# Maintenance

## Classes

I have written my program in 9 classes and an enum. These are each described below.

|  |  |
| --- | --- |
| **Class Name** | **Purpose** |
| CarDealership | The class that stores the main method. This also creates the original frame that is used to hold the GUI for the entire app.  CarDealership implements ActionLister so that the actionPerformed method can be overwritten to hold all of the user interactions for the entire app and manage the different JPanels linking together. |
| Car | A blueprint for the car object, the cars will be stored in an ArrayList of these objects  Car extends SearchableObject so it can be searched. |
| Customer | A blueprint for the customer object, the customers will be stored in an ArrayList of Customers.  Customer extends SearchableObject so it can be searched. |
| Employee | A blueprint for the employee object, employees are stored in an ArrayList of Employees.  Employee extends SearchableObject so it can be searched |
| SearchableObject | A basic object that the Car, Customer and Employee classes are based off of. This only has two attributes, one for name (which is the field that will be searched) and one for type; this is where the type of the extended object is stored. |
| Type (enum) | This is the type field that I mentioned above, this enumeration only contains 3 fields: CAR, EMPLOYEE and CUSTOMER |
| Login | This is a GUI class that was designed in NetBeans, this will contain a JPanel that holds the components required to login.  Login extends JPanel so that is can be easily added to the main JFrame in the CarDealership class. |
| Menu | This is another GUI class that has been designed in NetBeans, this is a JPanel that will hold the components of the menu. 4 JButtons and a JLabel.  Menu extends JPanel so that it can be added to the main JFrame the same as adding any other JPanel. |
| Search | The Search panel is also built on netbeans GUI, this is used multiple times in the app as it is constructed with the different ArrayLists of objects to generate different search pages |
| View | The last class is the view panel, this is a GUI class that will be shown when the user clicks the view button. I will find out the type of the object by using the type enum on searchable object. |

## Methods

Below is a table showing the methods I have used in each class.

|  |  |  |
| --- | --- | --- |
| Class | Method | What this method does |
| CarDealership | CarDealership() | This is the constructor for the CarDealership class, this is where the initial JFrame is made and where all of the panels get added to the frame |
| CarDealership | main(String[] args) | This is where the program starts running. This only contains the code the call the above constructor wrapped in SwingUtilies.InvokeLater – this speeds up the program by only creating the mainframe after Swing as finished loading its classes. |
| CarDealership | ActionPerfromed( ActionEvent ae ) | This is where all of the user interactions are stored for when the app will change pages. Pages are managed through JPanels and the action events will change the panels that are by switching though the card layout. |
| SearchableObject | SearchableObject() | This is the constructor that is called in the Employee, Customer and Car classes. This will add a name and a type attribute on the object. The name is the field that is searched and how this is constructed is defined in each class below |
| Car | Car() | This is the constructor for the Car class, this is what will manage the creation of the Car objects and also populate the Type attribute with the CAR value.  The name attribute is constructed by combining the make and the model of the car |
| Car | (Getters and Setters) | There is a large number of methods on this class that simply provided the getting and setting of different attributes on the class. |
| Employee | Employee() | This is the constructor for the employee class. This will build the employee objects and also populate the Type attribute with EMPLOYEE.  The name attributes is constructed by combining the first name and the last name of the employee. |
| Employee | (Getters and Setters) | This class also contains a large number of getters and setters for each attribute on the employee class. |
| Customer | Customer() | This is the constructor for the employee class; this will create the customer objects that will be used by the search pages. This will also populate the Type attribute with the CUSTOMER value.  The name attribute is constructed by combining the first name and the last name of the employee. |
| Customer | (Getters and Setters) | This class also contains a large number of getters and setters for each attribute on the customer class. |
| Login | Login() | This is the constructor for the login class. It must be constructed with the ArrayList of employees that are created in the CarDealership class as this class will search though the usernames in this class and then match up the passwords. |
| Login | initComponents() | This is the method that initialises the components that were made in the netbeans editor. It will initialise these based off of XML forms generated by netbeans that have the same name as the class |
| Login | loginActionPerformed() | This is the method that gets called when the login button is clicked. This will get the text from the username and password fields. I will then use Java 8 steams to filter the list of employees and find an employee object that has a username that matches the one on the employee object.  ArrayLists and filters are examined in more detail at the end of this guide.  Once it has found a matched employee with the username then it will check if the entered password matches the one on the matched employee.  If it matches it will set the done Boolean on the login object to true. If no matches are found then it will call the showWarning method |
| Login | showWarning() | When this is called it will display a warning message at the bottom of the login page telling the user that they have entered an incorrect username and password. |
| Login | clearLoginFields() | When this is called it will empty the values that have been entered into the login field. This is called when the user goes back to the menu |
| Login | jButton1ActionPerformed() | This is called when the help button is pressed it will show the help text for the application |
| Menu | Menu() | This the is the constructor for the Menu class, as this class is simple and just contains GUI components this will simply call the initCompnents() method. |
| Menu | initComponents() | This is the method that initialises the components that were made in the netbeans editor. It will initialise these based off of XML forms generated by netbeans that have the same name as the class |
| Search | Search() | The constructor for the Search class. This will populate the results list by overriding an AbstractListModel with the methods to get that will get the data from the ArrayList of objects that will be searched. |
| Search | initComponets() | This is the method that initialises the components that were made in the netbeans editor. It will initialise these based off of XML forms generated by netbeans that have the same name as the class |
| Search | SearchButtonActionPerformed() | This method will be called then the user clicks the search button. This will set an ArrayList called results to the output from a filter that will look up the object that has been passed through and match the name field.  This method will then set the resultsList to the another Abstract list model that is and override using the data from the results ArrayList |
| Search | JList1ValueChanged() | This method is called when one of the items in the JList has been clicked. This will display the view button and also set the selected attribute on the view object to the value that has been selected. It does this using a ternary that will check if there has been a search and either set the selected value to a value of the search or the value of the original object. |
| View | View() | This is the constructor for the View class that will build the view object. This contains a switch case that will match the type value of the object that it is constructed with and then cast the object to its original type so that the attributes can be easily obtained.  It will then build a string of all the attributes on the object and store that in the show variable. |
| View | initComponets() | This is the method that will construct all of the attributes for the view class and then put them in a JPanel. |

## Use of ArrayLists, filters and lambdas

The main functionally of this app is composed of manipulating ArrayLists to get data out of them. In the Login class (the first panel that the user sees) I construct the class with the ArrayList of employees that I created in my main. When validating the data that has been entered into the program I use a Java 8 filter to find all of the data in the ArrayList that matches the username field. This is done by turning the ArrayList into a stream that will feed all of the data out of the object in a constant steam. This stream is then used by the filter; this will take each instance of the object and compare the Username field with the one that has been entered. The filter is a lambda statement; this means that you are passing the method though as a variable to the filer function, this looks like:

filter(e -> e.getUserName().equlalsIgnoreCase(username))

When the filter finds a match it will store that matched object in the matchedEmployee variable.

## Imports

I have used multiple imports within this project these are listed below for each class:

##### Import

## Hardware and software used

For the development of my program I used IntelliJ IDEA, this is an IDE that allows you to write Java programs is an effect environment and compile and execute them in real time. When maintaining this program you should try and use the same IDE as it was originally developed with. This means that any snippets of code should still work in the same way and you shouldn’t see differences in performance or the functionality of different methods.

IntelliJ IDEA can be downloaded from the official jet brains website [www.jetbrains.com/idea/](http://www.jetbrains.com/idea/)

The minimum requirements to open this program in IntelliJ are as follows.

* 8GB RAM
* Windows 64-bit Operating System
* JDK 7