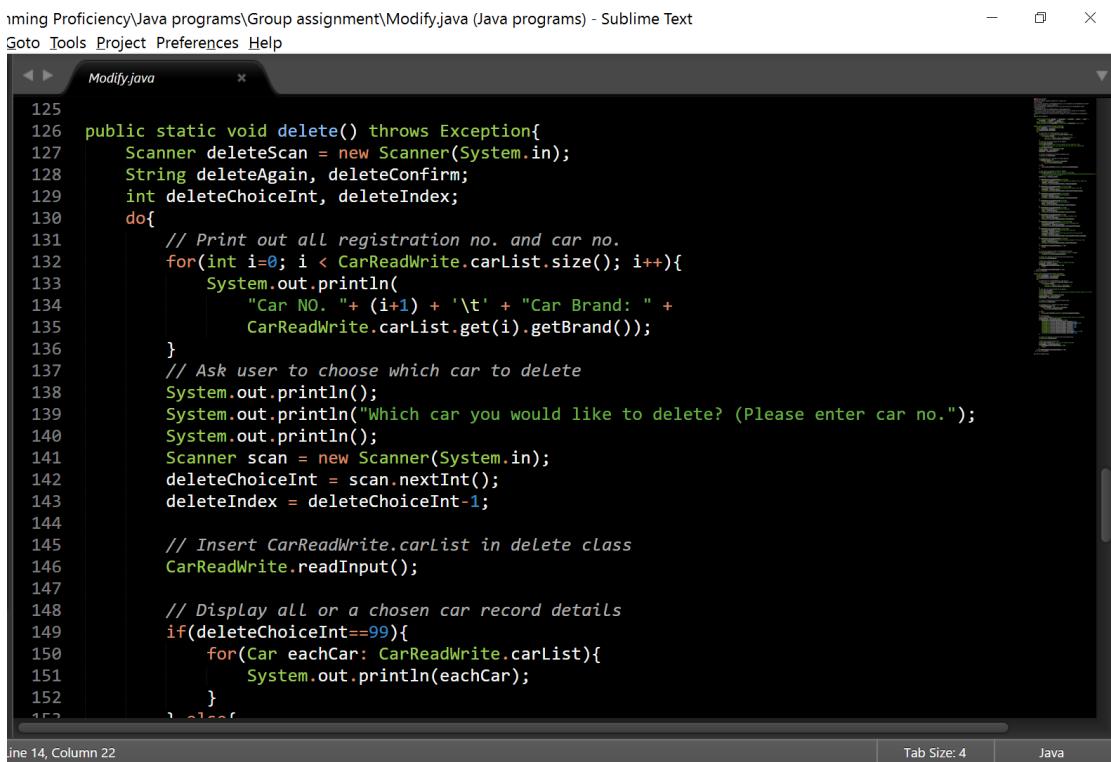
```
Please enter the service number you would like to use (from 1 to 9).
5

Which car you would like to delete? (Please enter car no.

5
Car registration number: C5
Availability: No
Brand: Ferrari
Model: Ferrari01
Color: silver
Mileage: 8222
Year of Produce: 1998
User Name: Tina Zhao
User Number: 0403827463

Are you sure you wanna delete this car record?
yes
Would you like to delete more?
```

The delete method is inside the Modify class, and the only difference between it and the modify method is that the delete method sets all fields in a car record into null. In this way, it implies the deletion of a car rental record. Here is the source code of the delete method in Modify class.



The screenshot shows the Java code for the `delete()` method in the `Modify.java` file. The code uses a `Scanner` to read input from the user. It prints out all car details, asks the user to choose which car to delete, and then sets the chosen car's attributes to null. The code is part of a larger class that interacts with a `CarReadWrite` class.

```
125
126 public static void delete() throws Exception{
127     Scanner deleteScan = new Scanner(System.in);
128     String deleteAgain, deleteConfirm;
129     int deleteChoiceInt, deleteIndex;
130     do{
131         // Print out all registration no. and car no.
132         for(int i=0; i < CarReadWrite.carList.size(); i++){
133             System.out.println(
134                 "Car NO. "+(i+1) + '\t' + "Car Brand: " +
135                 CarReadWrite.carList.get(i).getBrand());
136         }
137         // Ask user to choose which car to delete
138         System.out.println();
139         System.out.println("Which car you would like to delete? (Please enter car no.");
140         System.out.println();
141         Scanner scan = new Scanner(System.in);
142         deleteChoiceInt = scan.nextInt();
143         deleteIndex = deleteChoiceInt-1;
144
145         // Insert CarReadWrite.carlist in delete class
146         CarReadWrite.readInput();
147
148         // Display all or a chosen car record details
149         if(deleteChoiceInt==99){
150             for(Car eachCar: CarReadWrite.carList){
151                 System.out.println(eachCar);
152             }
153         }
154     } while(deleteAgain.equals("y"));
155 }
```

mming Proficiency\Java programs\Group assignment\Modify.java (Java programs) - Sublime Text

Goto Tools Project Preferences Help

```

153         } else{
154             System.out.println(CarReadWrite.carList.get(deleteIndex));
155         }
156         // Confirmation
157         System.out.println("Are you sure you wanna delete this car record?");
158         deleteConfirm = deleteScan.next();
159         if (deleteConfirm.equalsIgnoreCase("yes")){
160             CarReadWrite.carList.get(deleteIndex).setRegis(null);
161             CarReadWrite.carList.get(deleteIndex).setAvailability(null);
162             CarReadWrite.carList.get(deleteIndex).setBrand(null);
163             CarReadWrite.carList.get(deleteIndex).setModel(null);
164             CarReadWrite.carList.get(deleteIndex).setColor(null);
165             CarReadWrite.carList.get(deleteIndex).setMile(0);
166             CarReadWrite.carList.get(deleteIndex).setYear(0);
167             CarReadWrite.carList.get(deleteIndex).setUserName(null, null);
168             CarReadWrite.carList.get(deleteIndex).setUserNo(null);
169         }
170         // Write all objects in carList into the output file
171         CarReadWrite.writeOutput();
172
173         //Check for entering next loop
174         System.out.println("Would you like to delete more?");
175         deleteAgain = deleteScan.next();
176         if (deleteAgain.equalsIgnoreCase("no")){
177             break;
178         }
179     } while(deleteAgain.equalsIgnoreCase("yes"));
180
181 
```

Line 14, Column 22

Tab Size: 4 Java

### 3.7 Search car rental details

User can search cars by providing a search word for a field, such as registration number, availability, car brand, car color, year of produce, etc. Once the user decide which field to search, the system will ask a search word from the user, then display the search result. In the end, the user can decide whether to search again.

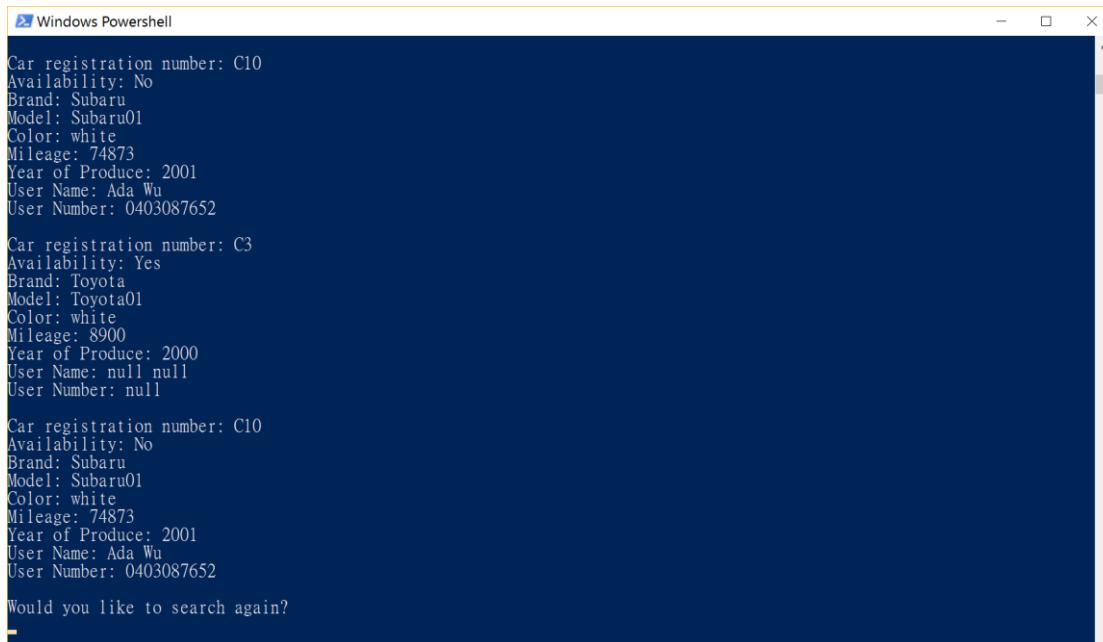
Windows Powershell

```

1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).
6
Whcih field would you like to search? Please enter 1 to 10.
1. Registration No.
2. Availability
3. Car Brand
4. Car Model
5. Car color
6. Mileage
7. Year of Produce
8. First Name of the User
9. Last Name of the User
10. User Number
5
Please type in a search word.
(Enter a number for searching mileage or year of produce)
white
This is the car you are searching for:

Car registration number: C3
Availability: Yes
Brand: Toyota
Model: Toyota01
Color: white
Mileage: 8900
Year of Produce: 2000
User Name: null null

```



```

Windows PowerShell

Car registration number: C10
Availability: No
Brand: Subaru
Model: Subaru01
Color: white
Mileage: 74873
Year of Produce: 2001
User Name: Ada Wu
User Number: 0403087652

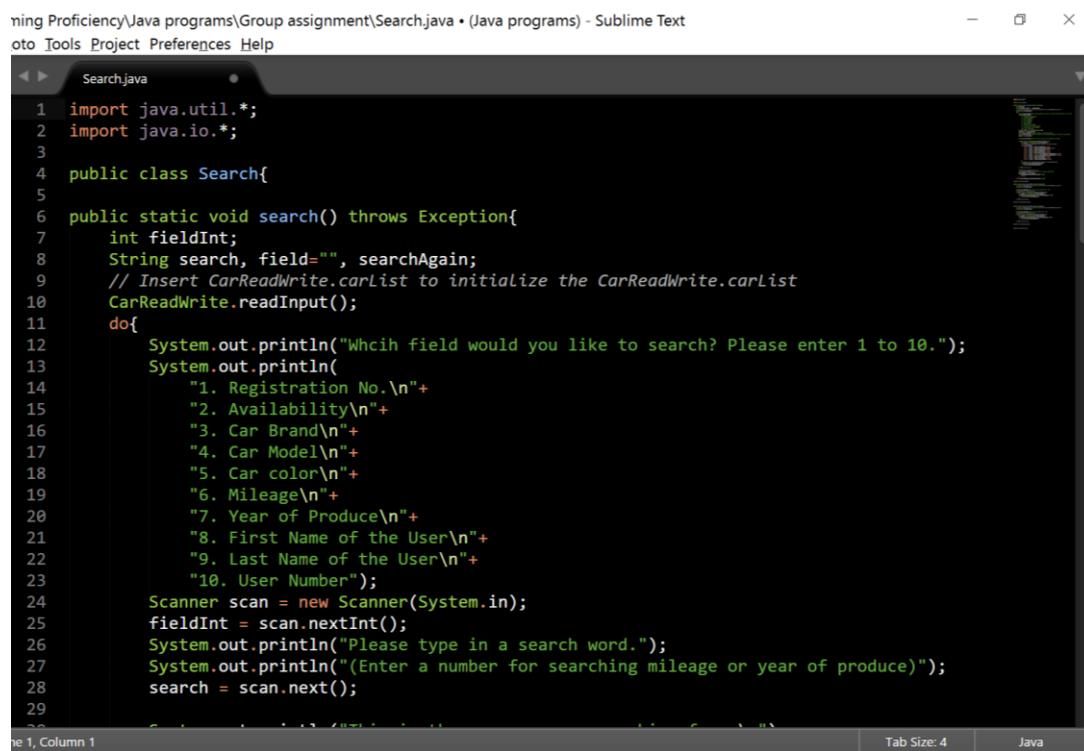
Car registration number: C3
Availability: Yes
Brand: Toyota
Model: Toyota01
Color: white
Mileage: 8900
Year of Produce: 2000
User Name: null null
User Number: null

Car registration number: C10
Availability: No
Brand: Subaru
Model: Subaru01
Color: white
Mileage: 74873
Year of Produce: 2001
User Name: Ada Wu
User Number: 0403087652

Would you like to search again?

```

For the backend technology, Search class is designed highly relying on the CarReadWrite class, which is used for accessing the database. During execution, the Search class firstly reads all car rental records from database via the function readInput in CarReadWrite class. According to the user's choice about which field to search, the system use switch statement to decide which getter function to call and match it with the given search word. If there is any matched result, it will be printed out automatically. The source code is shown as below.



```

ning Proficiency\Java programs\Group assignment\Search.java • (Java programs) - Sublime Text
File Tools Project Preferences Help
Search.java
1 import java.util.*;
2 import java.io.*;
3
4 public class Search{
5
6     public static void search() throws Exception{
7         int fieldInt;
8         String search, field="";
9         // Insert CarReadWrite.carList to initialize the CarReadWrite.carList
10        CarReadWrite.readInput();
11        do{
12            System.out.println("Whcih field would you like to search? Please enter 1 to 10.");
13            System.out.println(
14                "1. Registration No.\n"+
15                "2. Availability\n"+
16                "3. Car Brand\n"+
17                "4. Car Model\n"+
18                "5. Car color\n"+
19                "6. Mileage\n"+
20                "7. Year of Produce\n"+
21                "8. First Name of the User\n"+
22                "9. Last Name of the User\n"+
23                "10. User Number");
24            Scanner scan = new Scanner(System.in);
25            fieldInt = scan.nextInt();
26            System.out.println("Please type in a search word.");
27            System.out.println("(Enter a number for searching mileage or year of produce)");
28            search = scan.next();
29
30

```

ming Proficiency\Java programs\Group assignment\Search.java • (Java programs) - Sublime Text

Goto Tools Project Preferences Help

```

29         System.out.println("This is the car you are searching for: \n");
30
31     for(int i=0; i<CarReadWrite.carList.size();i++){
32         Car object = CarReadWrite.carList.get(i);
33         // according to different user choice, initialize field
34         switch(fieldInt){
35             case 1: field = object.getRegis(); break;
36             case 2: field = object.getAvailability(); break;
37             case 3: field = object.getBrand(); break;
38             case 4: field = object.getModel(); break;
39             case 5: field = object.getColor(); break;
40             case 6: field = Integer.toString(object.getMile()); break;
41             case 7: field = Integer.toString(object.getYear()); break;
42             case 8: field = object.getFName(); break;
43             case 9: field = object.getLName(); break;
44             case 10: field = object.getUserNo(); break;
45         }
46         // check if the assigned field equals to the search word
47         if(search.equalsIgnoreCase(field)){
48             System.out.println(object);
49         }
50     } //End of for Loop
51
52     //Check for entering next Loop
53     System.out.println("Would you like to search again?");
54     searchAgain = scan.next();
55     if (searchAgain.equalsIgnoreCase("no")){
56         break;
57     }
58 }
```

line 1, Column 1 | Tab Size: 4 | Java

mming Proficiency\Java programs\Group assignment\Search.java • (Java programs) - Sublime Text

Goto Tools Project Preferences Help

```

58     }
59     } while(searchAgain.equalsIgnoreCase("yes"));
60
61 }//End of search method
62 }
```

### 3.8 List Available Cars for Rent

Within the Search class, the function available can display all cars ready to rent. The user interface while execution looks as below.

```

6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).
7
Those are the avaialbe cars:
Car registration number: C1
Availability: Yes
Brand: Benz
Model: Benz01
Color: black
Mileage: 4800
Year of Produce: 2014
User Name: null null
User Number: null

Car registration number: C2
Availability: Yes
Brand: Nissan
Model: Nissan01
Color: black
Mileage: 12000
Year of Produce: 2010
User Name: null null
User Number: null

Car registration number: C3
Availability: Yes
Brand: Toyota
Model: Toyota01
Color: white
Mileage: 8900
Year of Produce: 2000

```

```

Car registration number: C4
Availability: Yes
Brand: Audi
Model: Audi01
Color: yellow
Mileage: 8900
Year of Produce: 1993
User Name: null null
User Number: null

Car registration number: C6
Availability: Yes
Brand: Mazda
Model: Mazda01
Color: blue
Mileage: 30044
Year of Produce: 2013
User Name: null null
User Number: null

```

```

Car registration number: C9
Availability: Yes
Brand: Peugeot
Model: Peugeot01
Color: grey
Mileage: 130229
Year of Produce: 1999
User Name: null null
User Number: null

1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).

```

For the backend part, with similar strategies used in the search function, the system examine which car's availability field equals to yes and display them to the user. For loop is used to visit all car records/objects in the database.

```

62
63 public static void available() throws Exception{
64     // Insert CarReadWrite.carList to initialize the CarReadWrite.carList
65     CarReadWrite.readInput();
66
67     System.out.println("Those are the avaialbe cars: ");
68     // Display all available cars
69     for(int i=0; i<CarReadWrite.carList.size();i++){
70         Car object = CarReadWrite.carList.get(i);
71         if(object.getAvailability().equalsIgnoreCase("yes")){
72             System.out.println(object);
73         }
74     } //End of for Loop
75
76 } //End of available method
77

```

Line 1, Column 1

Tab Size: 4

Java

### 3.9 List All Borrowed Cars

Similar with list all available car, the only difference between the borrowed function and the available function is that the system will examine which car's availability field equals to no and display them to the user. For loop is used to visit all car records/objects in the database.

```

Windows Powershell
Please enter the service number you would like to use (from 1 to 9).
8
Those are borrowed cars:
Car registration number: C5
Availability: No
Brand: Ferrari
Model: Ferrari01
Color: silver
Mileage: 8222
Year of Produce: 1998
User Name: Tina Zhao
User Number: 0403827463

Car registration number: C10
Availability: No
Brand: Subaru
Model: Subaru01
Color: white
Mileage: 74873
Year of Produce: 2001
User Name: Ada Wu
User Number: 0403087652

1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).

```

```

78
79 public static void borrowed() throws Exception{
80     // Insert CarReadWrite.carList to initialize the CarReadWrite.carList
81     CarReadWrite.readInput();
82
83     System.out.println("Those are borrowed cars: ");
84     // Display all available cars
85     for(int i=0; i<CarReadWrite.carList.size();i++){
86         Car object = CarReadWrite.carList.get(i);
87         if(object.getAvailability().equalsIgnoreCase("no")){
88             System.out.println(object);
89         }
90     } //End of for Loop
91
92 } //End of borrowed method
93

```

### 3.10 Exit

In the main menu, if the user enters 9, the system will terminate.

```

Windows Powershell
1. View car rental details
2. Add new car rental record
3. Borrow a car
4. Modify car rental details
5. Delete car rental details
6. Search car rental details
7. List available cars for rent
8. List all borrowed cars
9. Exit
Please enter the service number you would like to use (from 1 to 9).
9
PS C:\Users\Ada\Desktop\Programming Proficiency\Java programs\Group assignment>

```

## 4. Introduction to Java programming language

Concepts Classes and Objects.

- **Object** - Objects have states and behaviors. Example in this project: A Car has states-color, model, as well as behaviors –mileage. An object is an instance of a class.

- **Class** - A class can be defined as a blue print that describe the behaviors/states that object of its type support. Classes in Java:

`toString` method use to return all the information in car. Private variable is used to encapsulate data as a consequence no one can access the variable without the class name. Setter and getter method use to set and get private variables.

### **Constructors**

Every class has a constructor. If we do not explicitly write a constructor for a class the java compiler builds a default constructor for that class.

Each time a new object is created at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Example of a constructor is given below:

```
class Car{
    Declare class variables
    ivate String regisNo, available, carBrand, model, color, firstName, lastName, userNo;
    ivate int mile, yearOfProduce;

    Constructor, which initializes class variables
    blic Car (String regis, String avai, String brand, String mName, String colorName,
              int mileage, int yProduce, String fName, String lName, String uNo){
        regisNo = regis;
        available = avai;
        carBrand = brand;
        model = mName;
        color = colorName;
        mile = mileage;
        yearOfProduce = yProduce;
        firstName = fName;
        lastName = lName;
        userNo = uNo;
    }
}
```

### **Import statements that used in this project:**

#### **Loop Control statements**

Java has very flexible three looping mechanisms.

#### **The do...while Loop**

A do-while loop is a control structure that allows user to repeat a task a certain number of times. In do...while loop is guaranteed to execute at least one time.

Syntax:

The syntax of a do...while loop is:

```
do
{ //Statements } while(Boolean_exenterion);
```

In this program, user may choice things from menu. If user select 1, it will show user all existing car rental records. Then user can choose again until user put 8.

### ***The for Loop***

A for loop is a repetition control structure that allows user to efficiently write a loop that needs to execute a specific number of times.

A for loop is useful when user know how many times a task is to be repeated.

Syntax:

The syntax of a for loop is:

```
for(initialization; Boolean_exenterion; update)
{ //Statements }
```

### ***Enhanced for loop in Java***

For each loop uses when it is necessary to print the results only without modification of the variables in array.

Syntax:

The syntax of enhanced for loop is:

```
for(declaration : exenterion)
{ //Statements }
```

### ***The if Statement***

An if statement consists of a Boolean exenterion followed by one or more statements.

Syntax:

The syntax of an if statement is:

### ***The if...else Statement***

An if statement can be followed by an optional *else* statement, which executes when the Boolean exenterion is false.

### ***The switch Statement***

A *switch* statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

Syntax:

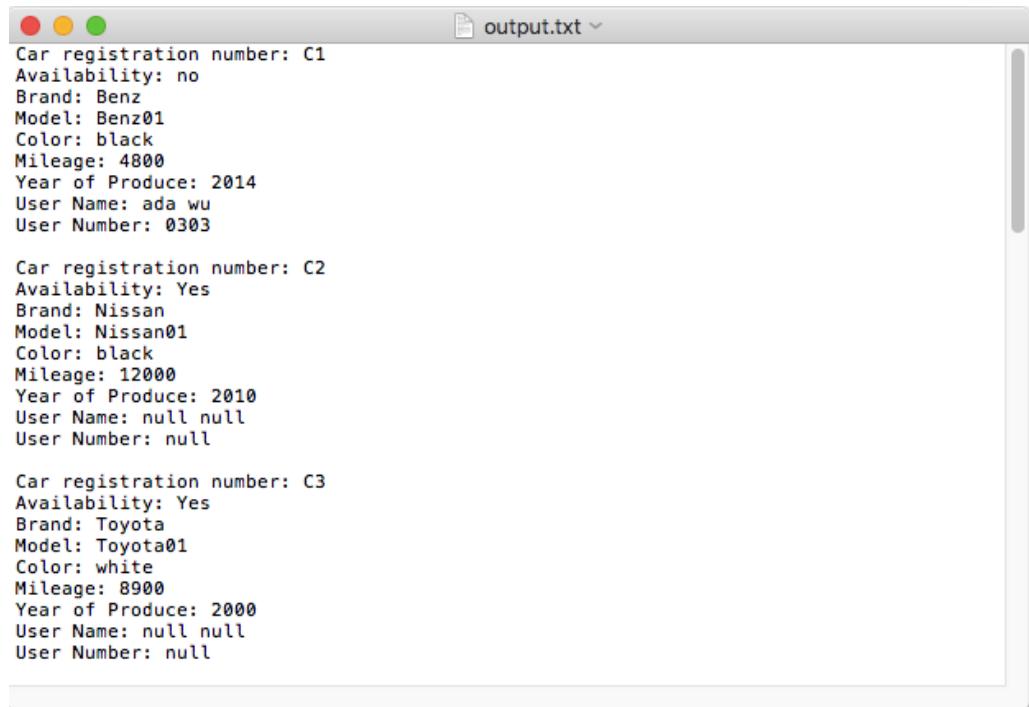
### ***Interface***

An interface is used to disclose functionality of the object without revealing its implementation. It is the concept of encapsulation.

### ***Exception handling***

Throws exception is the method of Java exception handling, which is a problem that occurs on the time of execution of program. As a consequence, the normal flow of the program interrupted and terminates the program. In this program, throws exception keyword use to read and write file methods in CarReadWrite class to read data from a file, if the file specified in its constructor doesn't exist, then a *FileNotFoundException* occurs, and the compiler prompts the programmer to handle the exception.

The write method is used the file in .text file. In the picture, output.txt files contain car rental records.



```
Car registration number: C1
Availability: no
Brand: Benz
Model: Benz01
Color: black
Mileage: 4800
Year of Produce: 2014
User Name: ada wu
User Number: 0303

Car registration number: C2
Availability: Yes
Brand: Nissan
Model: Nissan01
Color: black
Mileage: 12000
Year of Produce: 2010
User Name: null null
User Number: null

Car registration number: C3
Availability: Yes
Brand: Toyota
Model: Toyota01
Color: white
Mileage: 8900
Year of Produce: 2000
User Name: null null
User Number: null
```

### ArrayList

In this program, ArrayList is used, which supports dynamic arrays that can grow as needed. Standard Java arrays are of a fixed length. Here, carList is an Array List that used to store Car objects.

## Conclusion

To build this project, firstly, an interface is created that contain car registration number, its make, model, year, a flag indicating if the car is currently available for borrowing, and a borrower number and the corresponding borrower name if its borrowed. Secondly, it is imperative to create a main function that call another methods and execute the main code. After that user can choose the menu and based on user input, it executes and show the output. Thirdly, File read and write method use to load existing car records and if user add new cars, it automatically saves it as well. can save the rental records in .text file. The core Java object oriented concept is used to make this project.

The main objectives of this project is to build core java based car rental system program. An interface is used that helped to provide outline what class should use. Java object oriented principle is also used that helped to divide program in various class and made it more logical and understandable. There are also plenty of comments provided not only for programmer but also user to understand the program.

Java encapsulation and exception handling principles are used in this project, in order to hide the variables and throw exception. Another most important thing in this program is ArrayList that hold Car objects and display it. Lastly FileRead and Write methods are used to store data permanently in the computer on .text file.

#### Feature of Project and its Advantages

- Provides easy registration for the users
- User can rent car based on their desire car model
- Owner can add new car records and it saves to .text file.
- Wide range of search field to search car records and even user can modify records as well.
- User can see which car is available for rent.
- Most important, similar kinds of project might make buy following this project like DVD rental system, Library book rental system.

This project can be extended more for example more variable might be added in Car field and the program might be attached in SQL server. On the other hand, GUI (Graphical User Interface might be used to build this application more aesthetic and the person who does not have any programming knowledge, even can easily be able to operate the program.

## Bibliography

Baesens, Ba 2015, Beginning Java programming : the object-oriented approach, Indianapolis, Indiana : Wrox, 2015.

Brett Spell, T 2015, Pro Java 8 Programming, Berkeley, CA : Apress, 2015.

Farrell, Ja 2016, Java programming, Eighth edition. edn, Boston, Massachusetts : Cengage Learning, 2016.

Hillar, GnC 2015, Learning Object-Oriented Programming, Community experience distilled, Birmingham : Packt Publishing, Birmingham.

Lewis, Ja 2015, Java software solutions : foundations of program design, Eight edition. edn, BOSTON PEARSON

Boston : Pearson, 2015, BOSTON.

Sarcar, V 2015, Java Design Patterns, The expert's voice in Java, Berkeley, CA : Apress : Imprint: Apress, 2016.

Urma, R-Ga 2015, Java 8 in action : lambdas, streams, and functional-style programming, Shelter Island : Manning, 2015.

Wu, CT 2010, An introduction to object-oriented programming with Java, 5th ed. edn, Dubuque, Iowa : McGraw-Hill Higher Education, Dubuque, Iowa.