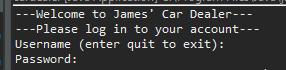
Object Oriented Programming Coursework Documentation

# How to use the User Interface

## Logging In

There was an error with the way using encryption, I am using SHA-256 using the message digest package, when creating ‘admin1’ with password ‘apple’ it makes a newline character, causing the password to not be found fully in the file, thus making it not work. I’ve had an error I don’t know how to fix with the ‘customer1’ with password ‘lemon’ that causes it not to give the same result as straight encrypting the string vs getting the encrypted string from a file, I don’t know what is causing this. I have set up three more accounts with passwords that work shown below.

First you must enter your username and password:



For ease of testing, 3 accounts have been set up, one for each role.

Username: kate

Password: disco

Role: Admin

Username: james

Password: jaffacake

Role: Staff

Username: jim

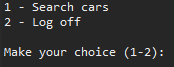
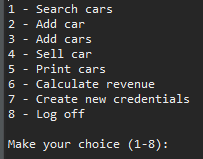
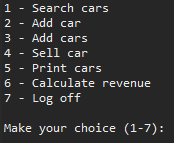
Password: disco

Role: Customer

Username: exit

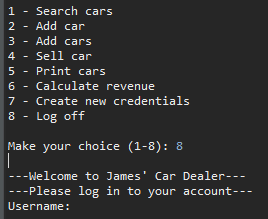
(this will exit the system)

Once logged in you can select a choice dependant on what role you logged in as:



Staff Admin Customer

Logging off will allow you to log back into the system with either the same or a different user.



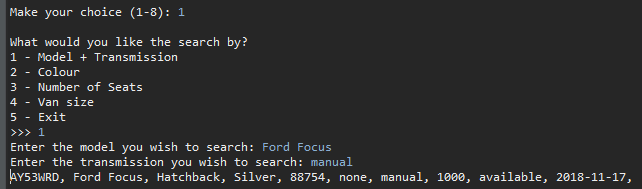
## 1 - Searching for Cars

When searching for cars, you enter 1 on any of the menus. This then takes you to another menu which you can select which type of search you want to complete. I will show the admin/staff searching, the only difference is when you search with a customer account, you will not get the accident history of the car.

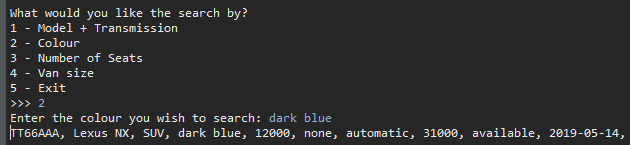
(all examples have had the cars-import.txt already into, to have more results. Assuming you are testing this, the database won’t have them already put into it, so you may get different results.)

You just need to input the number dependant on what you wish to search by, if the number is not in the range it will ask you again. All results that are given are of cars that are **available** in the system, no sold cars are shown.

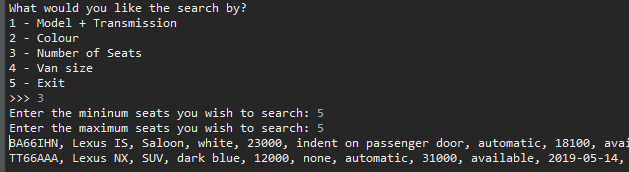
Searching by model + transmission gives you two inputs, Model can be anything, but transmission must be manual or automatic, it will not accept anything else.



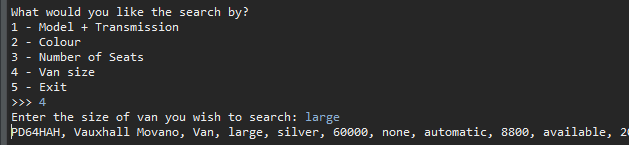
Searching by colour, lets you input any colour into the system, and returns cars that fit.



Searching by number of seats lets you put in a max a minimum number and displays all cars that have seats numbers in that range.



Searching by van size will let you input the size and show all vans that have that size.

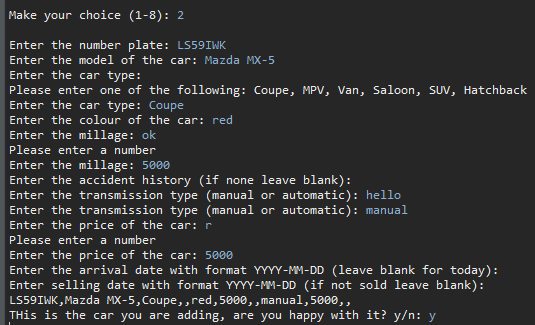


## 2 – Adding cars manually

To add a car, simple log in as staff or admin and choose option 2 on the main menu, you will then be given a bunch or inputs to specify the car further, such as number plate, model, car type, etc.

All, except number plate, model, colour and accident history have validation to make sure they are correct format to put into the database.

At the end it will ask if you are happy with it, also showing what you entered. If you are not happy and have made a mistake, say ‘n’. If you are not happy you are given the chance to not have to do it again, in case you mis-inputted to add a car. It will also check to see if the car is already in the system, giving a message if the number plate has already been entered before.



## 3 – Adding cars from ‘car-import.txt’

To add cars from the, firstly add them into the ‘cars-import.txt’ with the correct format:

numberPlate,model,carType,size(van),colour,millage,accidentHist,transmission,price,dateOfArrival,dateOfSelling

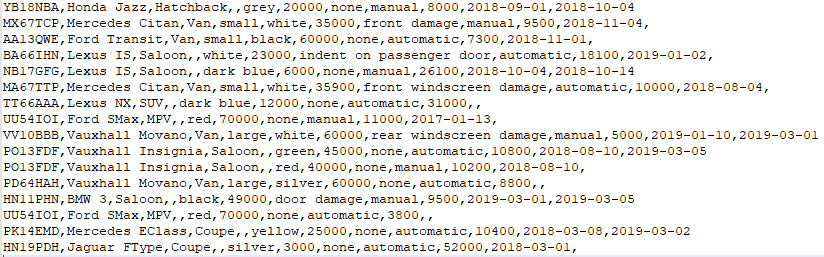
-**Always make sure there is the same number of commas in the correct places, even if there is nothing to put into the slot (i.e no size = ,,).**

-The availability is correctly given depending if the date of sale has been found in the record.

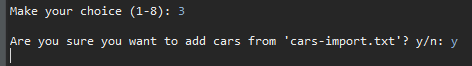
-If the date of arrival has not been given, then it is assumed that it came today.

-If there is no accident history put ‘none’.

-If the car isn’t a van, leave size as “”.



When using it from the interface, it will give a prompt to ask whether you are sure you want to add, just in case you haven’t doubled checked the file.



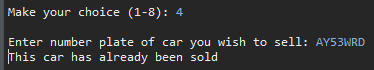
## 4 – Selling a car

To sell a car, you first log in as staff or admin, and select option 4 – Sell car.

You are then prompted to enter a number plate of the required car you wish to add.

If the car has already been sold it will tell you so, and not change anything. If the car hasn’t been sold, it will change the line in the database to have sold and put the selling date to the current date.



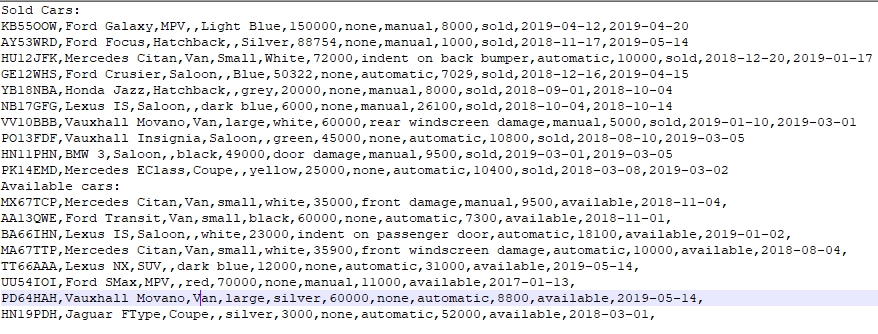


## 5 – Print cars to ‘car-output.txt’

To print all cars into the ‘car-output.txt’ you first log into a staff/admin account, then select option 5.

The cars will be sorted into ‘cars-output.txt’ into sold and available. You will get prompted to make sure you wanted to print them to the file.





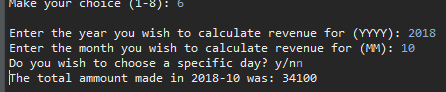
## 6 – Calculate revenue for certain year-month or year-month-day

To calculate the revenue, log in as staff/admin, then choose option 6.

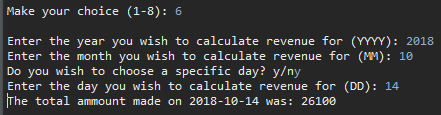
You get prompted to enter the year and month, if they are not within the range of values, then you will be prompted again.

You will then get prompted if you want to choose a specific day in the year and month you have already entered. If not, then that month’s revenue will get shown. If you choose a specific day, then that day’s revenue will get shown.

Year + Month



Year + Month + Day



## 7 – Create new credentials for users

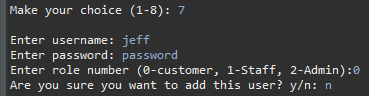
Creating new users is something an admin can do, so you must log in as an admin.

Once logged in you select option 7. You get prompted to enter a username, password and what role they can have. To stop the program getting confused, you cant have a comma ( , ) in your username, as it would break the csv file.

When you have entered their info, you get prompted if you’re sure you want to add them into the system.

If their username is already taken, then you will get prompted to say that that user has already been taken and you must enter a different username.

Due to some problems with the way I’m getting the file and weird symbols, some passwords that are encrypted don’t work, and even thought they will be the same as the ones in the file they won’t create a true statement when checked against each other.



# Packages used

- java.security.MessageDigest

Used for the encryption algorithm, SHA-256

- java.util.regex.\*

Used to verify patterns in strings, useful to make sure date is the correct format

- java.util.Scanner

Used to get user input from the console.

- java.security.NoSuchAlgorithmException

Had to be used in case the algorithm in the message digest wasn’t correct, even though it doesn’t change, removes an error in the compilation.

- java.util.\*

Used to get the List<> functionality,

- java.io.\*

Allows to throw any IOException and for the way I am writing and reading to files using BufferedWriter/Reader, allows reading of records at a time.

- java.time.LocalDate;

Gets the current date on your machine, for use in the selling date and if the car added does not have a arrival date.

# Functionality not Included

-I did not implement classes for the vehicles, this was because the way I handled them was to use a list of Array[12], I had already written the code for many of the functions and I would’ve had to rewrite many of them to fit with the new system of vehicles as objects. Due to this the code is not as object oriented as it could be.

-I didn’t add validation on the password so that any password that was added by an admin for a user didn’t break the system I am using to get the file from into a list of arrays. This would make sure that any password that was added could be used and stored safely without worry that the password wouldn’t work, which is the case at the moment.

-Functionality so that it would verify the cars-import.txt when adding multiple cars, this means that if you don’t have the correct format it will get confused and may cause an error.

-You can also only import from that one file, you cannot select a different one.

# Class Diagrams

|  |
| --- |
| **carDealer** |
|  |
| +main(Sting[] ags)  -showUI(Object user, String role, Scanner a)  -searchUI(Object user, Scanner a)  -addCarUI(Object user, Scanner a)  -addCarsUI(Object user, Scanner a)  -sellCarUI(Object user, Scanner a)  -printCarsUI(Object user, Scanner a)  -calcRevUI(Object user, Scanner a)  -newCredUI(Object user, Scanner a)  -Login(String user, String pass)  -validate(Object s, Object validationObj)  -validate(Object s, Object[] validationObjs |

|  |
| --- |
| **role (base customer)** |
| +String username  +Int role |
| +role(String user)  +searchCar(String model, String trans)  +searchColour(String colour)  +searchSeats(int minSeats, int maxSeats)  +searchVans(String size)  -printList(List<String[]> x)  -typeToSeats(String type) |

|  |
| --- |
| **Staff** inherits role |
| +String username  +int role |
| +Staff(String user)  +addCars()  +addCar(String numberPlate, String model, String carType, String size, String solour, String millage, String accHist, String transmission, String price, String arrivalDate, String sellingDate)  +sellCar(String numberPlate)  +printCars()  +calcRevenue(int month, int year)  +calcRevenue(int month, int year, int day) |

|  |
| --- |
| **Admin** inherits Staff |
| +String username  +int role |
| +Admin(String user)  +createNewCredentials(String username, String pass, int role) |

|  |
| --- |
| **ReadFile** |
| **-**List<String[]> csvReadFromFile  -BufferedReader fileReader |
| +ReadFile(String file)  -read(BufferedReader)  +getFile()  +searchUser(String user) |

|  |
| --- |
| **WriteFile** |
| -BufferedWriter writer  -ReadFile reader |
| +WriteFile(String file)  +append(String str)  +close()  +write(String[] str)  +changeLine(String[] newline, int index) |

# Object Oriented Design

Each time a user is created, a new user object is created, this then is used whenever the user wants to do one of the functions. As you cannot pass different types of objects into the same parameter, I utilized the casting of another object onto the already passed object. For example, I can have one printList function, but have both Admin and Staff use it through casting.

Also, each time I want to read or write to a file, a new object for that file is created and then that has methods allowing me to read, append, change a line in any text file.

# Testing

I have done unit testing throughout the development of the code, I didn’t and haven’t made a test plan, I just tested as a I went along. Everything I found I fixed, if there is bugs I apologize.