

Description

Above is a screenshot of the class diagram for this system. In this diagram the user uses the interface to call the various functions: Log in, Add Car(s), Sell Car, Search, Print Cars, Calculate Revenue and Create New User. The interface will also display the results of these functions back to the user.

The user database is composed of user objects which each have a username, password and role. The file will be loaded into the program only when needed as an ArrayList of user objects, this allows the system to append new user objects without writing directly to the text file. The password will be encrypted when in the database text file and it and the username will only be accessed when the user is created in the “Create New User” function and the log in function. The role will be stored as a global variable so that access to various parts of the system can be restricted when necessary.

There are 3 types of user objects. The admin type are the only users who will have total system access. The staff type users will have access to everything except the new user function; the customer type users will only have access to the search function in which they will not be able to see the accident history attribute.

Users will gain access to the system via the log in function which will error check the input and compare it to the user database. Only if the function finds a single user object which matches both the username and password will the user be successfully logged in and navigated to the appropriate page. The log in function will also need to use the encrypter class as the passwords will be encrypted when they are stored in the user database and can only be properly read when they are decrypted. The log in function will only access the user database it will not alter any of the objects within.

After logging in the user will be taken to the appropriate menu, admins will get the full main menu, staff will get a main menu without the create user function and customers will be taken directly to the search function. The reason for having separate menus (or no menu at all for customers) is so as not to confuse users by showing many functions that they can’t access. In addition, it also means that the only way for a user to access any of the other functions without the correct permission is to load the source files directly; the source files will require the user to be logged in to function and will therefore not run.

The create new user function is the only other function that will require encryption as it will append the new user object to the user database ArrayList and will then write the updated ArrayList back to the text file. Before creating the new object however, it will check that the user input is valid and does not contain any characters that will cause errors when reading the database file back into the system. It will also check that the username entered is unique, in that it will query the database for user objects whose username attribute matches that of the input.

The vehicle database will be composed of vehicle (car) objects. It will also be read into the system only when needed as an ArrayList as it will need to be altered and updated frequently. Unlike the user database however there is no need to encrypt any of the data in the text file.

Vehicle objects will be of 2 types (cars and vans) which will have the following attributes: registration number, model, body (object), colour, mileage, accident history, transmission, price, arrival date and sell date. Body objects will have the attributes body type, doors and seats; if the vehicle object is a van however then the body object will have the additional attribute size (small or large).

The add car (singular) function will take the user input and construct a new vehicle object which it will then append to the database ArrayList. This updated ArrayList will then be written to the database text file. The input will consist of registration number, model, body type, size (vans only), colour, mileage, accident history, transmission and price. The input will not include the arrival and sell dates as the arrival date will be set to the current date by the program and the sell date will be left empty to set the entered car as unsold. It will also error check the user input and query the database to see if there are already any vehicle objects that match the registration number input, if there are then an error message will be displayed and the database will not be altered. The function will also check if there are any characters in the input that will cause errors when reading the database file back into the system.

The add cars function will get the vehicle database as an ArrayList and will take the “cars-import” file as the input. The file will be error checked to make sure that each row has the correct number of fields and does not contain any invalid inputs. It will also check to see if there are any repeated registration numbers within the file and will check the vehicle database to see if any of the registrations the user is trying to import already exist. If the file passes the error check then as each line is read it will be used as the input for the vehicle object constructor and then appended to the database ArrayList. When all the entries have been read in from the file the database ArrayList will be written to the database text file.

The sell car function will take a single registration number as input. The input will be error checked and if it passes the function will then get the vehicle database and search it to find a car with a matching registration. If there are no matching objects, then the user will be informed, and the database will not be altered. In addition, if an object with a matching registration is found it will first be checked to see if the sell date field is empty; if it is not then the car has already been sold and the user will be informed. Once an object with a matching registration is found the sell date field will be set to the current date and the database file will be updated.

The search function will be able to take a variety of user inputs to be used as filters. It will begin by getting the vehicle database as an ArrayList, it will then filter out all of the sold cars. The remaining cars will be filtered out depending on the filters the user has input. Some of the filters will be error checked (the filters that require a numerical input) but the others will simply be left to the user’s common sense. There will also be a feature that allows the user to sort the search results by a number of fields. The output of this function will vary depending on the type of user accessing it; the customer users will not be able to see the accident history attribute of the cars. This function will only read the database, it will not alter the database in any way.

The print cars function will not take any input it will simply be triggered by the interface and the applicable event handler. When run the function will read the vehicle database into an ArrayList, it will then split the ArrayList into 2 smaller ArrayLists of sold and unsold vehicles. These 2 ArrayLists will then be printed to the output file “cars-output.txt” 1 object per line. This function will only read the database, it will not alter the database in any way.

The calculate revenue function will take a date as the user input and a duration (1 day or 1 month) and will calculate the total revenue generated by sales for that time period. To do this the function will get the sold cars from the vehicle database, it will then filter out all of the cars that were not sold in the input time period. After this it will sum the price attributes of the remaining cars and this sum will be output to the user. The input for thins function will be error checked to see if it is in the correct format and to see if the input can be parsed as a date. It will also be sanity checked, in that dates such as 32/1/2019 will be invalid for example. This function will only read the database, it will not alter the database in any way.