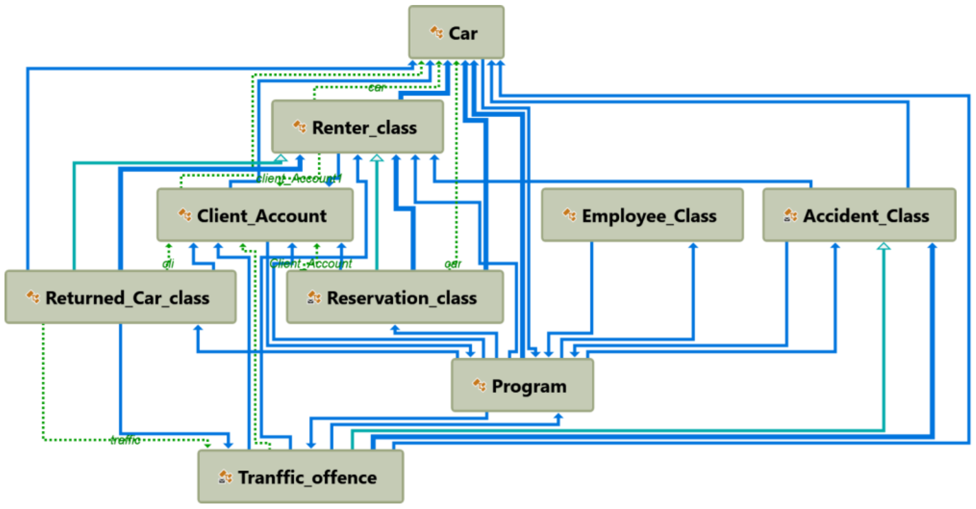
The Main Menu:

the basic explanation of the Main_Menu() function. It provides a user-friendly menu interface, allows the user to make a choice, and directs the program flow accordingly.

```
  =====================================================
||                                                     ||
||                                                     ||
||           (1). Management                           ||
||                                                     ||
||           (2). Rental Employee                      ||
||                                                     ||
||           (3). Client                               ||
||                                                     ||
||           (4). Exit                                 ||
||                                                     ||
  =====================================================
```

```
The Main_Menu() function is responsible for displaying a menu to the user and
handling their choice.

It starts by clearing the console to provide a clean display.

It then uses Console.WriteLine() and Console.ForegroundColor to print a colorful
menu with different options.

It prompts the user to enter their choice by reading an integer input using
int.Parse(Console.ReadLine()).

After reading the user's choice, it clears the console and uses a switch statement
to perform different actions based on the chosen option.

If the user enters 1, it calls the Managment_Menu() function.
If the user enters 2, it calls the Rental_Employee_section() function.
If the user enters 3, it calls the ClientMenue() function.
If the user enters 4, it displays an exit message and terminates the program.
If the user enters any other value, it displays an error message, waits for a key
press, clears the console, and calls Main_Menu() again to restart the men
```



The Management Menu:

```
  =====================================================
||                                                     ||
||     * MANAGMENT PART                                ||
||                                                     ||
||     (1). Resisting main System Data                 ||
||                                                     ||
||     (2). Resisting The Traffic Offence              ||
||                                                     ||
||     (3). Resisting An Accident Data                 ||
||                                                     ||
||     (4). Reset the System                           ||
||                                                     ||
||     (5). Back                                       ||
||                                                     ||
  =====================================================
```

The Managment_Menu() function is responsible for displaying the management menu options and handling the user's choice.

It starts by printing a colorful menu using Console.WriteLine() and Console.ForegroundColor to set the text color.

It prompts the user to enter their choice using int.Parse(Console.ReadLine()).

After reading the user's choice, it clears the console and uses a switch statement to perform different actions based on the chosen option.

If the user enters 1, it calls the Company_Menu() function.

If the user enters 2, it calls the Tranffic_Offence_menu() function.

If the user enters 3, it calls the Accident_Menue() function.

If the user enters 4, it calls the Reset_System() function.

If the user enters 5, it calls the Main_Menu() function.

If the user enters any other value, it displays an error message, waits for a key press, clears the console, and calls Main_Menu() again.

- The Reset_System() function is responsible for resetting the system by deleting stored data. It uses an array of file names and prompts the user for confirmation before deleting the files using FileMode.Truncate and FileAccess.Write.

- The Accident_Menue() function represents the accident data menu. It prompts the user for options and calls different functions based on their choice.

- The Tranffic_Offence_menu() function represents the traffic offense data menu. It prompts the user for options and calls different functions based on their choice.

- Resisting main System Data

(1). About Cars
(2). About Employees
(3). About Departments
(4). About Company
(5). Back

The Company_Menu() function is responsible for displaying the company-related menu options and handling the user's choice.

It starts by printing a colorful menu using Console.WriteLine() and Console.ForegroundColor to set the text color.

It prompts the user to enter their choice using int.Parse(Console.ReadLine()).

After reading the user's choice, it clears the console and uses a switch statement to perform different actions based on the chosen option.

If the user enters 1, it calls the Cars_Menue() function.

If the user enters 2, it calls the EmployeeMenue() function.

If the user enters 3, it calls the Departments_Menue() function.

If the user enters 4, it calls the CompanyMainMenu() function.

If the user enters 5, it calls the Managment_Menu() function.

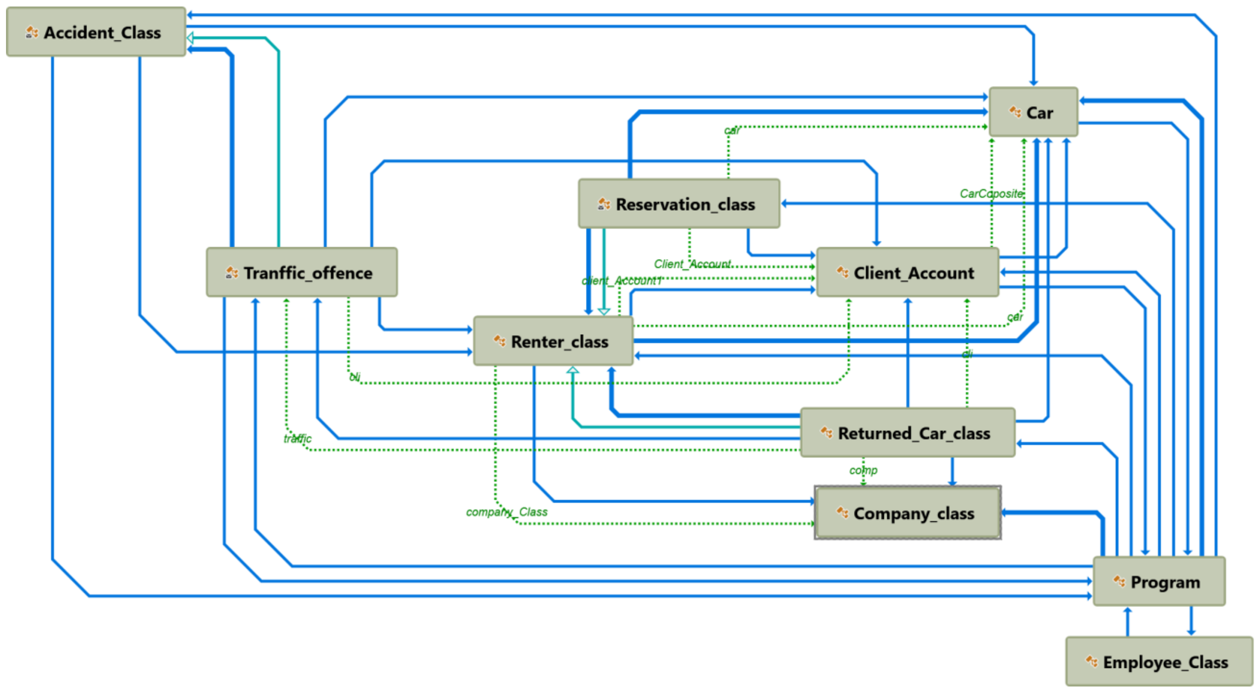
If the user enters any other value, it displays an error message, waits for a key press, clears the console, and calls Company_Menu() again.

The Cars_Menue() function represents the menu for managing cars. It prompts the user for options and calls different functions based on their choice.

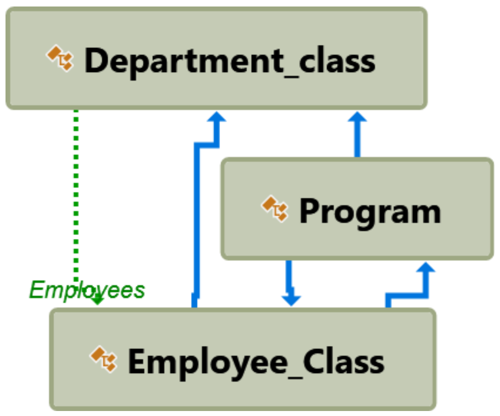
The EmployeeMenue() function represents the menu for managing employees. It prompts the user for options and calls different functions based on their choice.

The Departments_Menue() function represents the menu for managing departments. It prompts the user for options and calls different functions based on their choice.

The CompanyMainMenu() function represents the main menu for managing company-related information. It prompts the user for options and calls different functions based on their choice.



The Rental Employee section!



```
  ==========================================================
||                                                          ||
||     * RENTAL EMPLOYEE PART                               ||
||                                                          ||
||    (1). Display Cars List                                ||
||                                                          ||
||    (2). Search Car Status                                ||
||                                                          ||
||    (3). Rent a Car                                       ||
||                                                          ||
||    (4). Reservation a Car                                ||
||                                                          ||
||    (5). Canceling Reservation                            ||
||                                                          ||
||    (6). Display Specific Car Statue                      ||
||                                                          ||
||    (7). Search for Specific Car Renter                   ||
||                                                          ||
||    (8). Register Return data                             ||
||                                                          ||
||    (9).Display previous renting data                     ||
||                                                          ||
||    (0). Back                                             ||
||                                                          ||
  ==========================================================
```

Let's break down the code:

The Rental_Employee_section() function represents the menu for rental employees. It displays various options and performs different actions based on the user's choice.

It starts by creating instances of the Car, Renter_class, and Reservation_class classes.

The console is cleared and a colorful menu is displayed using Console.WriteLine() and Console.ForegroundColor to set the text color.

The user is prompted to enter their choice using int.Parse(Console.ReadLine()).

After reading the user's choice, the console is cleared again.

A switch statement is used to perform different actions based on the chosen option.

- If the user enters 1, it calls the Display_Details() function of the Car class to display the car details.

- If the user enters 2, it calls the Search_For_Car_Status() function to display the status of a specific car.

- If the user enters 3, it calls the Renting_Data() function of the Renter_class class to record the renting data.

- If the user enters 4, it calls the Renting_Data() function of the Reservation_class class to record the reservation data.

- If the user enters 5, it calls the Cancel_Reservation() function of the Reservation_class class to cancel a reservation.

- If the user enters 6, it calls the DisplayStatusType() function to display the status of cars based on user-specified types.

- If the user enters 7, it calls the Search_For_Renter() function of the Renter_class class to search for a specific car renter.

- If the user enters 8, it calls the Return_Data_Menue() function to register return data.

- If the user enters 9, it calls the display_previus_data() function of the Renter_class class to display previous renting data.

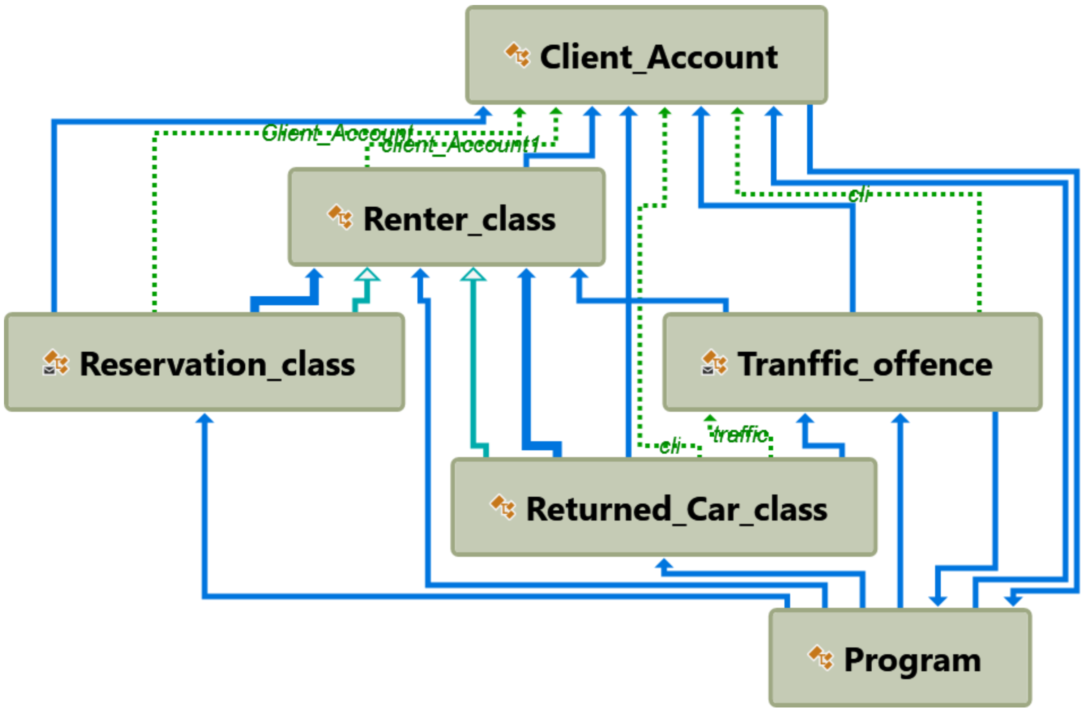
- If the user enters 0 or 10, it calls the Main_Menu() function to go back to the main menu.

- If the user enters any other value, an error message is displayed and the function is called again.

Finally, the user is prompted to press any key to continue, and the console is cleared before calling Rental_Employee_section() again.

The Client Menue:

```
  =====================================================
||                                                     ||
||                                                     ||
||           (1). Login My Account                     ||
||                                                     ||
||           (2). Create New Account                   ||
||                                                     ||
||           (3). Back                                 ||
||                                                     ||
  =====================================================
```



The ClientMenue() function represents the menu for clients. It displays various options and performs different actions based on the user's choice.

It starts by creating an instance of the Client_Account class.

The console is cleared and a colorful menu is displayed using Console.WriteLine() and Console.ForegroundColor to set the text color.

The user is prompted to enter their choice using int.Parse(Console.ReadLine()).

After reading the user's choice, the console is cleared again.

A switch statement is used to perform different actions based on the chosen option.

- If the user enters 1, it calls the login() function of the Client_Account class to log in to their account.
- If the user enters 2, it calls the Create_account() function of the Client_Account class to create a new account.
- If the user enters 3, it calls the Main_Menu() function to go back to the main menu.
- If the user enters any other value, an error message is displayed and the function is called again.

Finally, the user is prompted to press any key to continue, and the console is cleared before calling ClientMenue() again.

The code also includes the Return_Data_Menue() function, which represents the return car part for clients. It displays various options and performs different actions based on the user's choice.

It starts by creating instances of the Car and Returned_Car_class classes.

The console is cleared and a colorful menu is displayed using Console.WriteLine() and Console.ForegroundColor to set the text color.

The user is prompted to enter their choice using int.Parse(Console.ReadLine()).

After reading the user's choice, the console is cleared again.

A switch statement is used to perform different actions based on the chosen option.

- If the user enters 1, it calls the Display_Any_car_Status(1) function of the Car class to display rented cars.
- If the user enters 2, it calls the Returning_Car() function of the Returned_Car_class class to initiate the car return process.
- If the user enters 3, it calls the Main_Menu() function to go back to the main menu.
- If the user enters any other value, an error message is displayed and the function is called again.

Finally, the user is prompted to press any key to continue, and the console is cleared before calling Return_Data_Menue() again.

(OOP) concepts in this project:

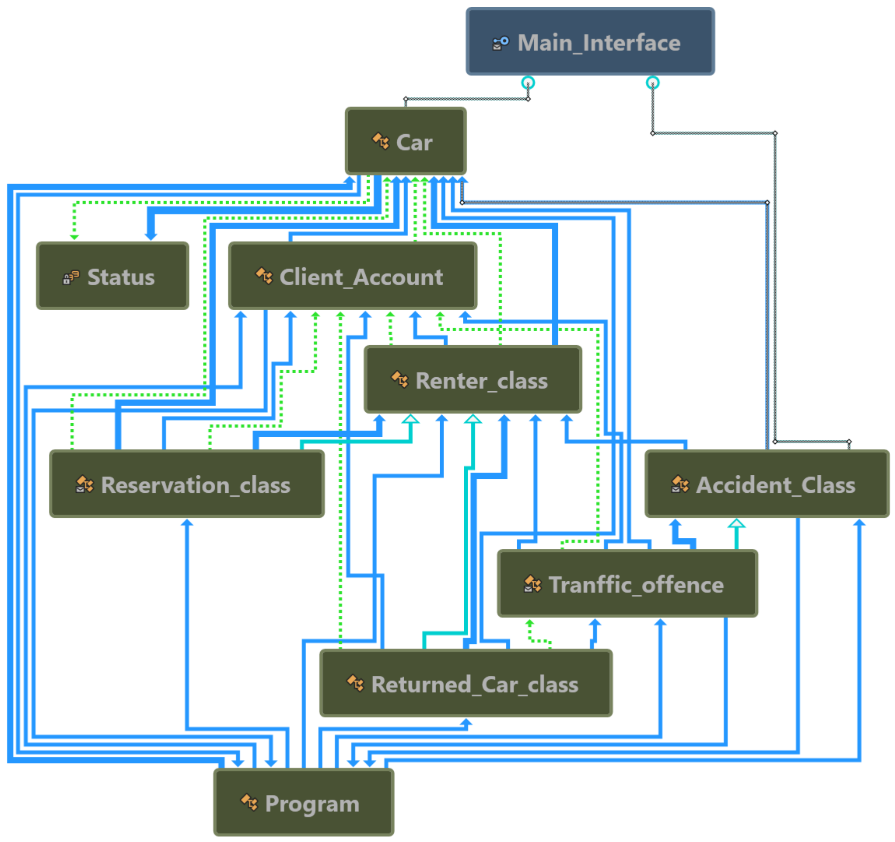
Encapsulation: The project demonstrates encapsulation by encapsulating related data and methods within classes. For example, the Car class encapsulates car details and status, the Renter_class class encapsulates renter information, and the Reservation_class class encapsulates reservation details. Encapsulation helps in organizing and structuring the code, making it more maintainable and modular.

Inheritance: Inheritance is used in the project to establish an "is-a" relationship between classes. For instance, the Car class may have derived classes based on car types, such as Sedan, SUV, and Convertible. These derived classes inherit properties and methods from the base Car class, allowing for code reuse and promoting a hierarchical structure.

Polymorphism: Polymorphism is demonstrated through method overriding and method overloading. Method overriding occurs when derived classes provide their own implementation of a method inherited from a base class. This can be seen in the Display_Any_car_Status() method, which can be overridden by derived classes to display specific car statuses. Method overloading occurs when multiple methods with the same name but different parameter lists exist. An example of method overloading is seen in the Renting_Data() method, which has different versions depending on whether it is called for a renter or a reservation.

Abstraction: Abstraction is employed by hiding implementation details and providing simplified interfaces. The classes in the project abstract away complex functionalities and expose only the necessary methods and properties for interaction. For example, the Car class exposes methods like Display_Details() and Display_Any_car_Status(), allowing users to interact with the car data without needing to know the inner workings of the class.

Association: The project showcases association between classes, where one class is related to another class. For instance, the Renter_class and Reservation_class classes have a relationship with the Car class. The Renter_class class records renting data for a specific car, while the Reservation_class class records reservation data for a specific car. These associations allow for better modeling of real-world relationships and interactions.


Main_Interface
Car
Status
Client_Account
Renter_class
Reservation_class
Accident_Class
Tranffic_offence
Returned_Car_class
Program