Table of Contents

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.

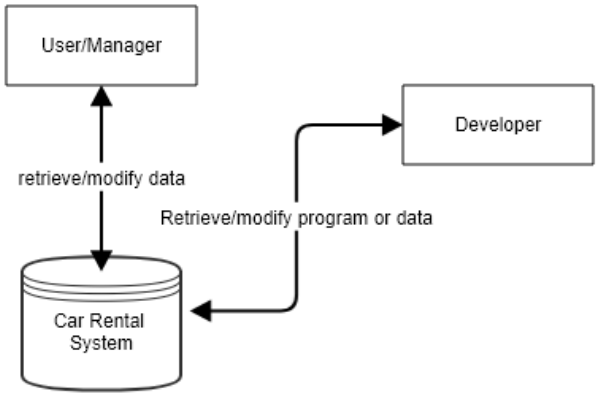


Figure 1 Data Flow on Context Level

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.

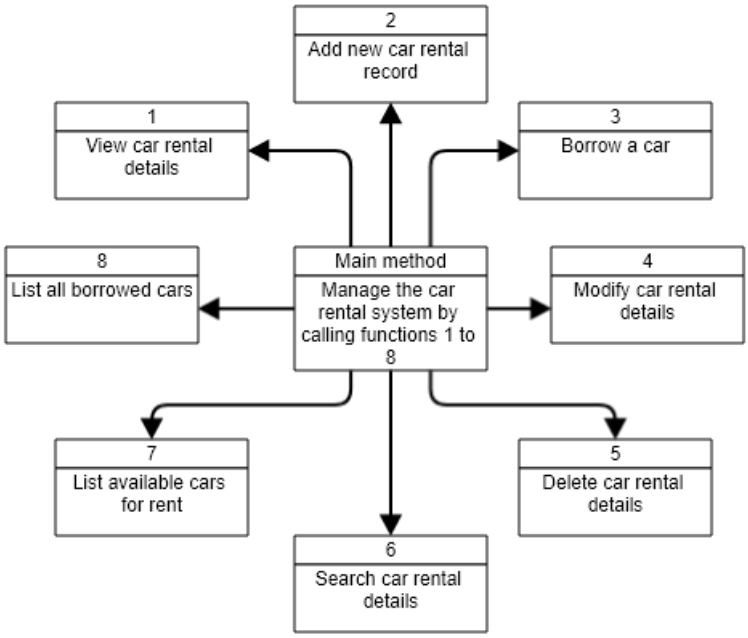


Figure 2 Control Flow in General

2.3 UML (Unified Modeling Language)

For each class in the car rental system, the model of the structure, attributes, and behavior are diagramed 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.

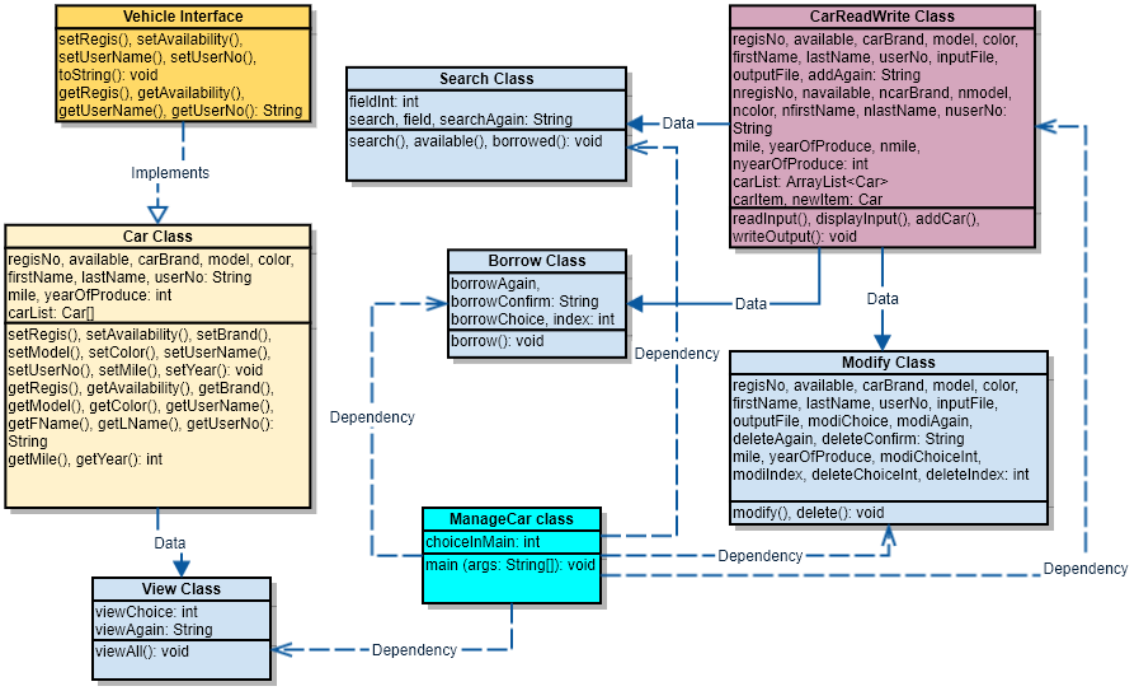


Figure 3 UML of the Car Rental System

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.

CarReadWrite.java

```
15
16      Scanner scanFile = new Scanner(new File(inputFile));
17
18      // Scan info from inputFile, add them intot 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          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
```

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).
```

ming Proficiency\Java programs\Group assignment\ManageCar.java (Java programs) - Sublime Text
goto Tools Project Preferences Help

ManageCar.java

```java
import java.io.*;
import java.util.*;

public class ManageCar{
// "throws Exception" is required in the function header below
// which will propagate all the runtime problems to the surface
public static void main(String[] args) throws Exception{
    int choiceInMain;
    do{
        // DISPLAY THE MAIN MENU
        System.out.println(
            "1. View car rental details"+'\n'+
            "2. Add new car rental record"+'\n'+
            "3. Borrow a car"+'\n'+
            "4. Modify car rental details"+'\n'+
            "5. Delete car rental details"+'\n'+
            "6. Search car rental details"+'\n'+
            "7. List available cars for rent"+'\n'+
            "8. List all borrowed cars"+'\n'+
            "9. Exit"+'\n'+
            "Please enter the service number you would like to use (from 1 to 9)."
        );

        Scanner scanMenu = new Scanner(System.in);
        choiceInMain = scanMenu.nextInt();
        switch(choiceInMain){
            case 1: View.viewAll(); break;
            case 2:
```

ne 16, Column 20 Tab Size: 4 Java

```java
            CarReadWrite.addCar(); CarReadWrite.readInput(); CarReadWrite.writeOutput(); break;
            case 3: Borrow.borrow(); break;
            case 4: Modify.modify(); break;
            case 5: Modify.delete(); break;
            case 6: Search.search(); break;
            case 7: Search.available(); break;
            case 8: Search.borrowed(); break;
            case 9: break;

            default: System.out.println("Please enter a valid number.");
        } //end of switch of main menu, every break statement in other class goes to here

    } while(choiceInMain!=9);

}//End of main method


}//End of ManageCar class
```

e 16, Column 20 Tab Size: 4 Java

3.2 View Car Rental Details

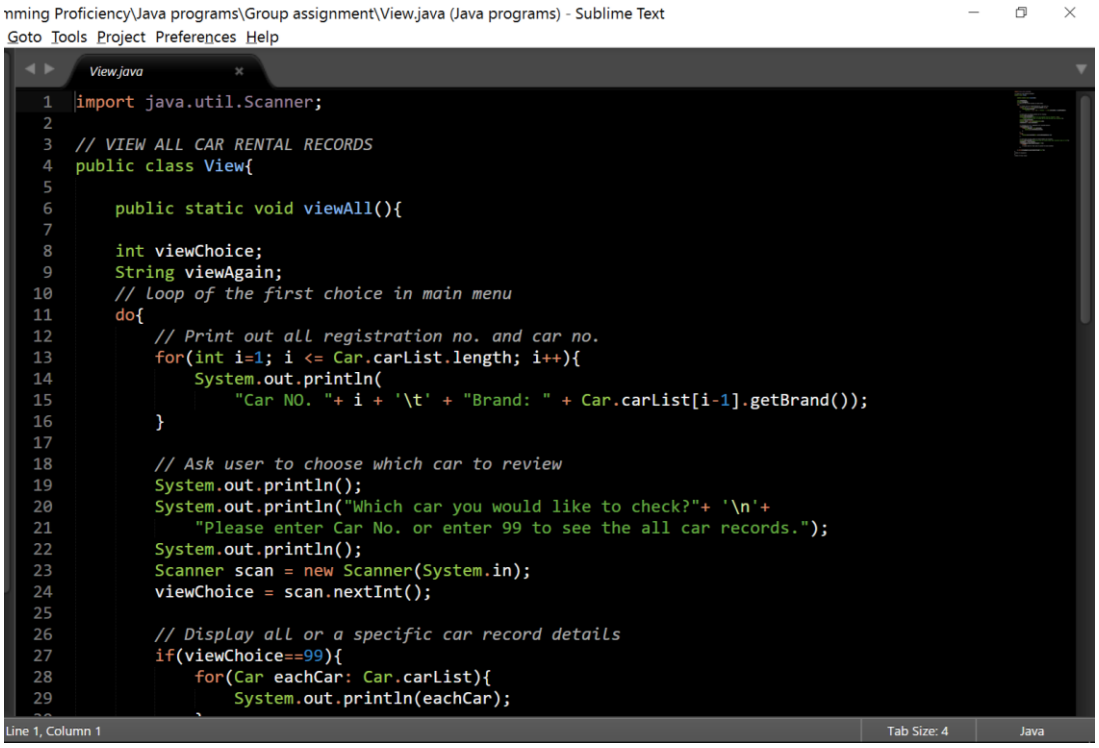
Once the user starts to run this car rental system, firstly the system displays the main menu and asks user's choice by number. If user enter 1, it calls View class and shows a list of cars and asks the user to choose a car to see details.

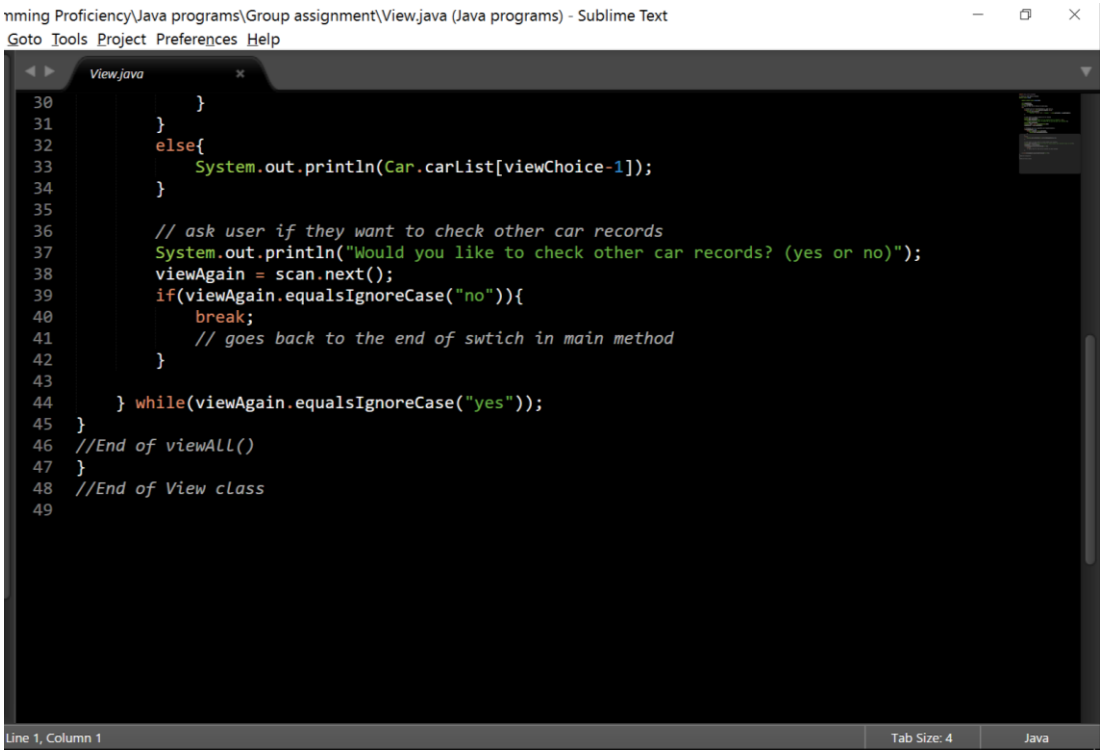
For example, if the user enter 2, the detailed car rental record of Car No. 2 will be display. Then the system will ask if user wants to see another car records.

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).
```

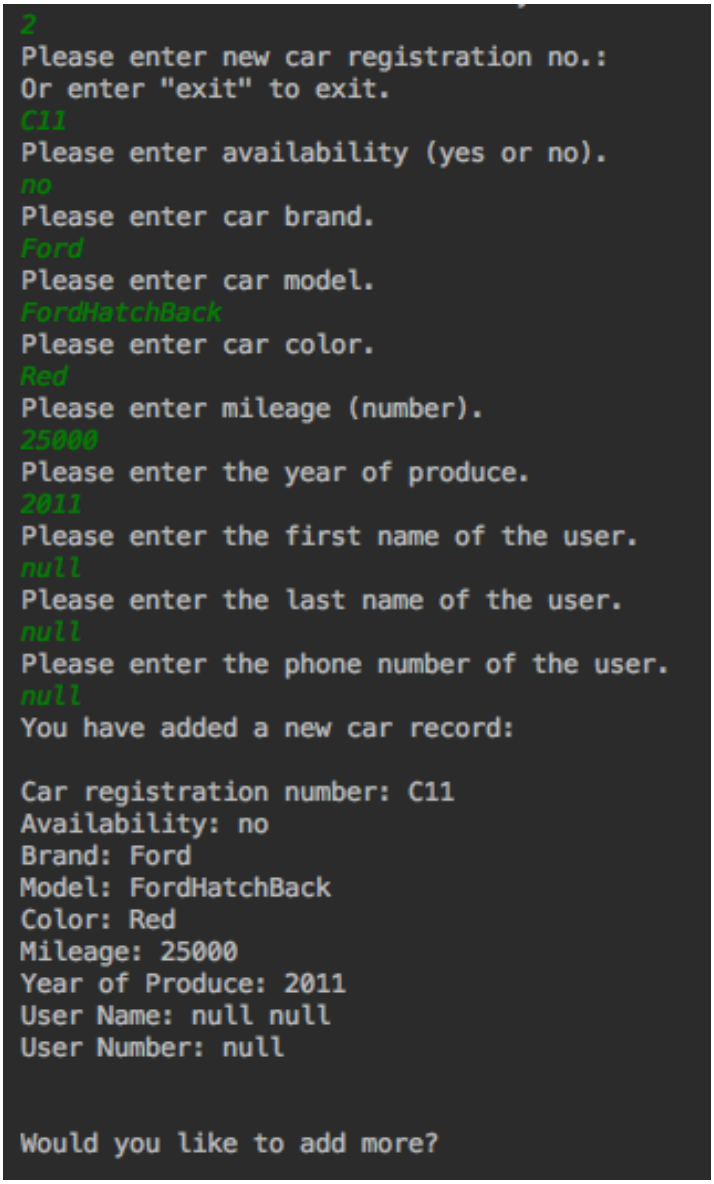
Source code of viewAll method in View class is shown as below. In viewAll method, the system starts with a do-while loop, then enter a for loop to display all registration no. and car no., ask user to choose which car to review, then display all or a specific car record details according to the user's 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.





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.

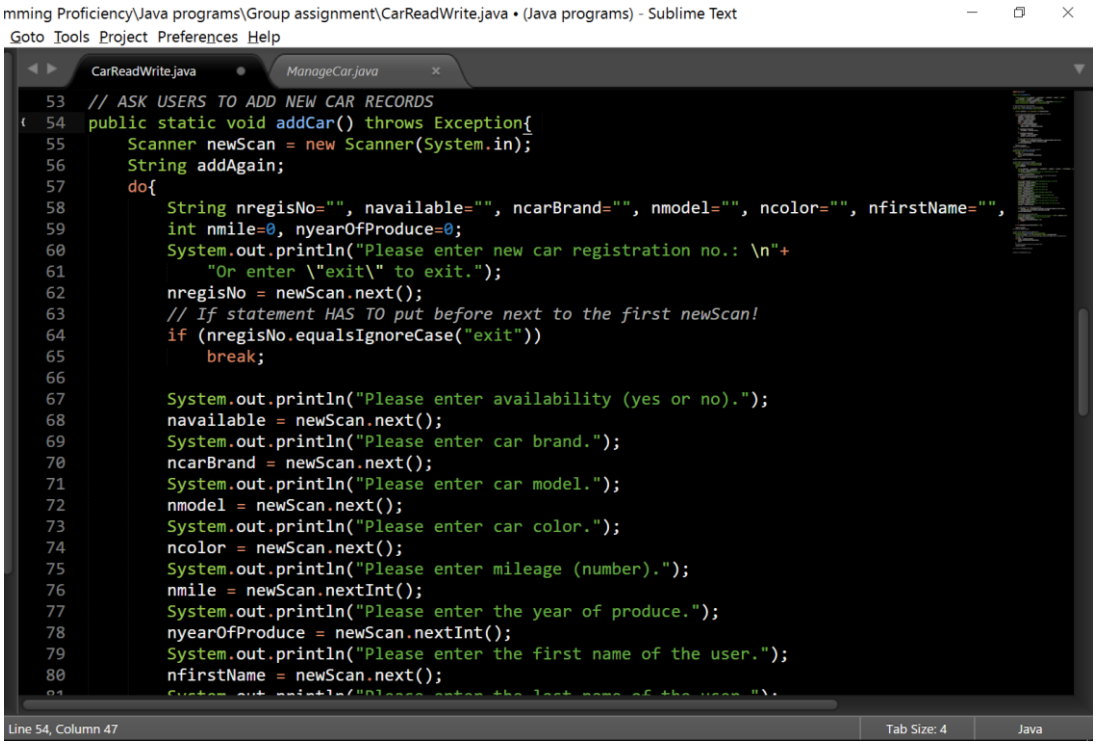


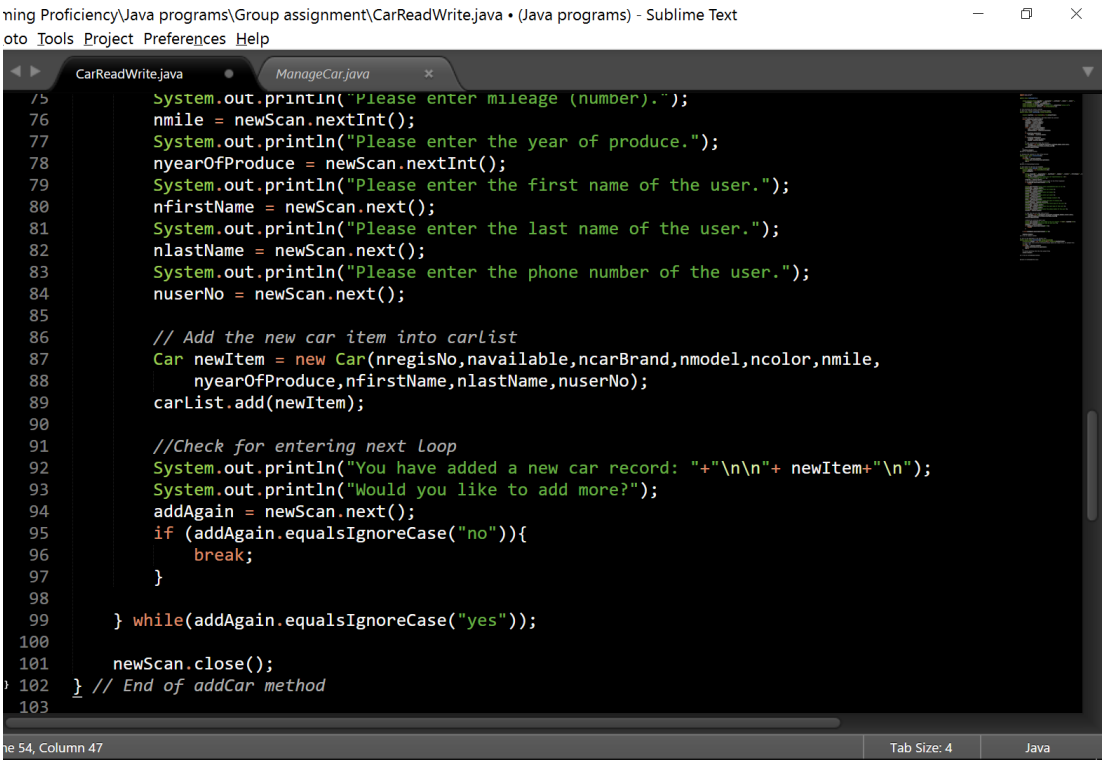
After adding the requested fields, the system will confirm the correctness of this new record. At his moment, the administrator or developer can find the added new car record inside the output text file.

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

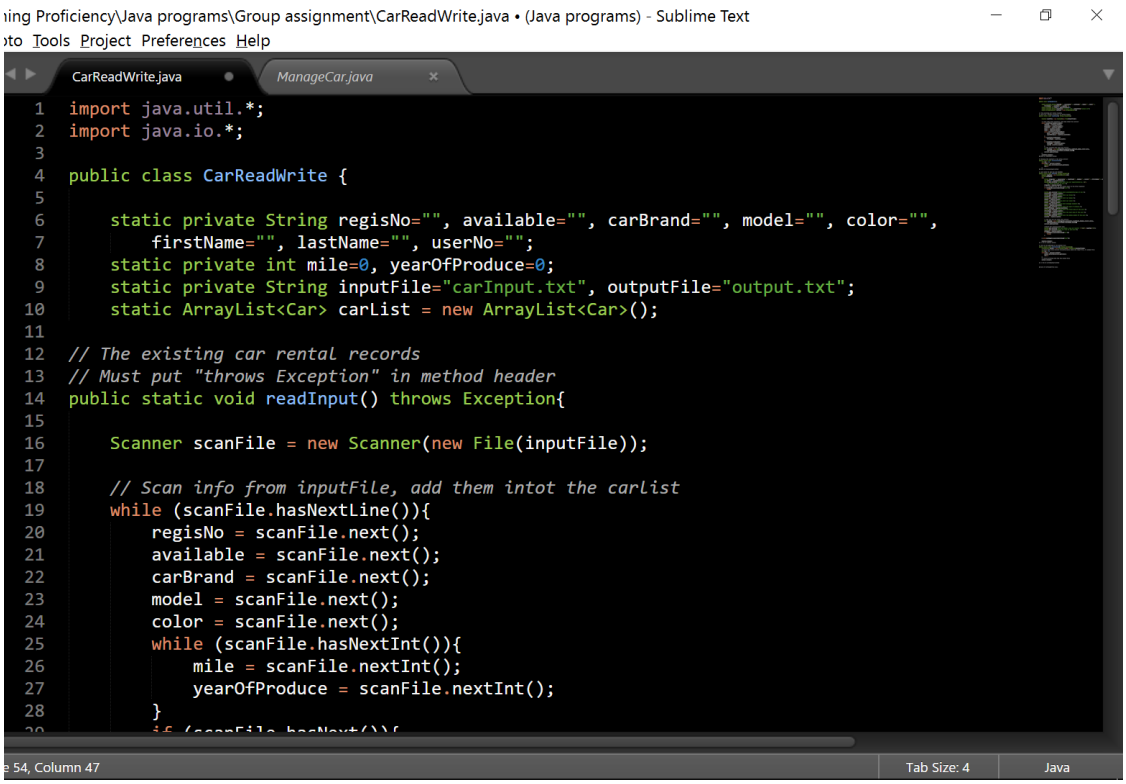
text file all together. Source code of addCar method, readInput method, writeOutput method are shown as below.

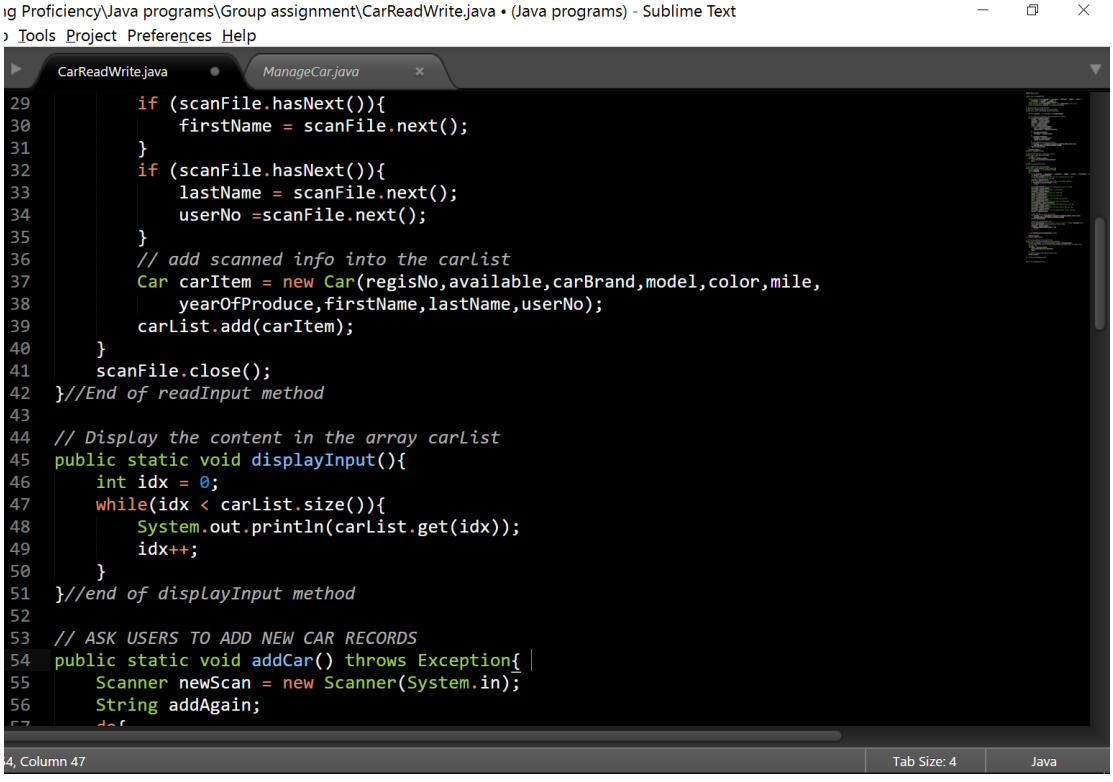
addCar method:



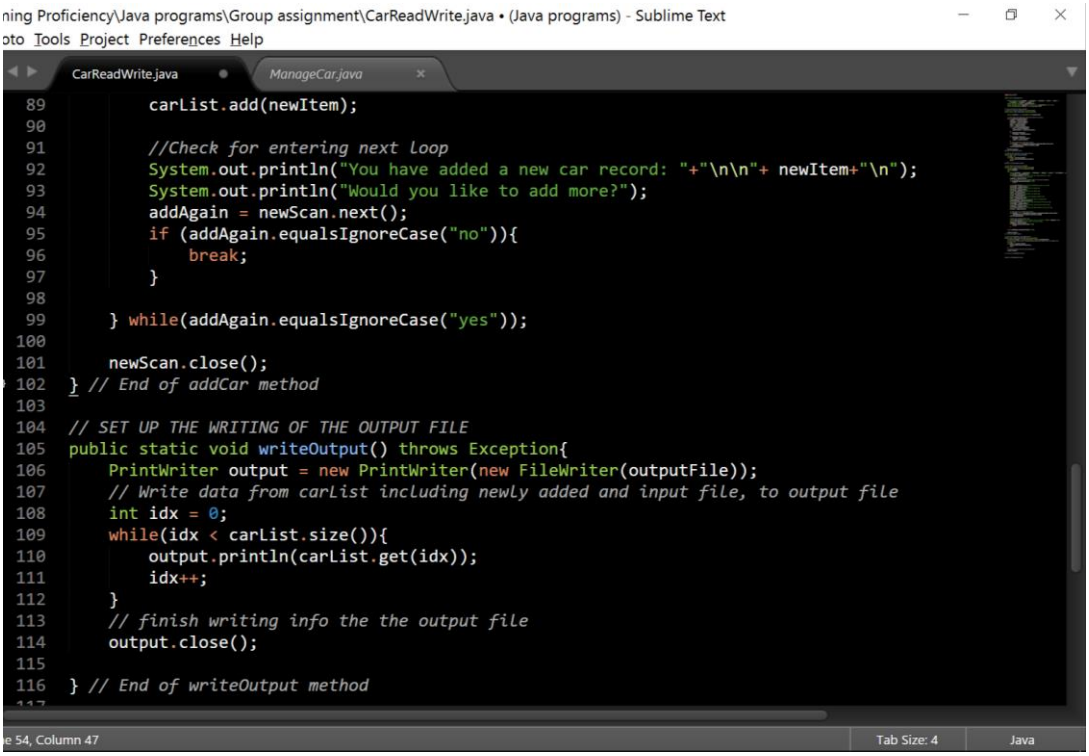


readInput method:



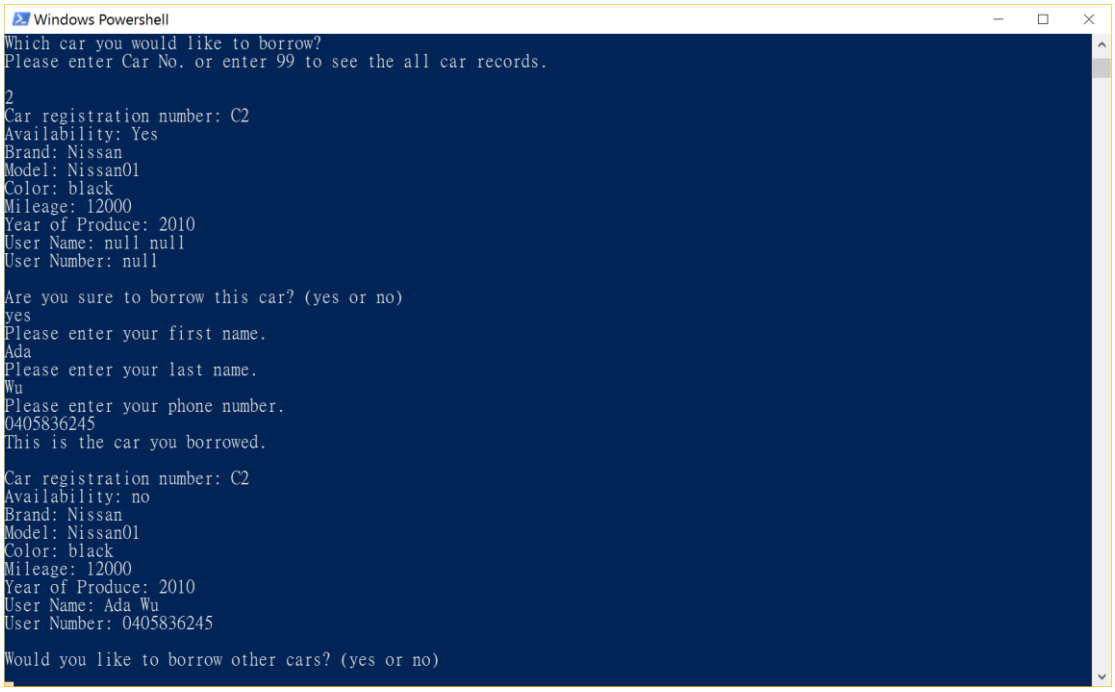


writeOutput method:



3.4 Borrow a Car

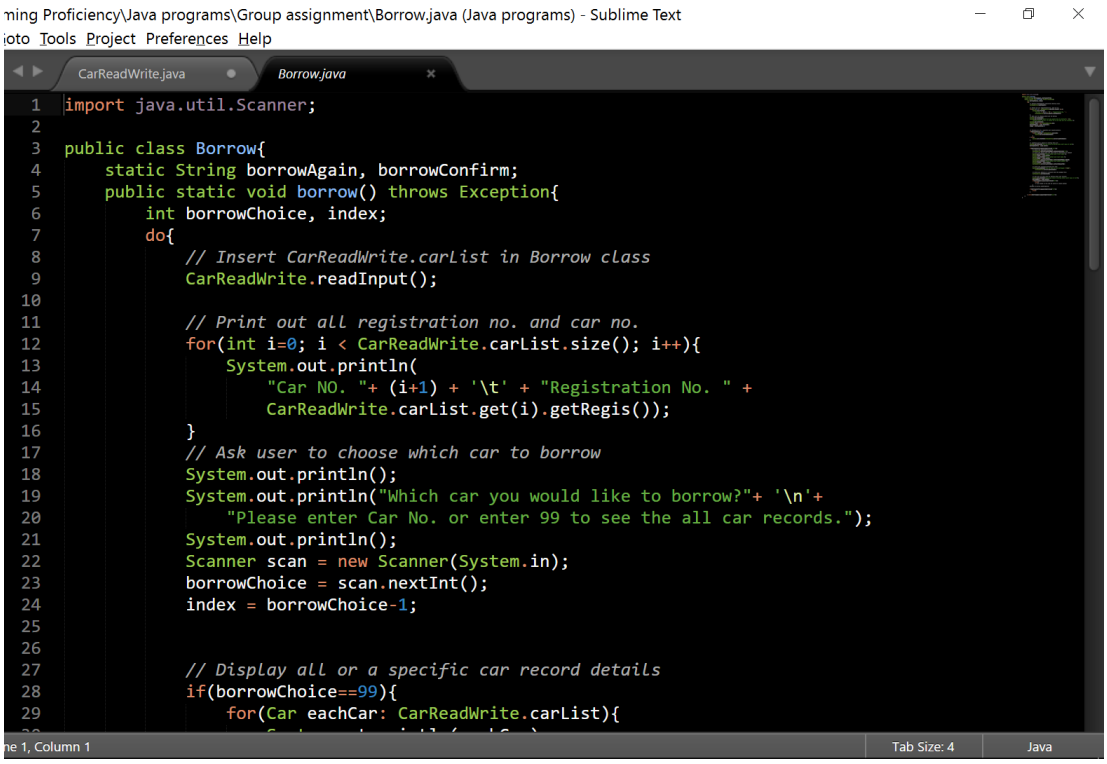
To borrow a car, the user needs to choose a car to borrow, and the program will prompt confirmation message to confirm whether the user wants to rent or not. If yes, the user can add the user name and phone number to borrow it. Once a car is borrowed, in the database, its availability status changes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

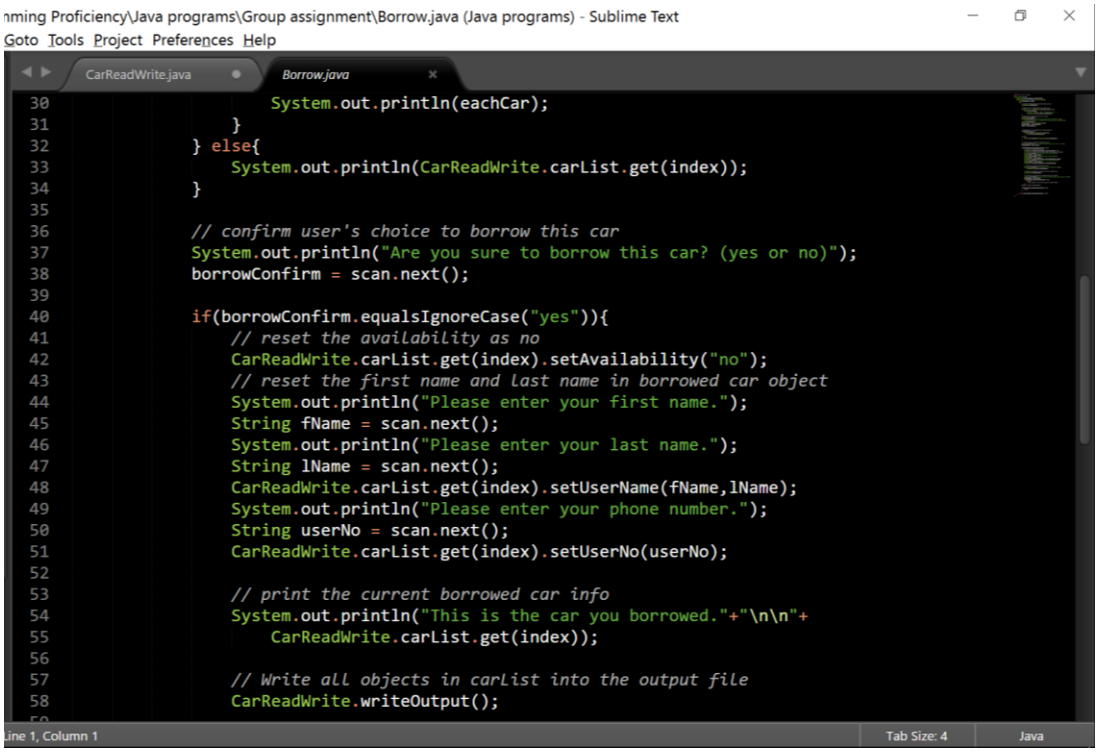


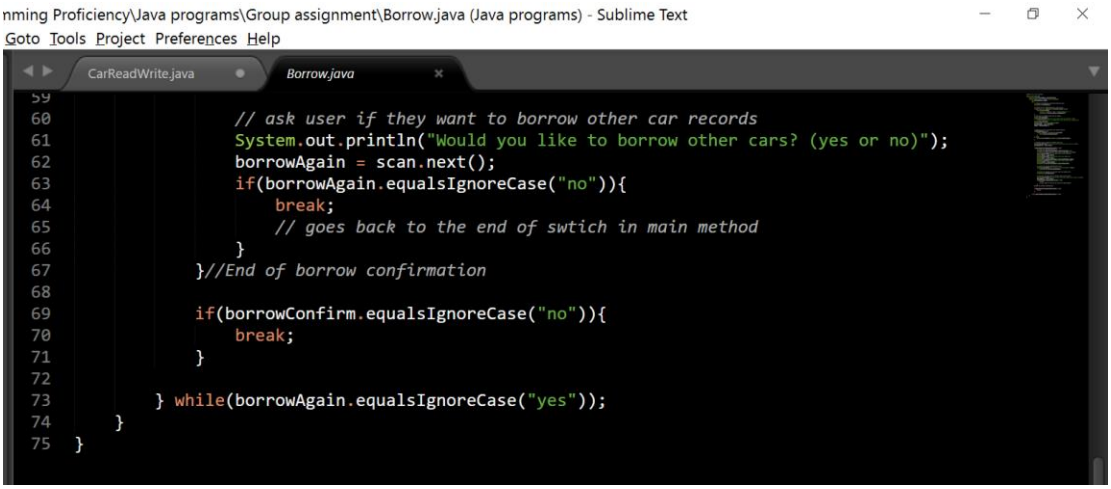
Regarding the backend part, first we insert carList from CarReadWrite class into Borrow class, then display all registration no. and car no. to the user by using for loop. Then we ask the user to choose a car to borrow, scan user's choice, and display the detailed car rental information to the user to 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.

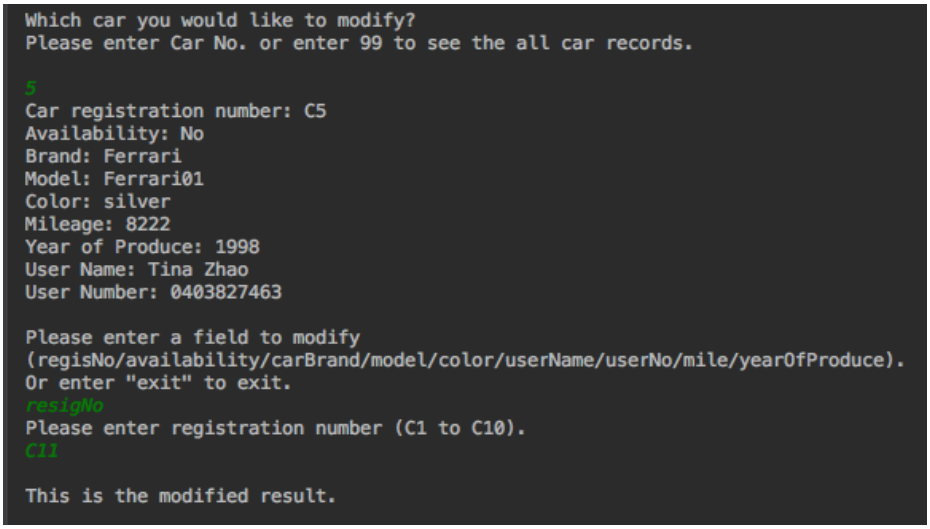


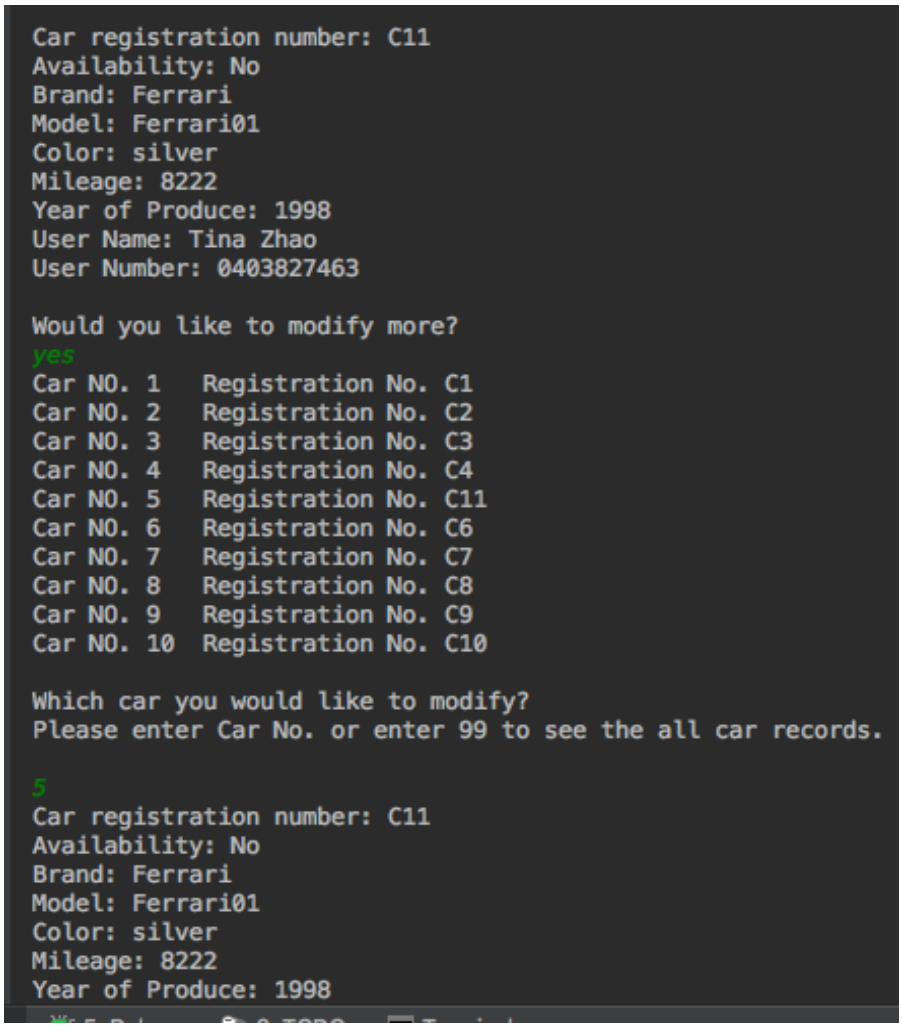




3.5 Modify Car Rental Details

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 filed 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.

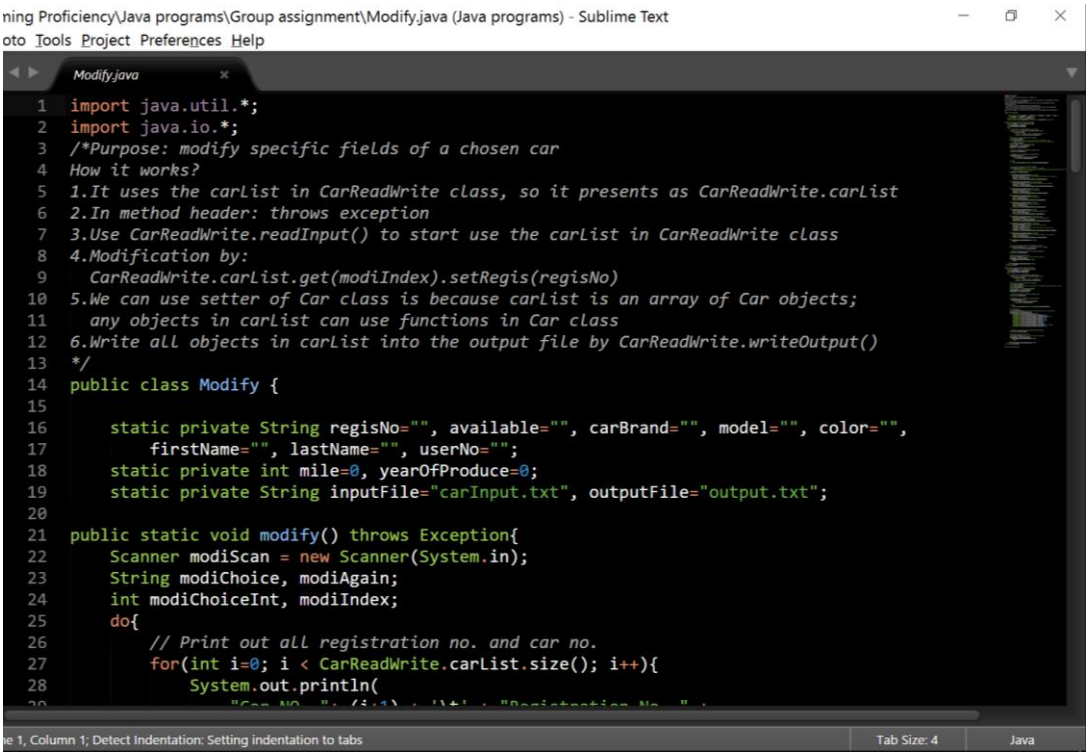


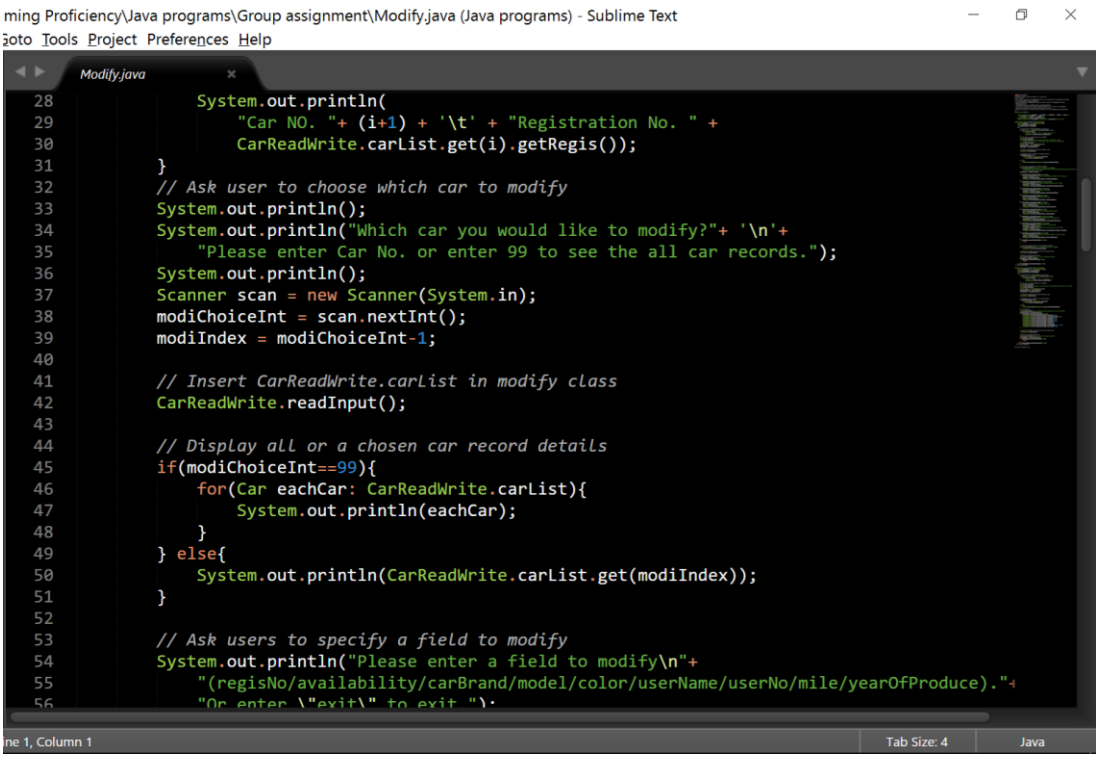


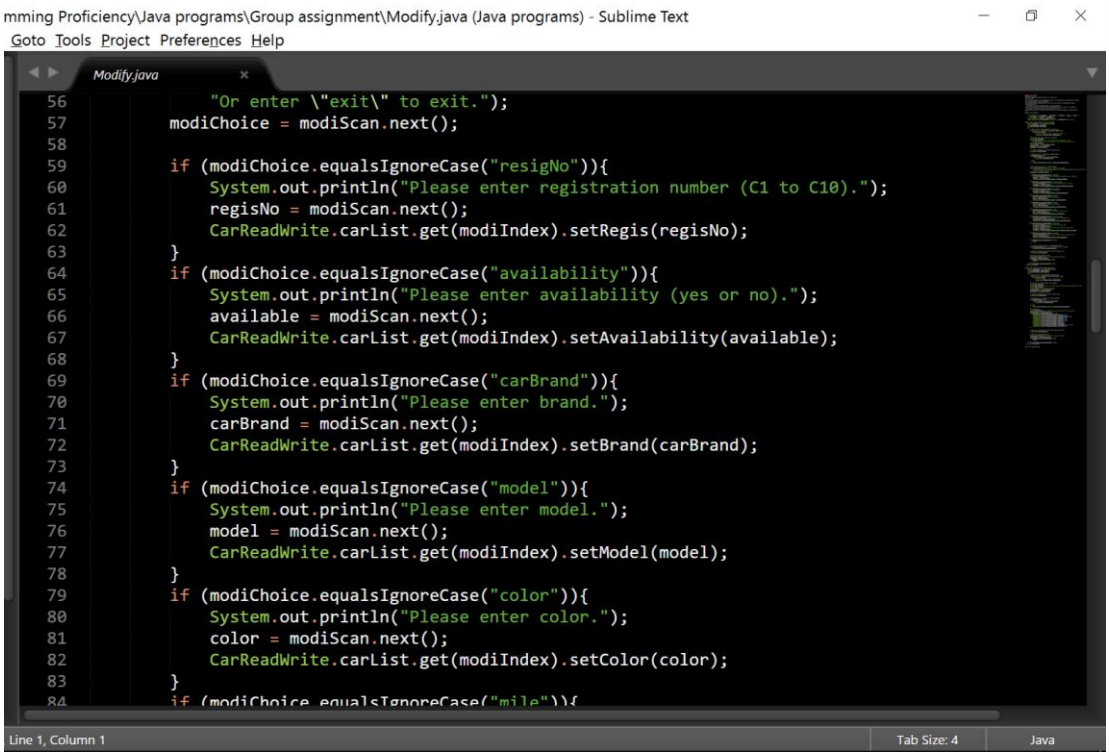
For the backend technology of the modify method in Modify class, there are a few basic steps for design. Firstly, the system uses the carList in CarReadWrite class, so it presents as CarReadWrite.carList. And although it uses static variable from CarReadWrite class, it still need to declare *throws exception* 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

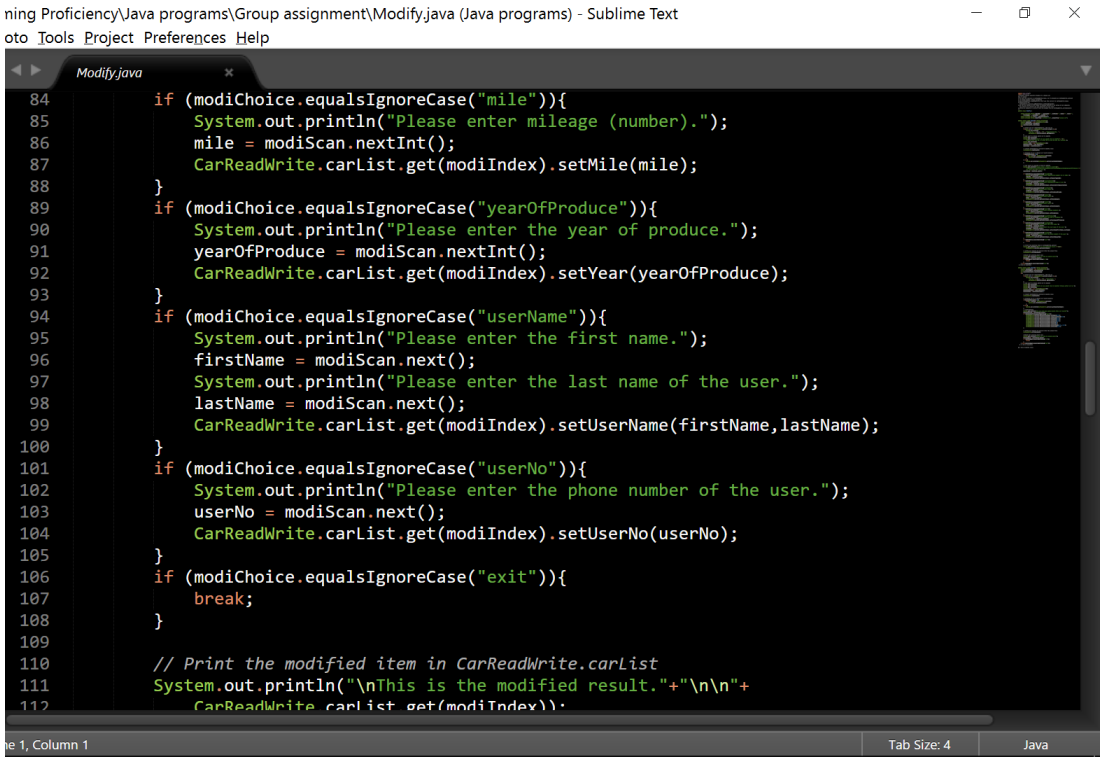
CarReadWrite.carList.get(modiIndex).setRegis(regisNo). Fourth, we can use setter of Car class is because carList is an array of Car objects, and any objects in carList can use functions in Car class. Fifth, write all objects in carList into the output file by CarReadWrite.writeOutput().

Source code of modify method in Modify class:









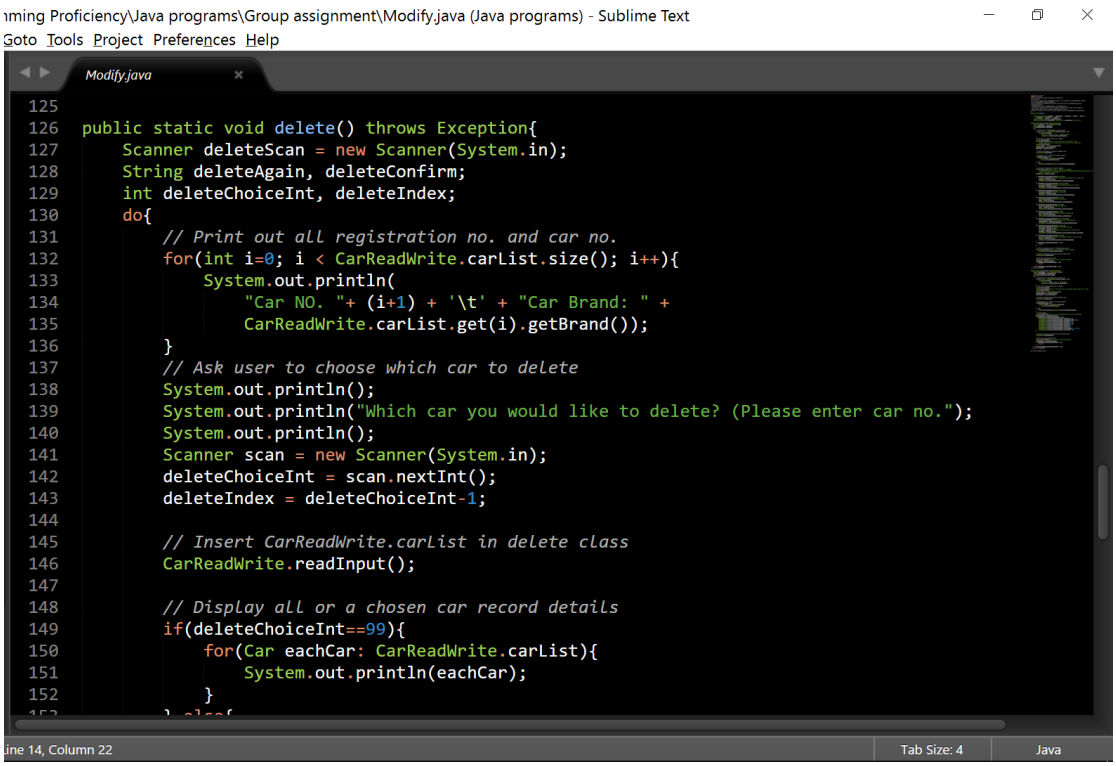

```
            CarReadWrite.carList.get(modiIndex));

        // Write all objects in carList into the output file
        CarReadWrite.writeOutput();

        //Check for entering next Loop
        System.out.println("Would you like to modify more?");
        modiAgain = modiScan.next();
        if (modiAgain.equalsIgnoreCase("no")){
            break;
        }
    } while(modiAgain.equalsIgnoreCase("yes"));
} //End of modify()
```

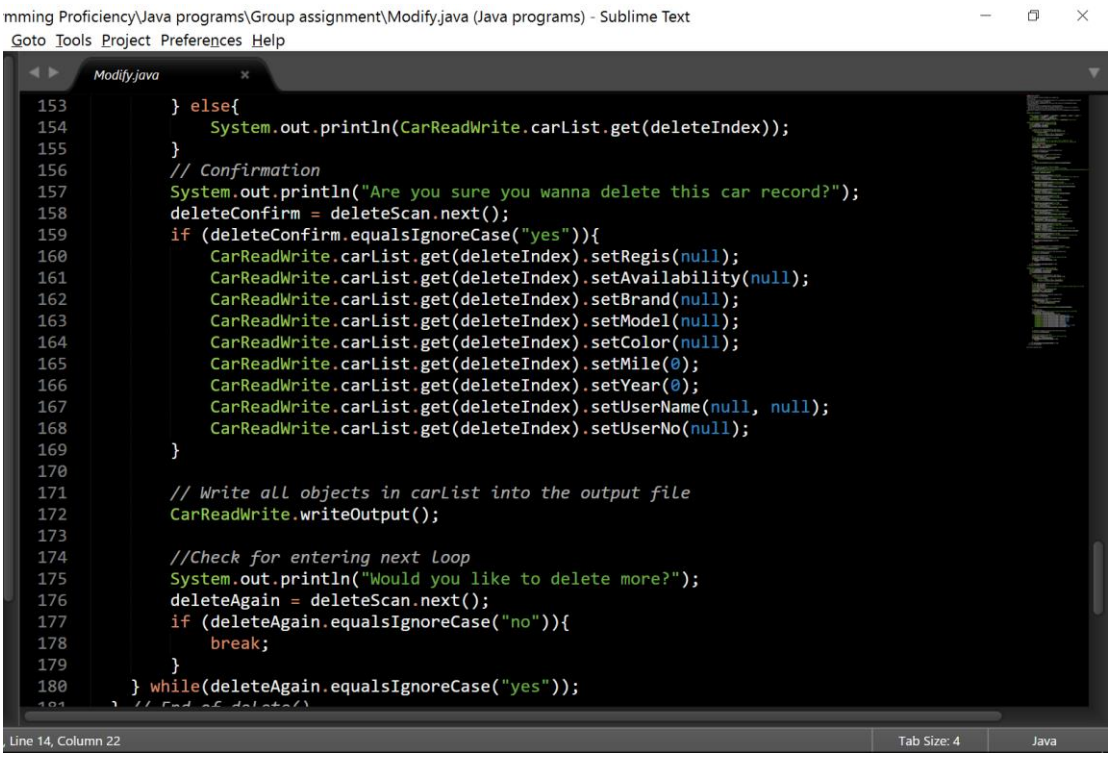
3.6 Delete Car Rental Details

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?
```

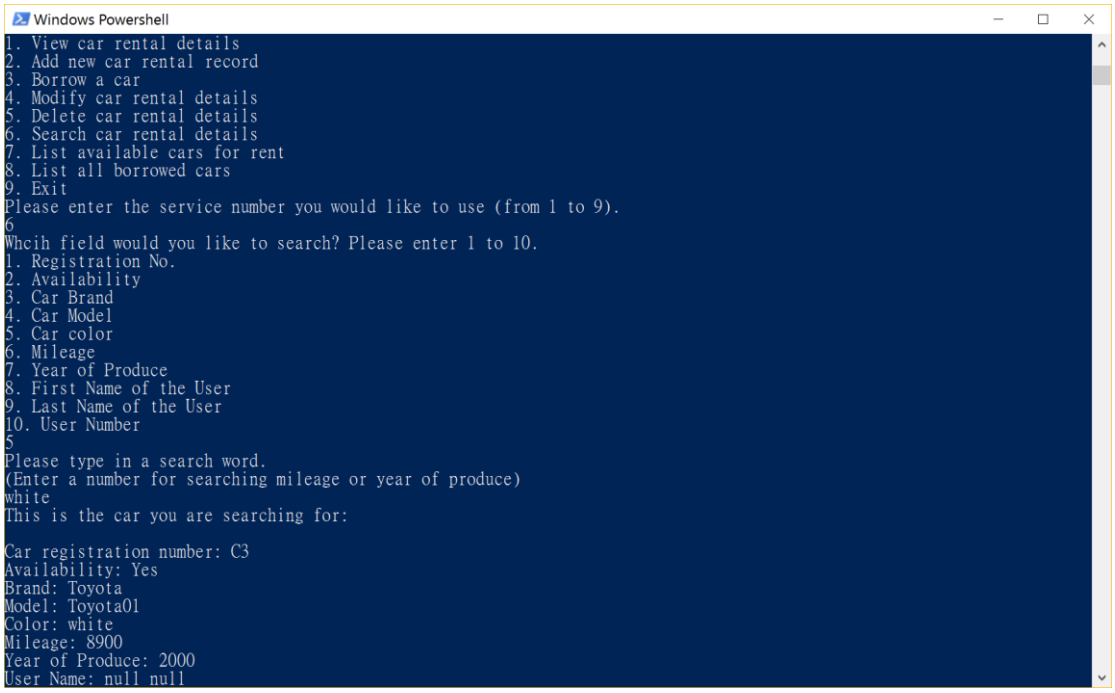
The delete method is inside the Modify class, and the only difference between it and the modify method is that the delete method sets all fields in a car record into null. In this way, it implies the deletion of a car rental record. Here is the source code of the delete method in Modify class.

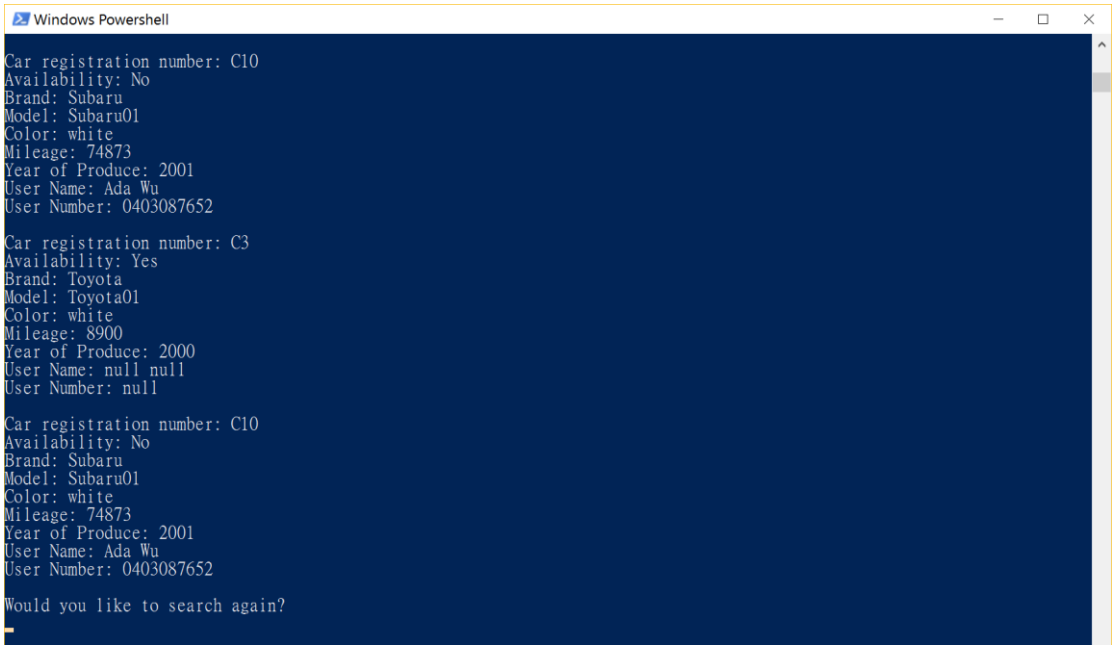




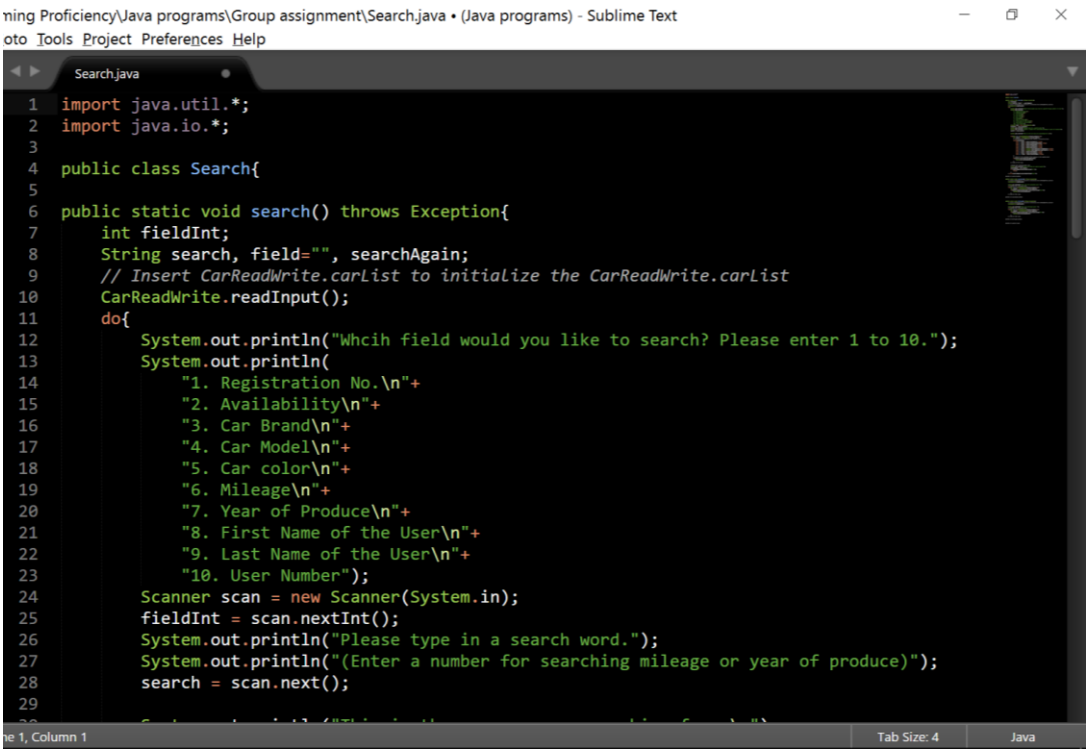
3.7 Search car rental details

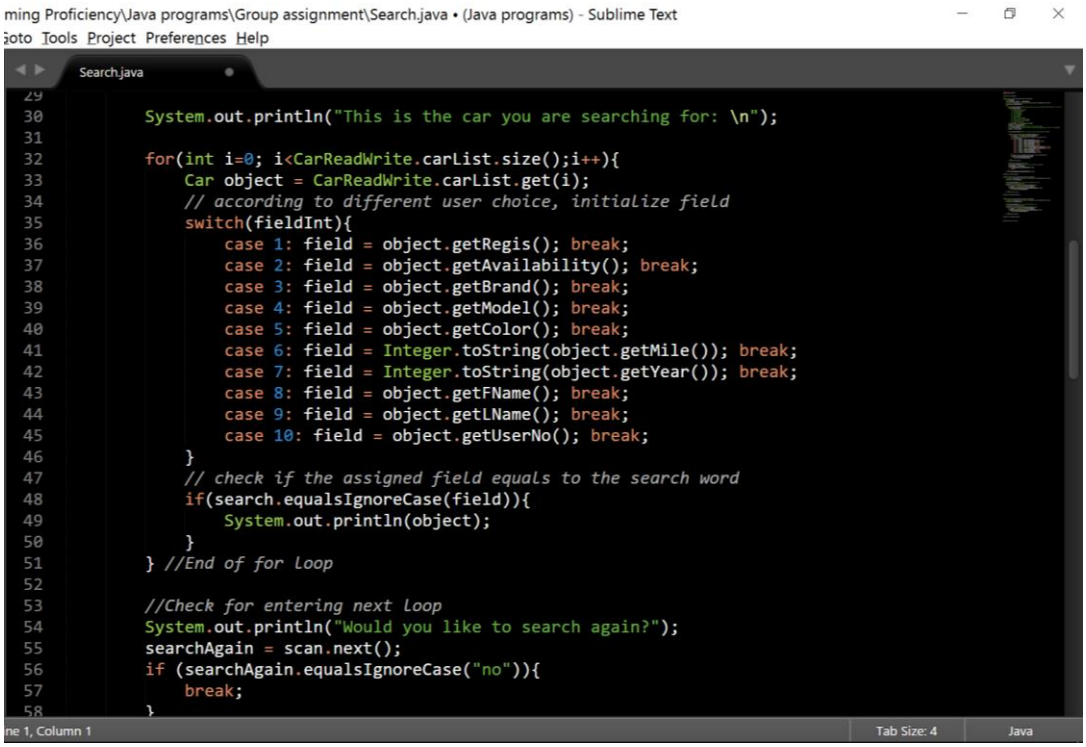
User can search cars by providing a search word for a field, such as registration number, availability, car brand, car color, year of produce, etc. Once the user decide which field to search, the system will ask a search word from the user, then display the search result. In the end, the user can decide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

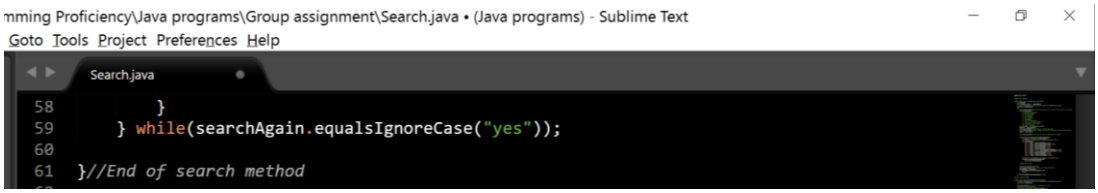




For the backend technology, Search class is designed highly relying on the CarReadWrite class, which is used for accessing the database. During execution, the Search class firstly reads all car rental records from database via the function readInput in CarReadWrite class. According to the user's choice 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.

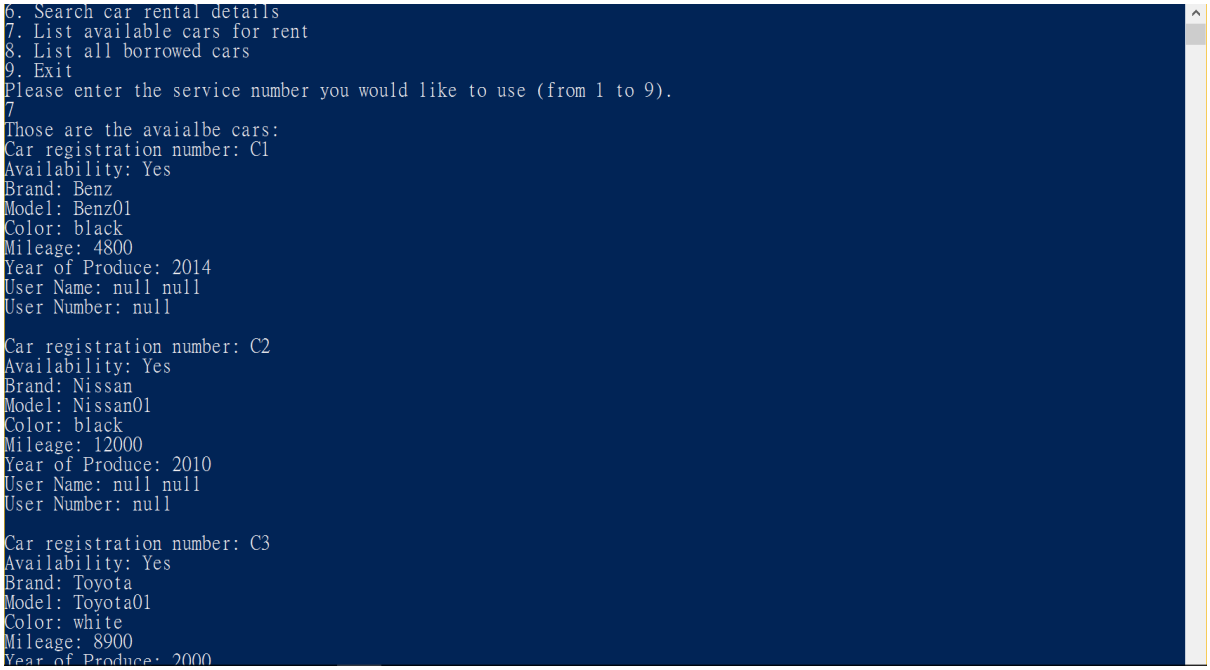






3.8 List Available Cars for Rent

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



```
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
```

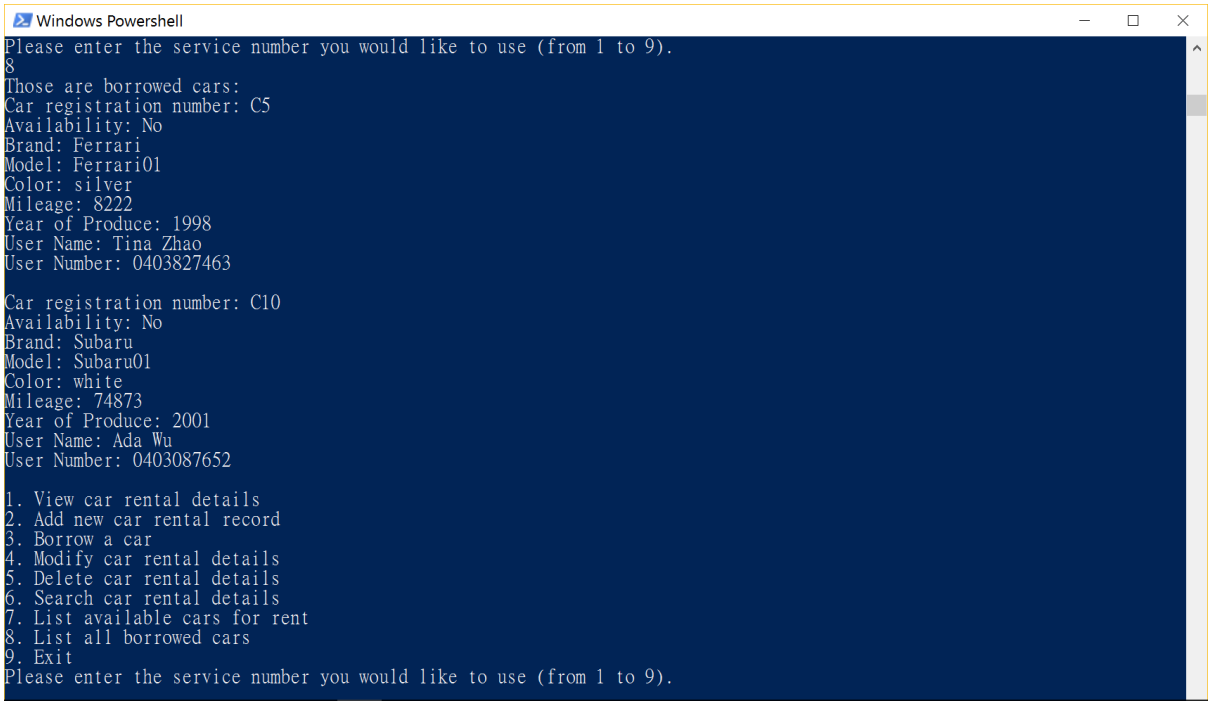
```
Car registration number: C9
Availability: Yes
Brand: Peugeot
Model: Peugeot01
Color: grey
Mileage: 130229
Year of Produce: 1999
User Name: null null
User Number: null

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

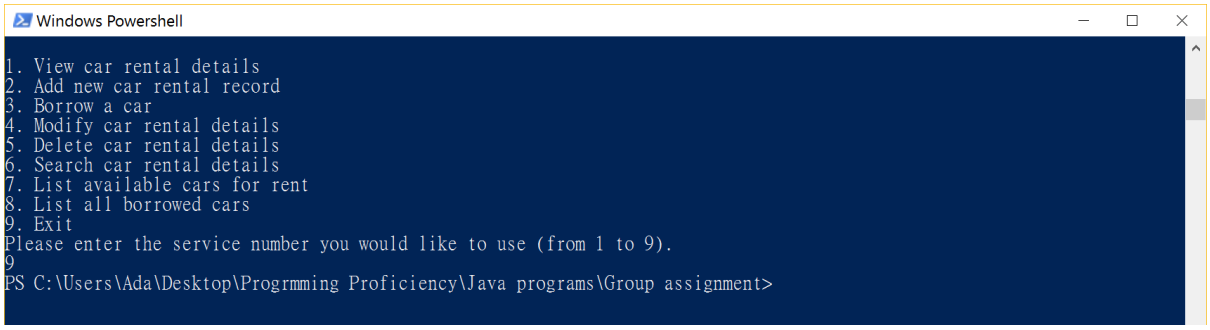
Similar with list all available car, the only difference between the borrowed function and the available function is that the system will examine which car's availability field equals to no and display them to the user. For loop is used to visit all car records/objects in the database.




```
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.



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;

 Constractor, 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:

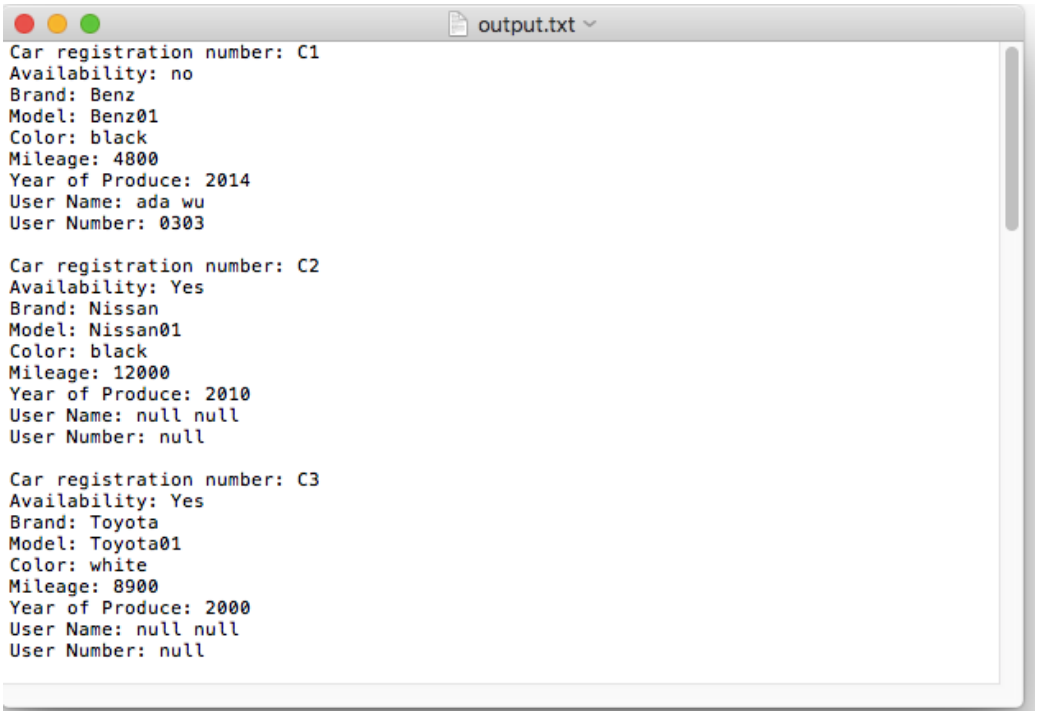
Interface

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.text files contain car rental records.



Array List

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 interfaced is used that helped to provide outline what class should use. Java object oriented principle is also used that helped to divided 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 exemption handling principled are used in this project, in order to hide the variables and throws exemption. 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 filed 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 in 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.