## Criterion A: Planning

Defining the problem

The client, Mr. John Belsis, is a family friend who owns a small car workshop. During a visit to his garage I asked John how was he able to keep track of all the vehicles in his workshop and he told me that he is using a paper block which had all the client’s necessary information and the date that is allocated for the maintenance of the client’s vehicle. I asked him the reason that he is not using a database management system to keep track of all this information and he told me that he was finding difficult to use one because he was not familiar with the interface and the different options confused him.

On our first interview[[1]](#footnote-1), I showed John MS Access, an application software which allows the user to create and manage a database, but he told me that it was not a good solution for him since he was unfamiliar with the interface and there were too many options which created confusion. With that in mind I proposed to create a program specialized for John’s needs which would save him valuable time and resources.

Rationale for the proposed solution

I decided to create my program on an object-oriented programming language, Java, which will allow me to abstract the characteristics and functions of a real car. By using Java, the program will be easier to understand, debug and expand its operations since an object-oriented language can be implemented more easily for large programs and allows for the use of encapsulation, polymorphism and inheritance to model real world entities.

Based on my first interview with John I understood that he wanted to have the ability to store the license plate, frame number, day that the client dropped off the car to the garage and the clients full name on a database (as inputs). The ability to search for a client’s car using the license plate will greatly benefit John with finding its frame number which will in turn allow him to order parts directly to a representative. John would also benefit by having a way to sort the cars in respect to the date that they arrived to the workshop so that he can appoint a suitable date in his calendar for the service.

Stating success criteria

* A data entry that allows John to add a client with his car to the database
* An option that allows John to remove a client with his car from the database
* A quick way of searching for a car using the license plate
* A quick way of searching for a client using his full name
* A quick way of sorting the client based on the day they delivered their car to the garage
* A way to change the appointed dates for the maintenance of a car
* A way to limit the number of cars undergoing maintenance in the garage
* The implementation of an automatic email mechanism to inform the client when the maintenance is over
* A validation method for all data inputs
* An easy to use user interface

## Criterion B: Record of tasks

|  |  |
| --- | --- |
|  |  |
| Task number | Planned action | | Planned outcome | Time estimated | Target completion date | Criterion |
| 1 | Discussed with my teacher the aim of my internal assessment | | I understood better the requirements of the IA | 30 minutes | 26th May 2017 | A |
| 2 | Search for a suitable client | | I found a client named John at a workshop and considered his problem | 20 minutes | 6th July 2017 | A |
| 3 | Think of how to solve John’s problem | | Evaluated that a database with the client’s information would help John with his problem | 30 minutes | 7th July 2017 | A |
| 4 | Discussed with my teacher the aim of my IA | | The teacher approved the aim | 15 minutes | 20th July 2017 | A |
| 5 | Set an appointment with John | | We decided the appointment date | Email | 22nd July 2017 | A |
| 6 | First interview with John | | Understood better John’s requirements for the program | 45 minutes | 25th July 2017 | A |
| 7 | Send the planning to my teacher for approval | | My teacher approved my work | Email | 25th July 2017 | A |
| 8 | Advised John for the rational of the proposed solution | | Showed John the rational for the proposed solution and he approved my idea | 30 minutes | 28th July 2017 | A |
| 9 | Took photographs of how the program will look like | | I placed the text in notepad and then took a screenshot | 1h 30 minutes | 16th August 2017 | B |
| 10 | Showed John the photographs | | John suggested that I include some artwork in the menu | 20 minutes | 17th August 2017 | B |
| 11 | Included artwork in the menu to get John’s approval | | John approved the new menu | Email | 18th August 2017 | B |
| 12 | Second Interview with John | | John was very happy with the changes in the menu | 20 minutes | 20th September 2017 | B |
| 13 | Design the flow of data | | Made a data flow diagram | 30 minutes | 21st September 2017 | B |
| 14 | Design the way the methods would operate | | Made flowcharts for all the methods of the program | 3 h | 23rd September 2017 |  |
| 15 | Design the main methods of the program | | Made pseudocode for the main methods of the program | 2 h | 24th September 2017 | B |
| 16 | Create a testing strategy based on the initial success criteria | | Constructed a testing strategy | 1h 30minutes | 24th September 2017 | B |
| 17 | Associate the responsibilities of each class | | Created a list of the classes and their proposed solutions | 1h | 25th September 2017 | B |
| 18 | Design UML diagrams | | Constructed the UML diagrams | 20 minutes | 25th September 2017 | B |
| 19 | Begin writing the program in BlueJ | | Created the main menu | 1 h | 26th September 2017 | C |
| 20 | Create the Car and Client object | | Completed the task | 30 minutes | 26th September 2017 | C |
| 21 | Create a readFile method responsible for reading a txt file | | Made the readFile method | 1h | 29th September 2017 | C |
| 22 | Create a addClient and remove Client method responsible for adding and removing a client to the file | | Made the addClient method | 1h | 29th September 2017 | C |
| 23 | Create an input terminal where the user would input the client’s information | | Made the input terminal | 1h | 1st October 2017 | C |
| 24 | Added regular expressions and error handling methods on the input terminal to ensure data accuracy | | Completed the task | 1h | 2nd October 2017 | C |
| 25 | Created a method responsible for outputting the information of all the clients in the file | | Made the outputAllClients method | 45 minutes | 2nd October 2017 | C |
| 26 | Asked my teacher his opinion on my progress | | My teacher approved my work | 20 minutes | 3rd October 2017 | C |
| 27 | Create a quicksort method that would sort the information in the file and override the file with the sorted information | | Made the quicksort method | 2h | 3rd October 2017 | C |
| 28 | Create a BinarySearch responsible for searching the sorted array of the clients information | | Made the Binary Search method | 2h | 4th October 2017 | C |
| 29 | Meeting with John to show him my progress | | John was pleased with my progress | 25 minutes | 4th October 2017 | C |
| 30 | Modify the BinarySearch method and make it recursive | | Modified the BinarySearch method into a recursive one | 1h 30 minutes | 9th October 2017 | C |
| 31 | Make a method that would change the appointed date for a client and override the information on the file | | Made the method changeAppointedDate | 2h | 10th October 2017 | C |
| 32 | Use encapsulation on the Car and Client objects | | Encapsulated the variables of the car and client object | 30 minutes | 1st November 2017 | C |
| 33 | Employ the artwork in the menu | | Successfully included the artwork in the menu | 25 minutes | 3rd November 2017 | C |
| 34 | Make a method that shows if the client’s car has been serviced | | Made the method serviceIsCompleted which used the binarySearch to search for the client with the serviced car | 1h | 10th November 2017 | C |
| 35 | Separate the file access methods from the main class of the program | | Seperated the methods that were accessing the file and put them in a separate class called CarFile | 2h | 11th November 2017 | C |
| 36 | Fix the main class so that it is connected with the other classes | | The class Garage was modified to access all the other classes | 5h | 20th November 2017 | C |
| 37 | Debug a problem from the read method | | Found a bug with the way the program was reading the file and fixed it by using tokens that separated the information of the client | 4h | 21st November 2017 | C |
| 38 | Create a method responsible for sending an email to the clients with a serviced car | | Created the emailClients method which emailed the clients with a serviced car informing them that their car is ready for pickup | 5h | 24th November 2017 | C |
| 39 | Create a method responsible for encrypting the email address when the program closes and decrypting the address when the program starts | | Created the encrypt and decrypt method | 2h | 28th November 2017 | C |
| 40 | Bind the computers calender to the program so that John would only be able to enter a later appointed date | | Completed the task | 30 minutes | 1st December 2017 | C |
| 41 | Create comments on all Classes and methods | | I created comments indicating what each method and class does on the program | 2h | 7th December 2017 | C |
| 42 | Debug the File class | | I fixed some bugs in the code so that it would operate harmoniously | 1h | 10th December 2017 | C |
| 43 | Finalize the program | | I made the program more aesthetically pleasing by taking care of the format of the menu | 3h | 11th December 2017 | C |
| 44 | Complete the program making sure that all the techniques worked correctly | | Completed the program | 15h | 23rd December-30th December 2017 | C |
| 45 | Cite all the sources used for the program | | Completed the task | 1h | 3rd January 2018 | C |
| 46 | Create a video of the final program | | Created the video | 3h | 4th January 2018 | D |
| 47 | Final interview with John | | I consulted John to see if the success criteria were met | 30 minutes | 7th of January 2018 | E |
| 48 | Evaluation of the program | | Evaluated based on the starting success criteria | 1h | 10th of January 2018 | E |

## Criterion B: Design

This Document Contains:

* Visualization
* System Flowcharts
* Data Flow Diagrams
* Structure chart
* Pseudocode of the main algorithms
* Proposed Class Responsibilities
* UML Diagrams
* Testing Strategy

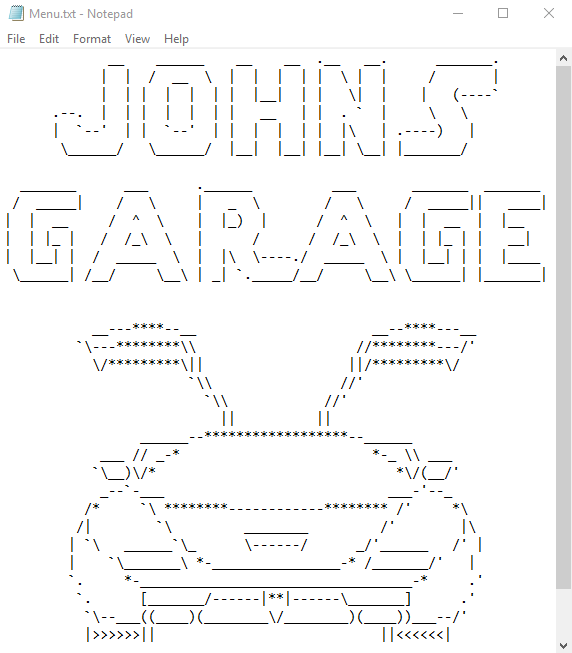
Visualization

Figure 1

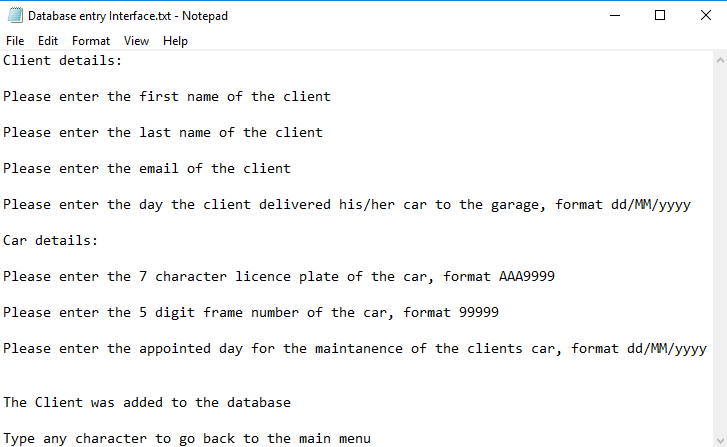
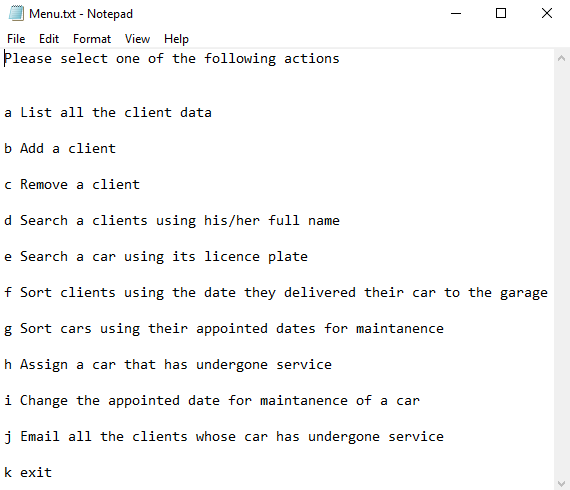


Figure 3

Figure 2

System Flowcharts

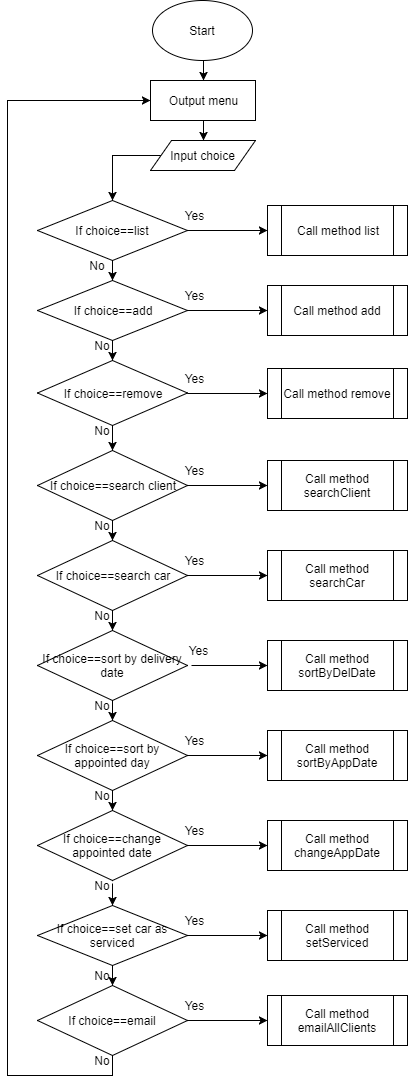
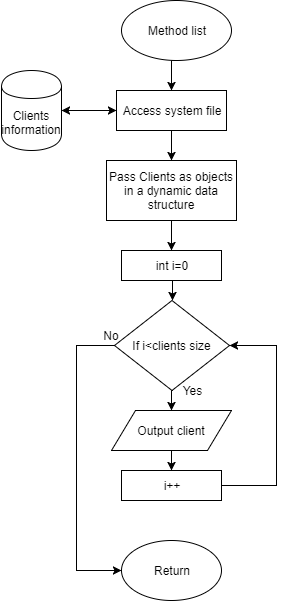


Figure 4



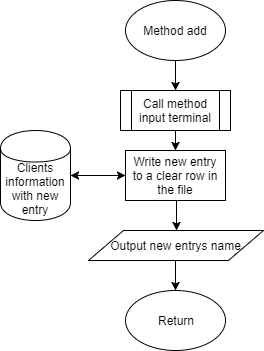


Figure 5

Figure 6

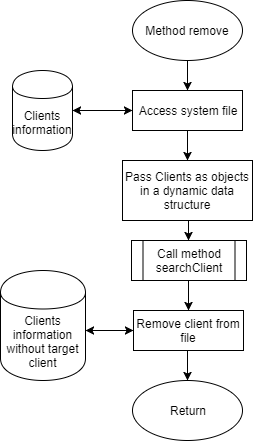


Figure 7

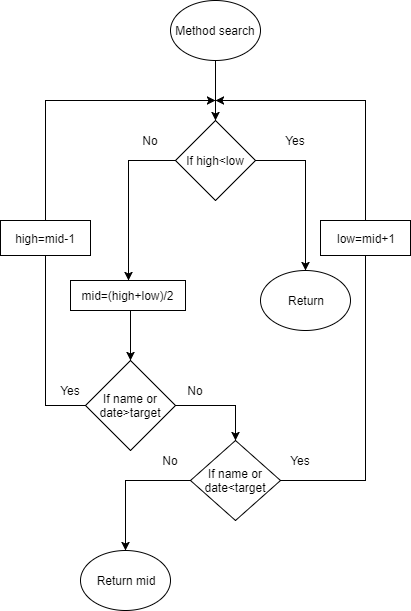


Figure 8

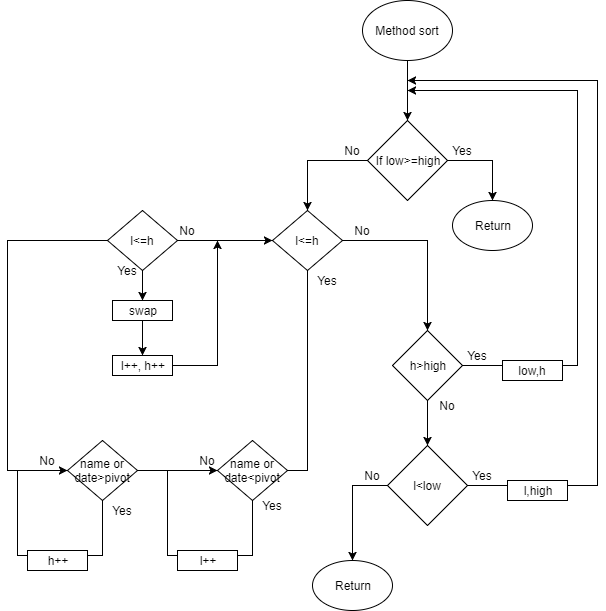
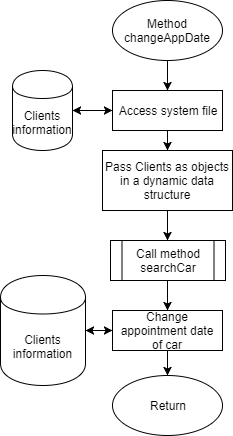


Figure 10



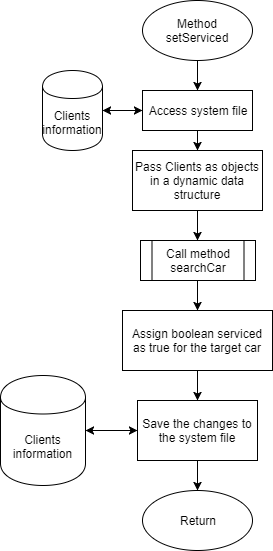


Figure 12

Figure 11

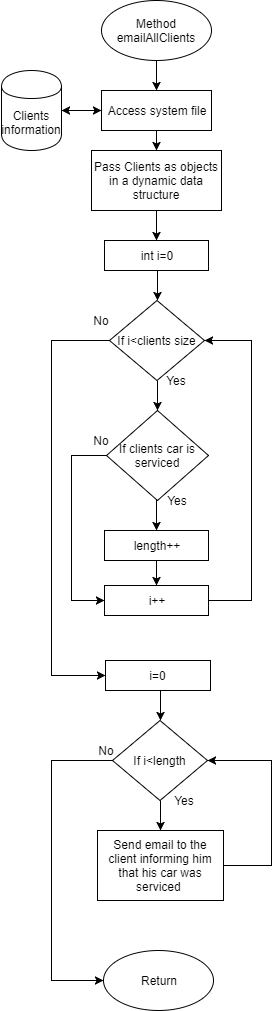


Figure 13

Data flow diagram

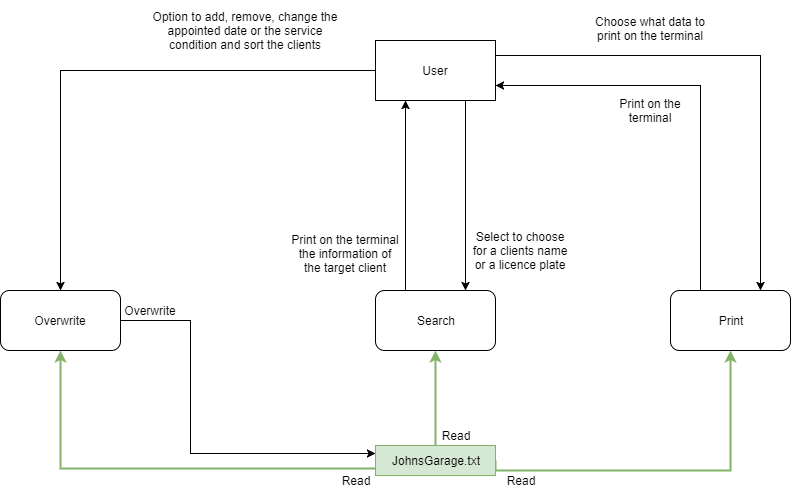


Figure 14

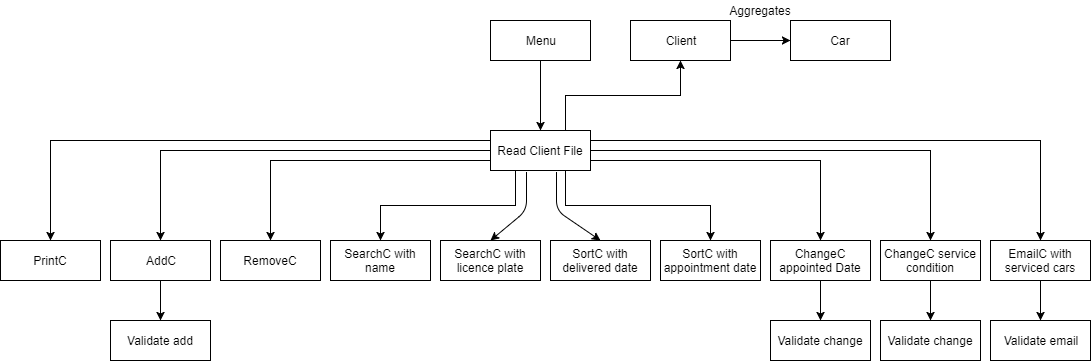
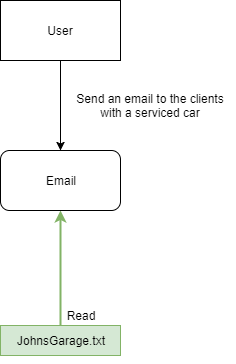


Figure 15

Structure Chart

Figure 16

Pseudocode of the main algorithms

Program readFile

Collection clients

loop A while readLine!null

token=indexOf(":")

nextToken=indexOf(":", token+1)

name=substring(0,token)

email=substring(token,nextToken)

token=nextToken

nextToken=indexOf(":", token+1)

dateOfDelivery=substring(token,nextToken)

token=nextToken

nextToken=indexOf(":", token+1)

licencePlate=substring(token,nextToken)

token=nextToken

nextToken=indexOf(":", token+1)

frameNumber=substring(token,nextToken)

token=nextToken

nextToken=indexOf(":", token+1)

appointedDate=substring(token,nextToken)

token=nextToken

nextToken=indexOf(":", token+1)

serviced=substring(token,nextToken)

CarEntry=Car(licencePlate, frameNumber,appointedDate,serviced)

ClientEntry=Client(name,email,dateOfDelivery,newEntryCar)

Clients.add(newEntryClient)

end loop

return clients

end Program

Program writeToFile

Collection clients=Call readFile

If clients.size()>8

output “Maximum Capacity”

return

end if

loop A clients.size()

output.printToFile(clients.get(A).Name+”:”)

output.printToFile(clients.get(A).email+”:”)

output.printToFile(clients.get(A).dateOfDelivery+”:”)

output.printToFile(clients.get(A).licencePlate+”:”)

output.printToFile(clients.get(A).frameNumber+”:”)

output.printToFile(clients.get(A).appointedDate+”:”)

output.printToFile(clients.get(A).isServiced+”:”)

end loop

close file

end Program

Program emailClients

Collection clients=Call readFile

Collection clientsWithServicedCar

loop A for clients.size()

if clients.get(A).getCar().isServiced()=true

clientsWithServicedCar.add(clients.get(A))

end if

end loop

loop B for clientsWithServicedCar.size()

[username=test@hotmail.com](mailto:username=test@hotmail.com)

password=test123

Create session

send mail to clientsWithServicedCar.get(B)

end loop

end Program

Testing Strategy

|  |  |  |  |
| --- | --- | --- | --- |
| **SUCCESS CRITERION** | **TEST TYPE** | **NATURE OF THE TEST** | **EXAMPLE** |
| Criterion 1 | The user should be able to add a client with his car to the file | Validate the data that the user inputs | *“Error The appointed date needs to have format dd/MM/yyyy”* |
| Criterion 2 | The user should be able to remove a client with his car from the database | Check that the file is not empty | *“Error the file is empty please enter a new client entry first”* |
| Criterion 3 | The user should be able to search for a car using its license plate | Check that the case of the letters is ignored | *Client inputs aBc1234*  *The program outputs the information of the client and his car with licence plate ABC1234* |
| Criterion 4 | The user should be able to search for a client using his full name | Check that the full name in lower case matches a name in the file | *“Error, please enter the name of a client which exist in the database”* |
| Criterion 5 | The user should be able to sort the clients based on the date they delivered their car | Sort the dates in ascending order and override them to the file | *The information of the clients and their car are sorted based on their delivery date* |
| Criterion 6 | The user should be able to change the appointment date for a vehicle | Validate that the appointed date is a later date than the date of delivery | *“The appointed date needs to be after the delivery date”* |
| Criterion 7 | There should be a way to limit the number of cars in the file | Read file and check if the number of cars is less than 8 | *“The garage has reached its maximum capacity”* |
| Criterion 8 | The user should be able to send an email to the clients who’s car is serviced | Validate if the given email of the client is valid | *“Error could not find clients email address”* |
| Criterion 9 | No data entry causes an error | Use regular expressions and error handling methods as a way validate input from the user | *“Error the email address was not of the right format”* |

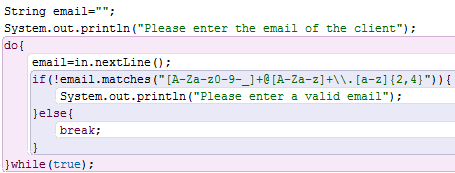
## Criterion C: Development

Techniques used:

* Regular Expressions
* Exception Handling
* ArrayList
* Aggregation
* Encapsulation
* External Library
* Encryption
* Email facility
* File Access
* File Back Up
* Recursion (QuickSort, Binary Search)

Regular Expressions

Regular Expressions are used to validate a string input from the user. I used them because they are easy to implement and they can validate all data types. An alternative to regular expressions would be a try catch method which would throw an exception when the user inputs a value that does not meet the format that I specified.

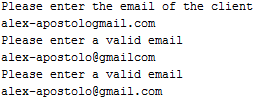


User input for the email address

The user can escape the loop only when he inserts a valid email

Required: An address in the format of aaaa+@aaaa+.com

Figure1: Regular Expression used to validate the email address



Invalid value – Misses the “@”

Invalid value – Misses the “.”

Valid value

Figure2: Example of the above code

Exception Handling

The try catch method is used to handle exceptions so that the program does not crash. I used the try catch so that the program does not stop when the user inputs a non-digit character or an out of range digit. An alternative solution to the try catch are regular expressions but exception handling is a better alternative since it is more reliable and more often updated by the Java Framework.

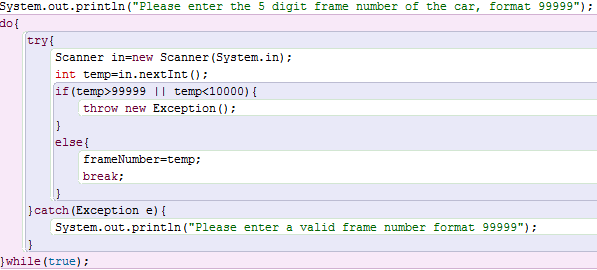


Figure3: Try catch method used to validate the frame number

User input for the frame number

If the user inputs a non-digit value the try catch will handle it

A new exception is thrown when the user inputs a digit larger than 99999 or smaller than 10000

The user can escape the loop only when he inserts a digit that meets the conditions

ArrayList

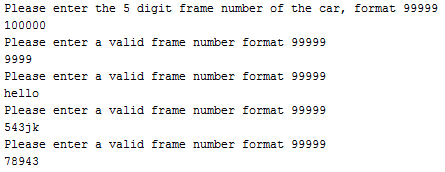


Figure 4: Example of the above code

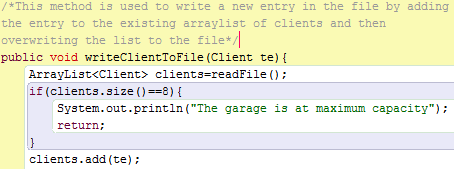
Valid value

Invalid value – The digit is larger than 99999

Invalid value – The digit is smaller than 10000

Invalid value – Non-digit input

ArrayLists are used to store and modify data types. I used ArrayLists because you can dynamically add and remove elements from it and because it is a part of the Java Collections API , providing the ArrayList object with a standard interface. An alternative to ArrayLists are arrays which are limited since they require a fixed size on declaration.



Adds the client te to the ArrayList clients

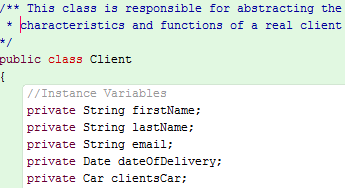
Figure 5: Method that adds the client defined in the input parameter to a temporary ArrayList consisting the clients from “John’s Garage” file

The method readFile returns an ArrayList of type Client

Declaring ArrayList of type Client with name clients

Checks if the number of clients is equal to 8 – The garage has a maximum capacity of eight cars

Aggregation

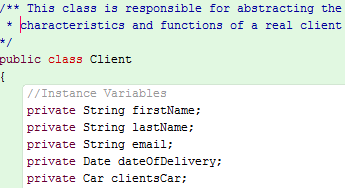
Aggregation is a form of one-way association which represents a HAS-A relationship between two classes. I used aggregation for code reusability purposes since I would have to access only the Client class when adding or removing a client

The class Client HAS-A instance variable, clientsCar, of the class Car

Figure7: Demonstration of the aggregation relationship

Encapsulation

Encapsulation, also known as data hiding, is used to hide the instance variables of a class from other classes. To access or change the values of these instance variables from other classes, mutators and accessors are used. I used encapsulation to ensure that instance variables of an object are not used accidentally by other classes.



These instance variables are private. Therefore, they can only be accessed from the class Client

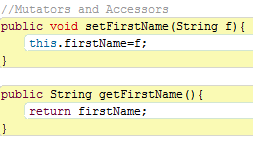
Figure 5: Example of Encapsulation

Returns the instance variable firstName

Sets the instance variable firstName to the input parameter of the method

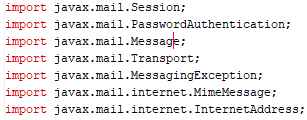
Public method accessible from other classes

Figure 6: Mutators and Accessors used to retrieve and modify instance variables from other classes



External Libraries

The Java framework has a limited Internal Library. External Libraries exist to expand the operations of the Java framework.



The javax.mail External Library is used to create an email facility

Figure 7: The javax.mail External Library

Encryption

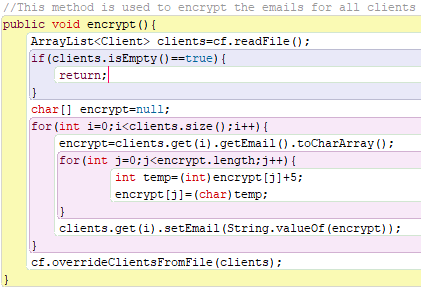
Encryption refers to the translation of plain text into cipher text. Using encryption is a way of achieving data security. I used encryption to secure that anyone accessing the file with the clients information cannot retrieve the email addressses. The encryption occurs when the program is terminated and the decryption occurs when the program is executed.

Overrides the clients in the file with those that have their email encrypted

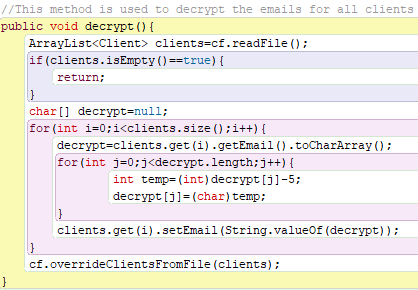
Retrieves the email of the client at index i and converts it to a character array

If the file is empty then the method returns

Figure 8: Encryption of the email address in the clients file



Encryption: Displaces the Ascii value of the characters by +5 making the email address unreadable



Retrieves the email of the client at index i and converts it to a character array

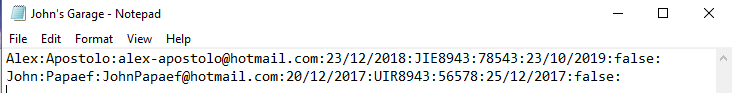
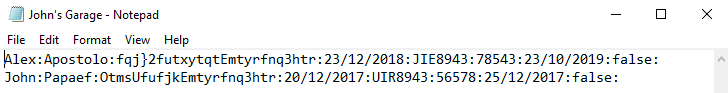
If the file is empty then the method returns

Overrides the clients in the file with those that have their email decrypted

Figure 9: Decryption of the email address

Encryption: Displaces the Ascii value of the characters by -5 making the email address unreadable

The email address is unreadable for anyone accessing the file

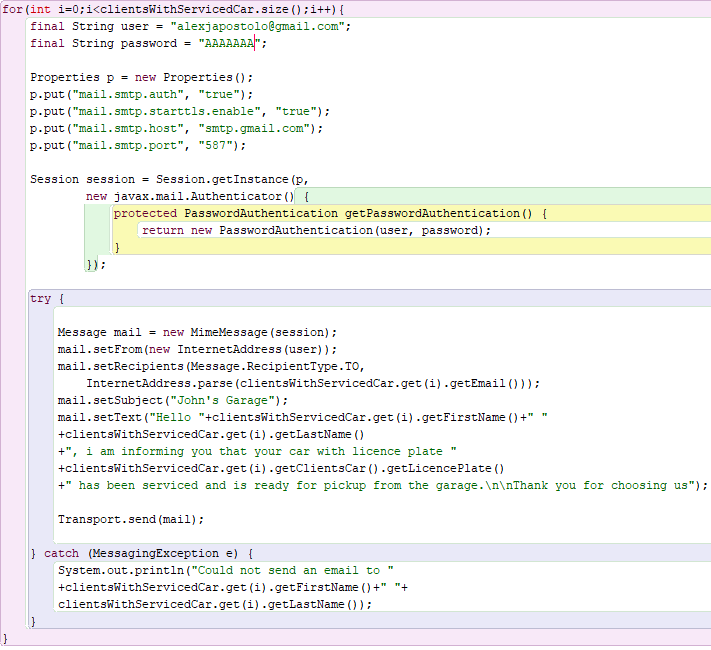
Email Facility

The email address is visible for the program when it is running

Figure 11: “John’s Garage” txt file accessed after executing the program

Figure 10: “John’s Garage” txt file accessed after terminating the program

The email facility is responsible for sending emails to the clients whose car has been serviced. The user can assign a car as serviced from the menu. I used an external library to decrease the complexity of sending an email through java.

File Access

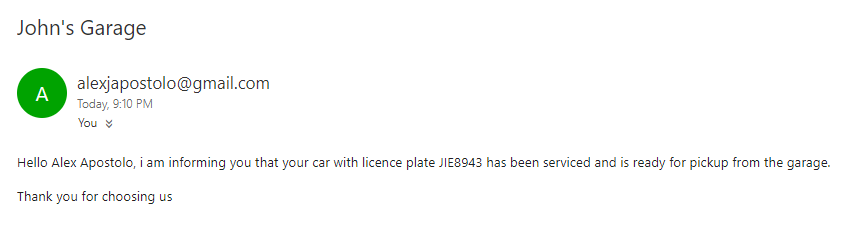


Figure 13: Email informing the client that his car has been serviced

Sends the email - If the receiver’s email is not valid an exception will be thrown

Contents of email

Receiver

Sender

Creates new message

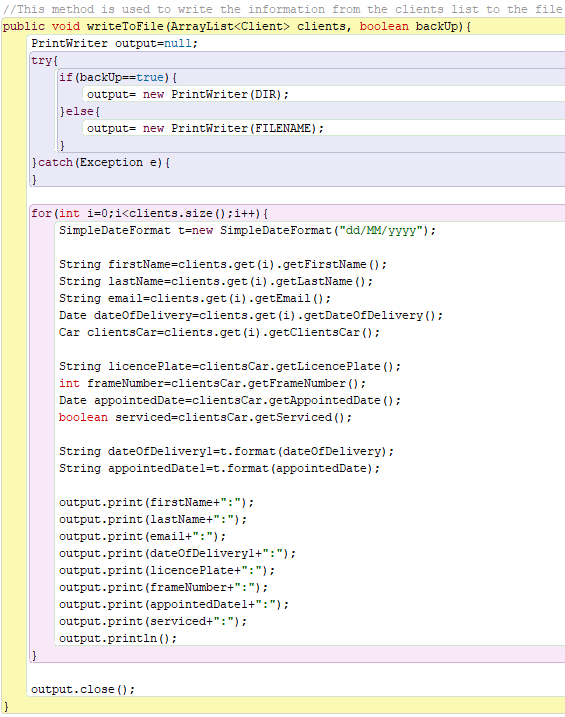
Authentication of the sender’s email and password

Emails are sent from the host

Email and password of the sender

Figure 12: Sending emails to the clients whose car is serviced

I/O operations are performed by the program to access and modify the contents of the file “John’s Garage”. The objects PrintWriter, BufferedReader and PrintWriter were imported from the java.io Internal Library.



Imported from the io Internal Library

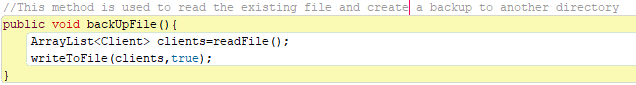
Receives the client’s information from the ArrayList consisting the clients with the new entry

Prints the client’s information to the “John’s Garage” file

Figure 13: Writing the clients information on the file “John’s Garage”

File Back Up

The file “John’s Garage” is backed up in a second location to protect from data loss in the case where the primary location is corrupted. The email addresses in the second location are also encrypted.



The Boolean “true” indicates that the file will be backed up in the second location

The method readFile returns an ArrayList of type Client

Figure 14: Method responsible for backing up “John’s Garage” file to a second location

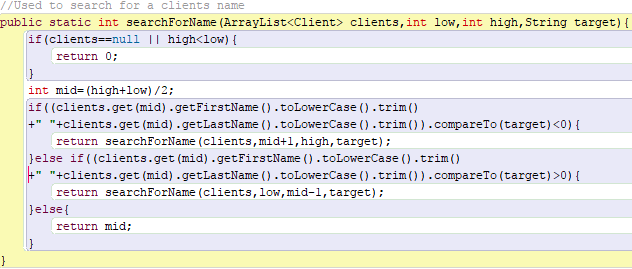
Recursion (QuickSort, BinarySearch)

Recursion occurs when a method calls itself until a terminal condition is met. I implemented the BinarySearch recursively because it has a better big-O than the non-recursive method. An alternative solution to Recursion is Iteration, where a loop is used to perform the binary search which is more time inefficient.

Terminal Condition returns integer

The method calls itself therefore recursion occurs

Figure 15: Recursive BinarySearch – Searches for a name in the clients list



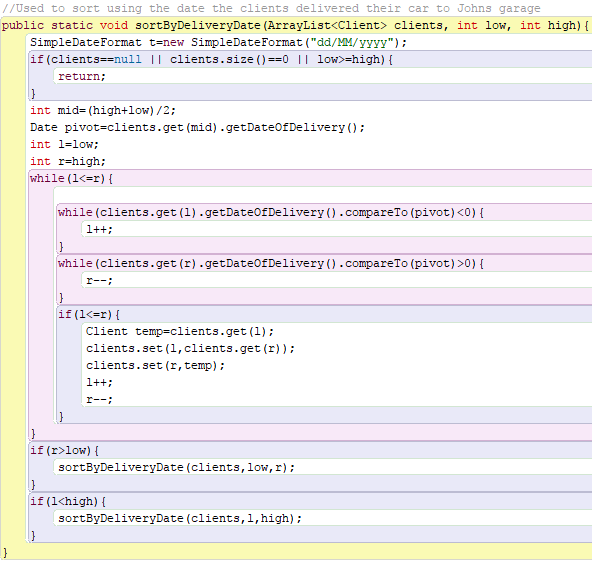


Figure 16: QuickSort – Sorts the clients list using the delivery date

The method calls itself therefore recursion occurs

Terminal Condition

## Criterion E: Evaluation

|  |  |  |
| --- | --- | --- |
| Success Criteria being tested | Completed | Feedback |
| A data entry that allows John to add a client with his car to the database | Yes | “I was able to add a client entry to the file” |
| An option that allows John to remove a client with his car from the database | Yes | “I was able to remove a client from the file” |
| A quick way of searching for a car using the license plate | Yes | “Yes, I was able to search very easily and quickly for a car using its license plate” |
| A quick way of searching for a client using his full name | Yes | “This criterion has been met since I could search for a client using his full name” |
| A quick way of sorting the client based on the day they delivered their car to the garage | Yes | “I was able to sort the clients based on the day they delivered their car to the garage” |
| A way to change the appointed dates for the maintenance of a car | Yes | “This criterion is met since I could appoint different dates for the maintenance of a car” |
| A way to limit the number of cars undergoing maintenance in the garage | Yes | “I couldn’t add more than 8 cars to the file, which is realistic since this is the capacity of my garage” |
| The implementation of an automatic email mechanism to inform the client when the maintenance is over | Yes | “I tested the email feature and it worked perfectly, informing my clients that their car is ready” |
| A validation method for all data inputs | Yes | “I tested the program using invalid data and out of bounds data and the program did not accept them” |
| An easy to use user interface | Yes | “The interface was easy to navigate” |

To assess whether my program had succeed John’s aim, I consulted him and we discussed if the success criteria were met.

**Recommendations for Further Development**

Effectiveness/Efficiency

The requirements of the client are fulfilled and the program runs effectively.

The performance of the recursive binary search is . I chose the recursive binary search instead of an ordinary binary search because it is easier to implement and debug. The disadvantage of using the recursive binary search instead of an ordinary binary search is that it may cause a stack overflow.

The performance of the quick sort is . I chose quick sort instead of bubble sort or selection sort because it is a more time efficient algorithm. The disadvantage of quick sort, like any other recursive algorithm, is that it may cause a stack overflow but for my case I am not sorting more than 8 elements at once so I am not concerned.

Recommendations for future development

Add encryption to the full name of the clients instead of encrypting only their email address.

Add the ability of automatically removing a client from the file after a period of time (e.g. remove a client who has been in the database for more than ten years)

Add the ability to print selected variables from the client file (e.g. Print only the license plate or the frame number of client’s car) (499 words)

## Appendix

First Interview with John

28/9/2017

What kind of business are you running?

I run a garage where me and my colleagues perform repairs and body work on cars. My customer base consists of local people who experience problems with their cars or want to modify them.

How do you keep track of the vehicles on your garage?

I currently use a paper block to keep track of my client’s emails, license plates and the date that I have allocated for the maintenance of their car.

Is the use of a paper block an efficient method of managing your database?

It is not that practical because there are many vehicles in the garage and it is very difficult to find the one you are looking for since you have to search through the whole paper block.

Have you ever considered to use a computer system as a database instead of your paper block?

I have thought about using a computer before but I am afraid that it will confuse me since I do not know how to operate a computer very well.

Could this database management system called MS ACCESS help your problem?

Unfortunately I have tried to use similar types of database management systems before but they have a difficult interface which I am not used to and they have to many options that confuse me.

If I were to create a program that manages your database what requirements would you have?

I would generally want to have information on the license plate, frame number, clients email and clients name. Also I would like for the interface to be easy to understand without many options.

Would the use of an automatic email that is sent through this program when the maintenance of a vehicle is over be helpful?

I do not know whether this is a good idea because there are clients who do not have an email address and need to be called directly. Although I would expect that it would be useful for clients who own an email address.

Second Interview with John

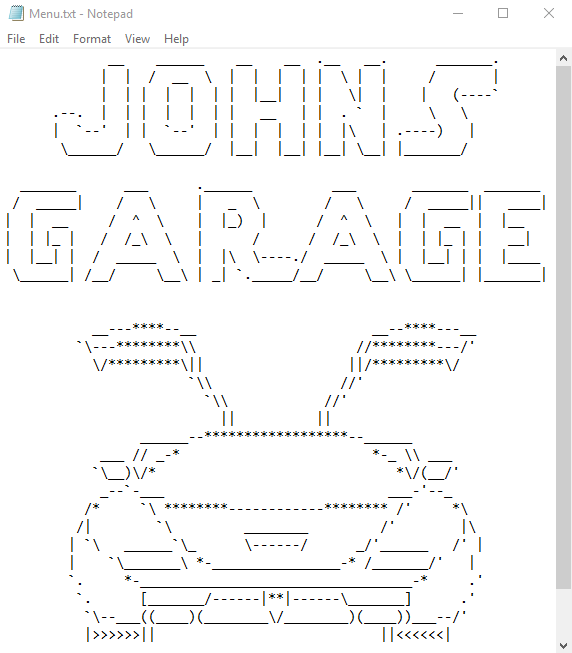
20/9/2017

Figure 2

What do you think of the changes I made to the interface?

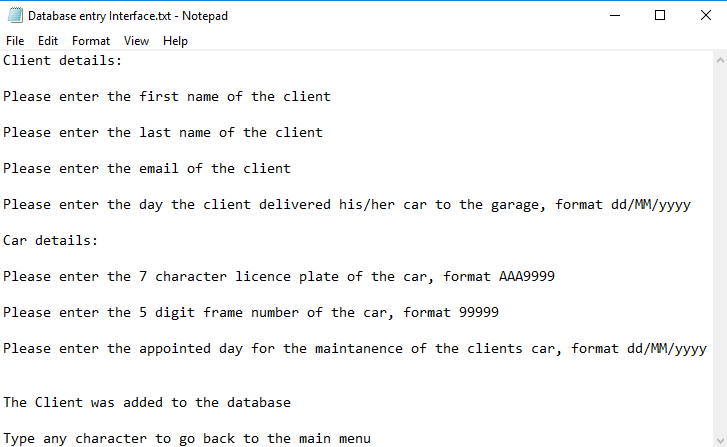
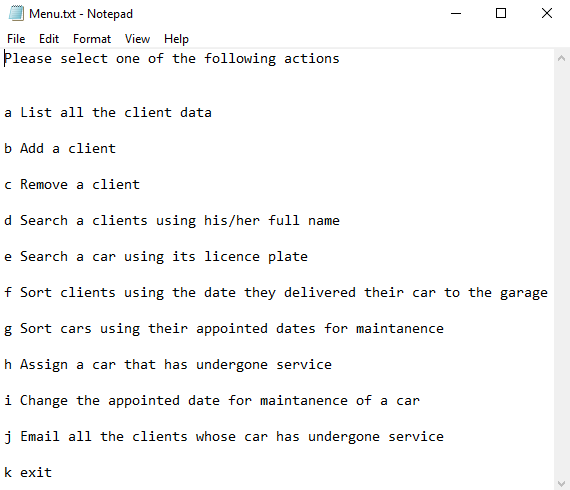


Figure 3

Figure 2

I really like the artwork that you have included, it adds character to the program. The menu seems easy to use and I don’t think I would have any problems navigating. The add function (figure 3) also seems organized and easy to operate.

Final Interview with John

7/1/2018

Have you tested the ability to add and remove a client entry from the file?

Yes, I was able to add and remove a client from the file.

Could you search for a car using its license plate effectively?

Yes, I was able to search for a car’s license plate very easily and quickly.

Where you able to search for a client using his/her full name?

Yes, I could search for a client using their full name and the program also gave me all of their information.

Could you sort the clients based on the day they delivered their car to the garage?

Yes, I was able to sort the clients based on the day they delivered their car to the garage.

Could you change the appointed date of a vehicle?

Yes, I was able to change the appointed date through the interface.

Were you able to add more than 8 cars to the garage without the program telling you that it has reached its limit?

I couldn’t add more than 8 cars to the file, which is realistic since this is the capacity of my garage.

Did the email feature I added to the program informed the clients who had their car serviced?

I tested the email feature and it worked perfectly, informing my clients that their car is ready

Where you able to validate if the program accepted incorrect data?

I tested the program using random data and out of bounds data and everything worked fine.

Was the interface easy to use and navigate?

Yes, I did not find any problems navigating to my selected destination.

Program

import java.util.Scanner;

import java.util.ArrayList;

import java.util.Date;

import java.util.Properties;

import java.text.SimpleDateFormat;

import javax.mail.Session;

import javax.mail.PasswordAuthentication;

import javax.mail.Message;

import javax.mail.Transport;

import javax.mail.MessagingException;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.InternetAddress;

/\*\*

\* This class is responsible for providing the user with an interactable menu

\* Furthermore this is the main class from where all the other classes are accessed

\*/

public class Garage{

CarFile cf;

boolean first=true;

Scanner in=new Scanner(System.in);

public static void main(String[] args){

new Garage();

}

public Garage(){

cf=new CarFile();

showMenu();

}

//This method displays a menu where the user can choose between the 11 different options

public void showMenu(){

if(first){

decrypt();

first=false;

System.out.println("");

System.out.println(" \_\_ \_\_\_\_\_\_ \_\_ \_\_ ,\_\_ \_\_, \_\_\_\_\_\_\_,");

System.out.println(" | | / \_\_ \\ | | | | | \\ | | / |");

System.out.println(" | | | | | | | |\_\_| | | \\| | | (----`");

System.out.println(" ,--, | | | | | | | \_\_ | | , ` | \\ \\ ");

System.out.println(" | `--' | | `--' | | | | | | |\\ | ,----) |");

System.out.println(" \\\_\_\_\_\_\_/ \\\_\_\_\_\_\_/ |\_\_| |\_\_| |\_\_| \\\_\_| |\_\_\_\_\_\_\_/");

System.out.println(" ");

System.out.println(" \_\_\_\_\_\_\_ \_\_\_ ,\_\_\_\_\_\_ \_\_\_ \_\_\_\_\_\_\_ \_\_\_\_\_\_\_");

System.out.println(" / \_\_\_\_\_| / \\ | \_ \\ / \\ / \_\_\_\_\_|| \_\_\_\_|");

System.out.println("| | \_\_ / ^ \\ | |\_) | / ^ \\ | | \_\_ | |\_\_ ");

System.out.println("| | |\_ | / /\_\\ \\ | / / /\_\\ \\ | | |\_ | | \_\_| ");

System.out.println("| |\_\_| | / \_\_\_\_\_ \\ | |\\ \\----,/ \_\_\_\_\_ \\ | |\_\_| | | |\_\_\_\_ ");

System.out.println(" \\\_\_\_\_\_\_| /\_\_/ \\\_\_\\ | \_| `,\_\_\_\_\_/\_\_/ \\\_\_\\ \\\_\_\_\_\_\_| |\_\_\_\_\_\_\_|");

System.out.println("");

System.out.println("");

try{

Thread.sleep(1500);

}catch(Exception e)

{

}

System.out.println(" \_\_---\*\*\*\*--\_\_ \_\_--\*\*\*\*---\_\_");

System.out.println(" `\\---\*\*\*\*\*\*\*\*\\\\ //\*\*\*\*\*\*\*\*---/' ");

System.out.println(" \\/\*\*\*\*\*\*\*\*\*\\|| ||/\*\*\*\*\*\*\*\*\*\\/ ");

System.out.println(" `\\\\ //'");

System.out.println(" `\\\\ //'");

System.out.println(" || ||");

System.out.println(" \_\_\_\_\_\_--\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*--\_\_\_\_\_\_ ");

System.out.println(" \_\_\_ // \_-\* \*-\_ \\\\ \_\_\_ ");

System.out.println(" `\\\_\_)\\/\* \*\\/(\_\_/' ");

System.out.println(" \_--`-\_\_\_ \_\_\_-'--\_ ");

System.out.println(" /\* `\\ \*\*\*\*\*\*\*\*------------\*\*\*\*\*\*\*\* /' \*\\ ");

System.out.println(" /| `\\ \_\_\_\_\_\_\_\_ /' |\\ ");

System.out.println(" | `\\ \_\_\_\_\_\_`\\\_ \\------/ \_/'\_\_\_\_\_\_ /' | ");

System.out.println(" | `\\\_\_\_\_\_\_\_\\ \*-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-\* /\_\_\_\_\_\_\_/' |");

System.out.println(" `. \*-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-\* .' ");

System.out.println(" `. [\_\_\_\_\_\_\_/------|\*\*|------\\\_\_\_\_\_\_\_] .'");

System.out.println(" `\\--\_\_\_((\_\_\_\_)(\_\_\_\_\_\_\_\_\\/\_\_\_\_\_\_\_\_)(\_\_\_\_))\_\_\_--/' ");

System.out.println(" |>>>>>>|| ||<<<<<<|");

System.out.println("");

System.out.println("");

try{

Thread.sleep(1000);

}catch(Exception e)

{

}

}

System.out.println("Please select one of the following actions");

System.out.println();

System.out.println("a List all the client data");

System.out.println("b Add a client");

System.out.println("c Remove a client");

System.out.println("d Search a client using their full name");

System.out.println("e Search a car using its licence plate");

System.out.println("f Sort clients using the date they delivered their car to the garage");

System.out.println("g Sort cars using their appointed dates for maintanence");

System.out.println("h Assign a car that has undergone service");

System.out.println("i Change the appointed date for maintanence of a car");

System.out.println("j Email all the clients whose car has undergone service");

System.out.println("k exit");

try{

Scanner in=new Scanner(System.in);

String selection=in.nextLine();

char selection1=selection.charAt(0);

if(selection.length()>1 || selection==null){

System.out.println("\nError please enter a character between a and k\n");

showMenu();

}else if(selection1<'a'||selection1>'k'){

System.out.println("\nError please enter a character between a and k\n");

showMenu();

}else{

switch(selection1){

case 'a':listAllClients();

break;

case 'b':addClient();

break;

case 'c':removeClient();

break;

case 'd':binarySearch(true);

break;

case 'e':binarySearch(false);

break;

case 'f':quickSort(true);

break;

case 'g':quickSort(false);

break;

case 'h':serviceIsCompleted();

break;

case 'i':changeAppointedDate();

break;

case 'j':emailClients();

break;

case 'k':encrypt();

cf.backUpFile();

System.exit(0);

}

}

}catch(Exception e){

}

}

//This method lists all the clients to the terminal

public void listAllClients(){

cf.outputAllClients();

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used to access the InputTerminal class and then the CarFile class to add a new Client with his car to the database

public void addClient(){

InputTerminal addClient=new InputTerminal();

Client b=addClient.terminal();

cf.writeClientToFile(b);

System.out.println("\nThe Client "+b.getFirstName()+" "+b.getLastName()+" was added to the database\n ");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used to receive the target clients information and then call the CarFile class to remove the target

public void removeClient(){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

System.out.println("\nThere are no cars to the file\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

System.out.println("\nThe names in alphabetical order are ");

QuickSort.sortByName(clients, 0, clients.size()-1);

for(int i=0;i<clients.size();i++){

System.out.println(clients.get(i).getFirstName().trim()+" "+clients.get(i).getLastName().trim());

}

System.out.println();

System.out.println("Type the full name of the client");

String fn;

boolean match=false;

do{

fn=in.nextLine();

if(!fn.matches("[A-Za-z]+[ ]{1}[A-Za-z]+")){

System.out.println("Please enter a valid full name");

}else{

for(int i=0;i<clients.size();i++){

if(fn.toLowerCase().trim().matches(clients.get(i).getFirstName().toLowerCase().trim()+" "+clients.get(i).getLastName().toLowerCase().trim())){

match=true;

}

}

if(match){

break;

}else{

System.out.println("Please enter a valid full name");

}

}

}while(true);

System.out.println();

cf.removeClientFromFile(fn.toLowerCase());

System.out.println("The Client "+fn+" was removed from the database\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used to call the QuickSort class and to provide the sorted arraylist as a parameter for the BinarySearch class

public void binarySearch(boolean useFullName){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

System.out.println("\nThere are no cars to the file\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

Client temp=null;

if(useFullName){

QuickSort.sortByName(clients, 0, clients.size()-1);

System.out.println("\nThe names in alphabetical order are ");

for(int i=0;i<clients.size();i++){

System.out.println(clients.get(i).getFirstName().trim()+" "+clients.get(i).getLastName().trim());

}

System.out.println();

System.out.println("Type the full name of the client");

String fn;

boolean match=false;

do{

fn=in.nextLine();

if(!fn.matches("[A-Za-z]+[ ]{1}[A-Za-z]+")){

System.out.println("Please enter a valid full name");

}else{

for(int i=0;i<clients.size();i++){

if(fn.toLowerCase().trim().matches(clients.get(i).getFirstName().toLowerCase().trim()+" "+clients.get(i).getLastName().toLowerCase().trim())){

match=true;

}

}

if(match){

break;

}else{

System.out.println("Please enter a valid full name");

}

}

}while(true);

int temporary=BinarySearch.searchForName(clients, 0, clients.size()-1,fn.toLowerCase().trim());

temp=clients.get(temporary);

}else{

QuickSort.sortByLicencePlate(clients, 0, clients.size()-1);

System.out.println("\nThe licence plates in decsending order are ");

for(int i=0;i<clients.size();i++){

System.out.println(clients.get(i).getClientsCar().getLicencePlate().trim());

}

System.out.println();

System.out.println("Type the licence plate of the car");

String fn;

boolean match=false;

do{

fn=in.nextLine();

if(!fn.matches("[A-Za-z]{3}[0-9]{4}")){

System.out.println("Please enter a valid licence plate");

}else{

for(int i=0;i<clients.size();i++){

if(fn.toLowerCase().trim().matches(clients.get(i).getClientsCar().getLicencePlate().toLowerCase().trim())){

match=true;

}

}

if(match){

break;

}else{

System.out.println("Please enter a valid licence plate");

}

}

}while(true);

int temporary=BinarySearch.searchForLicencePlate(clients, 0, clients.size()-1,fn.toLowerCase().trim());

temp=clients.get(temporary);

}

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

System.out.println("\nClient details\n");

System.out.println("Full Name: "+temp.getFirstName().trim()+" "+temp.getLastName().trim());

System.out.println("Email: "+temp.getEmail().trim());

System.out.println("Day the client delivered the car: "+t.format(temp.getDateOfDelivery()));

System.out.println("\nCar details\n");

System.out.println("Licence plate: "+ temp.getClientsCar().getLicencePlate());

System.out.println("Frame number: "+temp.getClientsCar().getFrameNumber());

System.out.println("Appointed Day for the maintanence: "+t.format(temp.getClientsCar().getAppointedDate()));

System.out.println("Has it been serviced: "+temp.getClientsCar().getServiced()+"\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used for accessing the QuickSort class and then the CarFile to overwrite all the clients information with the sorted ones

public void quickSort(boolean useDeliveryDate){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

System.out.println("\nThere are no cars to the file\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

if(useDeliveryDate){

QuickSort.sortByDeliveryDate(clients, 0, clients.size()-1);

}else{

QuickSort.sortByAppointedDate(clients, 0, clients.size()-1);

}

cf.writeToFile(clients,false);

cf.outputAllClients();

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used for assigning a car which has been serviced, furthermore this method accesses the BinarySearch class to find the target client and the CarFile to overwrite the data with the new information

public void serviceIsCompleted(){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

System.out.println("\nThere are no cars to the file\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

QuickSort.sortByLicencePlate(clients, 0, clients.size()-1);

System.out.println("\nThe licence plates in decsending order are ");

for(int i=0;i<clients.size();i++){

System.out.println(clients.get(i).getClientsCar().getLicencePlate().trim());

}

System.out.println("\nType the licence plate of the serviced car");

String fn;

boolean match=false;

do{

fn=in.nextLine();

if(!fn.matches("[A-Za-z]{3}[0-9]{4}")){

System.out.println("Please enter a valid licence plate");

}else{

for(int i=0;i<clients.size();i++){

if(fn.toLowerCase().trim().matches(clients.get(i).getClientsCar().getLicencePlate().toLowerCase().trim())){

match=true;

}

}

if(match){

break;

}else{

System.out.println("Please enter a valid licence plate");

}

}

}while(true);

int temporary=BinarySearch.searchForLicencePlate(clients, 0, clients.size()-1,fn.toLowerCase().trim());

clients.get(temporary).getClientsCar().setServiced(true);

cf.writeToFile(clients,false);

System.out.println("\nThe car of the client "+clients.get(temporary).getFirstName()+" "+clients.get(temporary).getLastName()+" has been assigned as serviced");

System.out.println("\nType any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used to change the appointed date for the service of a car by accessing the CarFile and searching for a target client through the BinarySearch class

public void changeAppointedDate(){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

System.out.println("\nThere are no cars to the file\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

QuickSort.sortByLicencePlate(clients, 0, clients.size()-1);

System.out.println("\nThe licence plates in decsending order are ");

for(int i=0;i<clients.size();i++){

System.out.println(clients.get(i).getClientsCar().getLicencePlate().trim());

}

System.out.println("\nType the licence plate of the car");

String fn;

boolean match=false;

do{

fn=in.nextLine();

if(!fn.matches("[A-Za-z]{3}[0-9]{4}")){

System.out.println("Please enter a valid licence plate");

}else{

for(int i=0;i<clients.size();i++){

if(fn.toLowerCase().trim().matches(clients.get(i).getClientsCar().getLicencePlate().toLowerCase().trim())){

match=true;

}

}

if(match){

break;

}else{

System.out.println("Please enter a valid licence plate");

}

}

}while(true);

int temporary=BinarySearch.searchForLicencePlate(clients, 0, clients.size()-1,fn.toLowerCase().trim());

System.out.println("\nPlease enter the new appointed day for maintanence for the car "+clients.get(temporary).getClientsCar().getLicencePlate());

Date appointedDate=null;

do{

try{

String date=in.nextLine();

if(date.matches("[0-9]{1,2}\\/[0-9]{1,2}\\/[2]{1}[0-9]{1}[0-9]{1}[0-9]{1}")){

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

t.setLenient(false);

appointedDate=t.parse(date);

System.out.println();

break;

}else{

throw new Exception();

}

}catch(Exception e){

System.out.println("Please enter a valid Date format dd/MM/yyyy");

}

}while(true);

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

clients.get(temporary).getClientsCar().setAppointedDate(appointedDate);

cf.writeToFile(clients,false);

System.out.println("The new appointade date of maintanence for the car "+clients.get(temporary).getClientsCar().getLicencePlate()+" has been assigned to "+t.format(appointedDate)+"\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used to email all the clients whose car has undergone service

public void emailClients(){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

System.out.println("\nThere are no clients in the file\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

ArrayList<Client> clientsWithServicedCar=new ArrayList<Client>();

for(int i=0;i<clients.size();i++){

if(clients.get(i).getClientsCar().getServiced()==true){

clientsWithServicedCar.add(clients.get(i));

}

}

for(int i=0;i<clientsWithServicedCar.size();i++){

final String user = "alexjapostolo@gmail.com";

final String password = "Popolola1";

Properties p = new Properties();

p.put("mail.smtp.auth", "true");

p.put("mail.smtp.starttls.enable", "true");

p.put("mail.smtp.host", "smtp.gmail.com");

p.put("mail.smtp.port", "587");

Session session = Session.getInstance(p,

new javax.mail.Authenticator() {

protected PasswordAuthentication getPasswordAuthentication() {

return new PasswordAuthentication(user, password);

}

});

try {

Message mail = new MimeMessage(session);

mail.setFrom(new InternetAddress(user));

mail.setRecipients(Message.RecipientType.TO,

InternetAddress.parse(clientsWithServicedCar.get(i).getEmail()));

mail.setSubject("John's Garage");

mail.setText("Hello "+clientsWithServicedCar.get(i).getFirstName()+" "+clientsWithServicedCar.get(i).getLastName()+", i am informing you that your car with licence plate "+clientsWithServicedCar.get(i).getClientsCar().getLicencePlate()+" has been serviced and is ready for pickup from the garage.\n\nThank you for choosing us");

Transport.send(mail);

} catch (MessagingException e) {

System.out.println("Could not send an email to "+clientsWithServicedCar.get(i).getFirstName()+" "+clientsWithServicedCar.get(i).getLastName());

}

}

System.out.println("\nYou have successfully send an email to the clients who had their car serviced\n");

System.out.println("Type any character to go back to the main menu");

in.nextLine();

System.out.println();

showMenu();

}

//This method is used to encrypt the emails for all clients

public void encrypt(){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

return;

}

char[] encrypt=null;

for(int i=0;i<clients.size();i++){

encrypt=clients.get(i).getEmail().toCharArray();

for(int j=0;j<encrypt.length;j++){

int temp=(int)encrypt[j]+5;

encrypt[j]=(char)temp;

}

clients.get(i).setEmail(String.valueOf(encrypt));

}

cf.writeToFile(clients,false);

}

//This method is used to decrypt the emails for all clients

public void decrypt(){

ArrayList<Client> clients=cf.readFile();

if(clients.isEmpty()==true){

return;

}

char[] decrypt=null;

for(int i=0;i<clients.size();i++){

decrypt=clients.get(i).getEmail().toCharArray();

for(int j=0;j<decrypt.length;j++){

int temp=(int)decrypt[j]-5;

decrypt[j]=(char)temp;

}

clients.get(i).setEmail(String.valueOf(decrypt));

}

cf.writeToFile(clients,false);

}

}

import java.util.Scanner;

import java.util.Calendar;

import java.util.Date;

import java.text.SimpleDateFormat;

/\*\*Through this class the user can input the information of a new client

which is then used to create an instance of the Client class and the Car class\*/

class InputTerminal{

Scanner in=new Scanner(System.in);

public Client terminal(){

//Client details

System.out.println();

System.out.println("Client details:\n");

String firstName="";

System.out.println("Please enter the first name of the client");

do{

firstName=in.nextLine();

if(!firstName.matches("[A-Za-z]+")){

System.out.println("Please enter a valid name");

}else{

break;

}

}while(true);

System.out.println();

String lastName="";

System.out.println("Please enter the last name of the client");

do{

lastName=in.nextLine();

if(!lastName.matches("[A-Za-z]+")){

System.out.println("Please enter a valid name");

}else{

break;

}

}while(true);

System.out.println();

String email="";

System.out.println("Please enter the email of the client");

do{

email=in.nextLine();

if(!email.matches("[A-Za-z0-9-\_]+@[A-Za-z]+\\.[a-z]{2,4}")){

System.out.println("Please enter a valid email");

}else{

break;

}

}while(true);

System.out.println();

Date dateOfDelivery=null;

System.out.println("Please enter the day the client left his car in the format dd/MM/yyyy");

do{

try{

String date=in.nextLine();

Date d=new Date();

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

t.setLenient(false);

dateOfDelivery=t.parse(date);

Calendar c = Calendar.getInstance();

c.setTime (d);

c.add(Calendar.DATE, -1);

d = c.getTime();

if(dateOfDelivery.compareTo(d)<=0){

throw new Exception();

}

System.out.println();

break;

}catch(Exception e){

System.out.println("Please enter a valid Date format dd/MM/yyyy");

}

}while(true);

//Car details

System.out.println("Car details:\n");

String licencePlate="";

System.out.println("Please enter the 7 character licence plate of the car, format AAA9999");

do{

licencePlate=in.nextLine();

if(!licencePlate.matches("[A-Za-z]{3}[0-9]{4}")){

System.out.println("Please enter a valid licence plate format AAA9999");

}else{

break;

}

}while(true);

System.out.println();

int frameNumber=0;

System.out.println("Please enter the 5 digit frame number of the car, format 99999");

do{

try{

Scanner in=new Scanner(System.in);

int temp=in.nextInt();

if(temp>99999 || temp<10000){

throw new Exception();

}

else{

frameNumber=temp;

break;

}

}catch(Exception e){

System.out.println("Please enter a valid frame number format 99999");

}

}while(true);

System.out.println();

Date appointedDate=null;

System.out.println("Please enter the appointed day for the maintanence of the clients car in the format dd/MM/yyyy");

do{

try{

String date=in.nextLine();

Date d=new Date();

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

t.setLenient(false);

appointedDate=t.parse(date);

Calendar c = Calendar.getInstance();

c.setTime (d);

c.add(Calendar.DATE, -1);

d = c.getTime();

if(appointedDate.compareTo(d)<=0){

throw new Exception();

}

if(dateOfDelivery.compareTo(appointedDate)>=0){

System.out.println("The appointed date needs to have a higher value than the date of delivery\n");

throw new Exception();

}

System.out.println();

break;

}catch(Exception e){

System.out.println("Please enter a valid Date format dd/MM/yyyy");

}

}while(true);

boolean serviced=false;

Car temp=new Car(licencePlate,frameNumber,appointedDate,serviced);

Client temp1=new Client(firstName, lastName, email, dateOfDelivery, temp);

return temp1;

}

}

import java.util.ArrayList;

/\*\*This class is responsible for the search operations of the program

\*/

public class BinarySearch{

//Used to search for a clients name

public static int searchForName(ArrayList<Client> clients,int low,int high,String target){

if(clients==null || high<low){

return 0;

}

int mid=(high+low)/2;

if((clients.get(mid).getFirstName().toLowerCase().trim()

+" "+clients.get(mid).getLastName().toLowerCase().trim()).compareTo(target)<0){

return searchForName(clients,mid+1,high,target);

}else if((clients.get(mid).getFirstName().toLowerCase().trim()

+" "+clients.get(mid).getLastName().toLowerCase().trim()).compareTo(target)>0){

return searchForName(clients,low,mid-1,target);

}else{

return mid;

}

}

//Used to search for a cars licence plate

public static int searchForLicencePlate(ArrayList<Client> clients,int low,int high,String target){

if(clients==null || high<low){

return 0;

}

int mid=(high+low)/2;

if(clients.get(mid).getClientsCar().getLicencePlate().toLowerCase().trim().compareTo(target)<0){

return searchForLicencePlate(clients,mid+1,high,target);

}else if(clients.get(mid).getClientsCar().getLicencePlate().toLowerCase().trim().compareTo(target)>0){

return searchForLicencePlate(clients,low,mid-1,target);

}else{

return mid;

}

}

}

import java.util.ArrayList;

import java.util.Date;

import java.text.SimpleDateFormat;

/\*\*This class is responsible for the sorting operations of the program

\*/

public class QuickSort{

//Used to sort using the name of the clients

public static void sortByName(ArrayList<Client> clients, int low, int high){

if(clients==null || clients.size()==0 || low>=high){

return;

}

int mid=(high+low)/2;

String t=clients.get(mid).getFirstName().toLowerCase().trim();

String t1=clients.get(mid).getLastName().toLowerCase().trim();

String pivot=t + t1;

int l=low;

int r=high;

while(l<=r){

while((clients.get(l).getFirstName().toLowerCase().trim()+clients.get(l).getLastName().toLowerCase().trim()).compareTo(pivot)<0){

l++;

}

while((clients.get(r).getFirstName().toLowerCase().trim()+clients.get(r).getLastName().toLowerCase().trim()).compareTo(pivot)>0){

r--;

}

if(l<=r){

Client temp=clients.get(l);

clients.set(l,clients.get(r));

clients.set(r,temp);

l++;

r--;

}

}

if(r>low){

sortByName(clients,low,r);

}

if(l<high){

sortByName(clients,l,high);

}

}

//Used to sort using the licence plate of the cars

public static void sortByLicencePlate(ArrayList<Client> clients, int low, int high){

if(clients==null || clients.size()==0 || low>=high){

return;

}

int mid=(high+low)/2;

String pivot=clients.get(mid).getClientsCar().getLicencePlate().toLowerCase().trim();

int l=low;

int r=high;

while(l<=r){

while(clients.get(l).getClientsCar().getLicencePlate().toLowerCase().trim().compareTo(pivot)<0){

l++;

}

while(clients.get(r).getClientsCar().getLicencePlate().toLowerCase().trim().compareTo(pivot)>0){

r--;

}

if(l<=r){

Client temp=clients.get(l);

clients.set(l,clients.get(r));

clients.set(r,temp);

l++;

r--;

}

}

if(r>low){

sortByLicencePlate(clients,low,r);

}

if(l<high){

sortByLicencePlate(clients,l,high);

}

}

//Used to sort using the date the clients delivered their car to Johns garage

public static void sortByDeliveryDate(ArrayList<Client> clients, int low, int high){

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

if(clients==null || clients.size()==0 || low>=high){

return;

}

int mid=(high+low)/2;

Date pivot=clients.get(mid).getDateOfDelivery();

int l=low;

int r=high;

while(l<=r){

while(clients.get(l).getDateOfDelivery().compareTo(pivot)<0){

l++;

}

while(clients.get(r).getDateOfDelivery().compareTo(pivot)>0){

r--;

}

if(l<=r){

Client temp=clients.get(l);

clients.set(l,clients.get(r));

clients.set(r,temp);

l++;

r--;

}

}

if(r>low){

sortByDeliveryDate(clients,low,r);

}

if(l<high){

sortByDeliveryDate(clients,l,high);

}

}

//Used to sort using the date the user has set for the maintanence of the cars

public static void sortByAppointedDate(ArrayList<Client> clients, int low, int high){

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

if(clients==null || clients.size()==0 || low>=high){

return;

}

int mid=(high+low)/2;

Date pivot=clients.get(mid).getClientsCar().getAppointedDate();

int l=low;

int r=high;

while(l<=r){

while(clients.get(l).getClientsCar().getAppointedDate().compareTo(pivot)<0){

l++;

}

while(clients.get(r).getClientsCar().getAppointedDate().compareTo(pivot)>0){

r--;

}

if(l<=r){

Client temp=clients.get(l);

clients.set(l,clients.get(r));

clients.set(r,temp);

l++;

r--;

}

}

if(r>low){

sortByAppointedDate(clients,low,r);

}

if(l<high){

sortByAppointedDate(clients,l,high);

}

}

}

import java.io.PrintWriter;

import java.io.BufferedReader;

import java.io.FileReader;

import java.util.ArrayList;

import java.util.Date;

import java.text.SimpleDateFormat;

/\*\*This class is responsible for accessing and modifying Johns Garage file and the backup file

\*/

class CarFile{

static final String FILENAME="John's Garage";

static final String DIR="C:\\Users\\Public\\Johns Garage Backup";

//This method is used to access the txt file and read the clients information which is then stored in an arraylist

public ArrayList<Client> readFile(){

ArrayList<Client> ArrayOfClients= new ArrayList<Client>();

try{

FileReader Carfile=new FileReader(FILENAME);

BufferedReader input=new BufferedReader(Carfile);

String line;

while((line=input.readLine())!=null){

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

int token =line.indexOf(":");

int nextToken=line.indexOf(":",token+1);

String firstName=line.substring(0, token);

String lastName=line.substring(token+1, nextToken);

token=nextToken;

nextToken=line.indexOf(":", token+1);

String email=line.substring(token+1,nextToken);

token=nextToken;

nextToken=line.indexOf(":", token+1);

Date dateOfDelivery=t.parse(line.substring(token+1, nextToken));

token=nextToken;

nextToken=line.indexOf(":", token+1);

String licencePlate=line.substring(token+1,nextToken);

token=nextToken;

nextToken=line.indexOf(":", token+1);

int frameNumber=Integer.parseInt(line.substring(token+1,nextToken));

token=nextToken;

nextToken=line.indexOf(":", token+1);

Date appointedDate=t.parse(line.substring(token+1, nextToken));

token=nextToken;

nextToken=line.indexOf(":", token+1);

boolean serviced=Boolean.parseBoolean(line.substring(token+1, nextToken));

Car newEntry=new Car(licencePlate, frameNumber, appointedDate, serviced);

Client newEntry1=new Client(firstName, lastName, email, dateOfDelivery, newEntry);

ArrayOfClients.add(newEntry1);

}

}catch(Exception e){

}

return ArrayOfClients;

}

//This method is used to write a new entry in the file by adding

//the entry to the existing arraylist of clients and then overwriting the list to the file

public void writeClientToFile(Client te){

ArrayList<Client> clients=readFile();

if(clients.size()==8){

System.out.println("The garage is at maximum capacity, please delete a car which has been received by the client");

return;

}

clients.add(te);

writeToFile(clients,false);

}

//This method is used to remove an entry from the file

public void removeClientFromFile(String remove){

ArrayList<Client> clients=readFile();

QuickSort.sortByName(clients, 0, clients.size()-1);

int temporary=BinarySearch.searchForName(clients, 0, clients.size()-1,remove);

clients.remove(temporary);

writeToFile(clients,false);

}

//This method is used to read the existing file and create a backup to another directory

public void backUpFile(){

ArrayList<Client> clients=readFile();

writeToFile(clients,true);

}

//This method is used to write the information from the clients list to the file

public void writeToFile(ArrayList<Client> clients, boolean backUp){

PrintWriter output=null;

try{

if(backUp==true){

output= new PrintWriter(DIR);

}else{

output= new PrintWriter(FILENAME);

}

}catch(Exception e){

}

for(int i=0;i<clients.size();i++){

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

String firstName=clients.get(i).getFirstName();

String lastName=clients.get(i).getLastName();

String email=clients.get(i).getEmail();

Date dateOfDelivery=clients.get(i).getDateOfDelivery();

Car clientsCar=clients.get(i).getClientsCar();

String licencePlate=clientsCar.getLicencePlate();

int frameNumber=clientsCar.getFrameNumber();

Date appointedDate=clientsCar.getAppointedDate();

boolean serviced=clientsCar.getServiced();

String dateOfDelivery1=t.format(dateOfDelivery);

String appointedDate1=t.format(appointedDate);

output.print(firstName+":");

output.print(lastName+":");

output.print(email+":");

output.print(dateOfDelivery1+":");

output.print(licencePlate+":");

output.print(frameNumber+":");

output.print(appointedDate1+":");

output.print(serviced+":");

output.println();

}

output.close();

}

//This method is used to output the data from the file to the console

public void outputAllClients(){

ArrayList<Client> clients=readFile();

if(clients.isEmpty()!=true){

for(int i=0;i<clients.size();i++){

SimpleDateFormat t=new SimpleDateFormat("dd/MM/yyyy");

String firstName=clients.get(i).getFirstName();

String lastName=clients.get(i).getLastName();

String email=clients.get(i).getEmail();

Date dateOfDelivery=clients.get(i).getDateOfDelivery();

Car clientsCar=clients.get(i).getClientsCar();

String licencePlate=clientsCar.getLicencePlate();

int frameNumber=clientsCar.getFrameNumber();

Date appointedDate=clientsCar.getAppointedDate();

boolean serviced=clientsCar.getServiced();

String dateOfDelivery1=t.format(dateOfDelivery);

String appointedDate1=t.format(appointedDate);

System.out.println("\nClient "+(i+1)+" details\n");

System.out.println("Name: "+firstName+" "+lastName);

System.out.println("Email: "+email);

System.out.println("Day the client delivered the car: "+dateOfDelivery1);

System.out.println("\nCar "+(i+1)+" details\n");

System.out.println("Licence plate: "+ licencePlate);

System.out.println("Frame number: "+frameNumber);

System.out.println("Appointed Day for the maintanence: "+appointedDate1);

System.out.println("Has it been serviced: "+serviced+"\n");

}

}else{

System.out.println("\nThere are no cars to the file\n");

}

}

}

import java.util.Date;

/\*\* This class is responsible for abstracting the characteristics and functions of a real car

\*/

public class Car

{

//Instance Variables

private String licencePlate;

private int frameNumber;

private Date appointedDate;

private boolean serviced;

//Constructor

public Car(String licencePlate,int frameNumber,Date appointedDate,boolean serviced){

this.licencePlate=licencePlate;

this.frameNumber=frameNumber;

this.appointedDate=appointedDate;

this.serviced=serviced;

}

//Mutators and Accessors

public void setLicencePlate(String l){

this.licencePlate=l;

}

public String getLicencePlate(){

return this.licencePlate;

}

public void setFrameNumber(int f){

this.frameNumber=f;

}

public int getFrameNumber(){

return this.frameNumber;

}

public void setAppointedDate(Date a){

this.appointedDate=a;

}

public Date getAppointedDate(){

return this.appointedDate;

}

public void setServiced(boolean s){

this.serviced=s;

}

public boolean getServiced(){

return this.serviced;

}

}

import java.util.Date;

/\*\* This class is responsible for abstracting the characteristics and functions of a real client

\*/

public class Client

{

//Instance Variables

private String firstName;

private String lastName;

private String email;

private Date dateOfDelivery;

private Car clientsCar;

//Constructor

public Client(String firstName, String lastName, String email, Date dateOfDelivery, Car clientsCar){

this.firstName=firstName;

this.lastName=lastName;

this.email=email;

this.dateOfDelivery=dateOfDelivery;

this.clientsCar=clientsCar;

}

//Mutators and Accessors

public void setFirstName(String f){

this.firstName=f;

}

public String getFirstName(){

return firstName;

}

public void setLastName(String l){

this.lastName=l;

}

public String getLastName(){

return lastName;

}

public void setEmail(String e){

this.email=e;

}

public String getEmail(){

return email;

}

public void setDateOfDelivery(Date d){

this.dateOfDelivery=d;

}

public Date getDateOfDelivery(){

return dateOfDelivery;

}

public void setClientsCar(Car c){

this.clientsCar=c;

}

public Car getClientsCar(){

return clientsCar;

}

}

1. See appendix 1 [↑](#footnote-ref-1)