Alexandru Majeru

CSC 330

Project #1

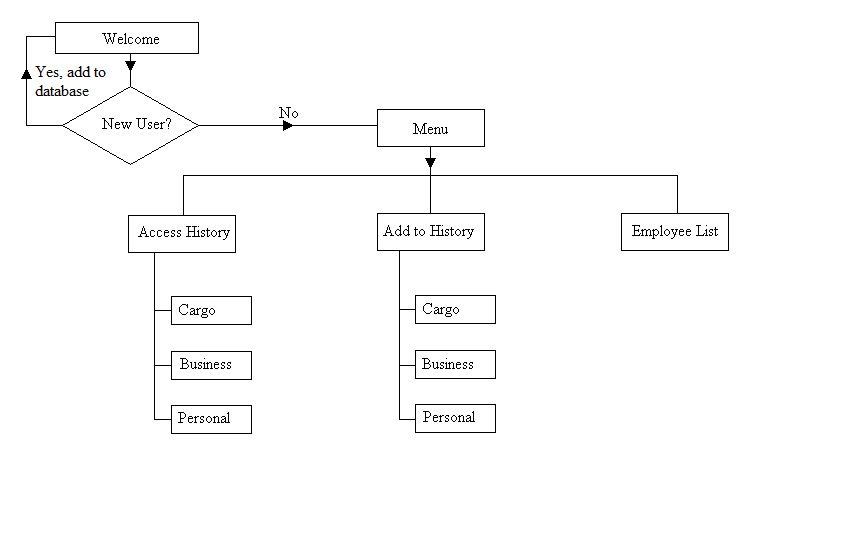
Design Specifications

The system will operate via a menu, through which the user will be presented with a list of options. Initially, the system asks the user to input personal data – the ID string. If the string is present in the system (external file), the program will welcome the user to the system and present he menu. If the ID is not recognized, the user will have an option to input his or her credentials (first name and last name), with the system saving the above. The main menu itself will present the user with three options:

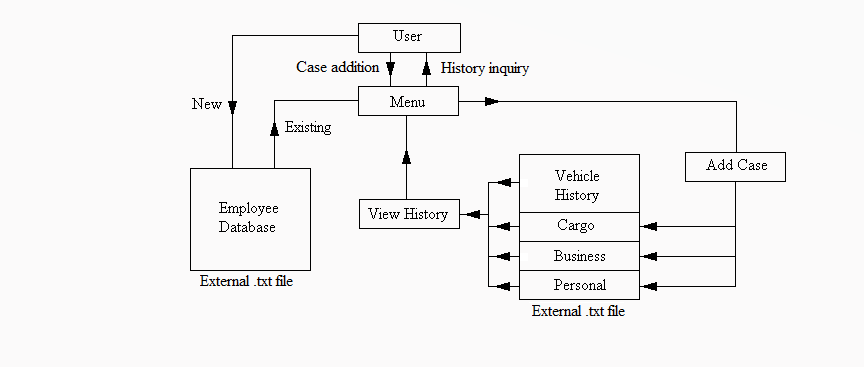
1. View history of vehicle use
2. Add a new case of vehicle use
3. See the list of known employees

Options 1 and 2 will lead to the next step of the program – choosing the type of vehicle the user wants to either see the history of or add to the history of cases.

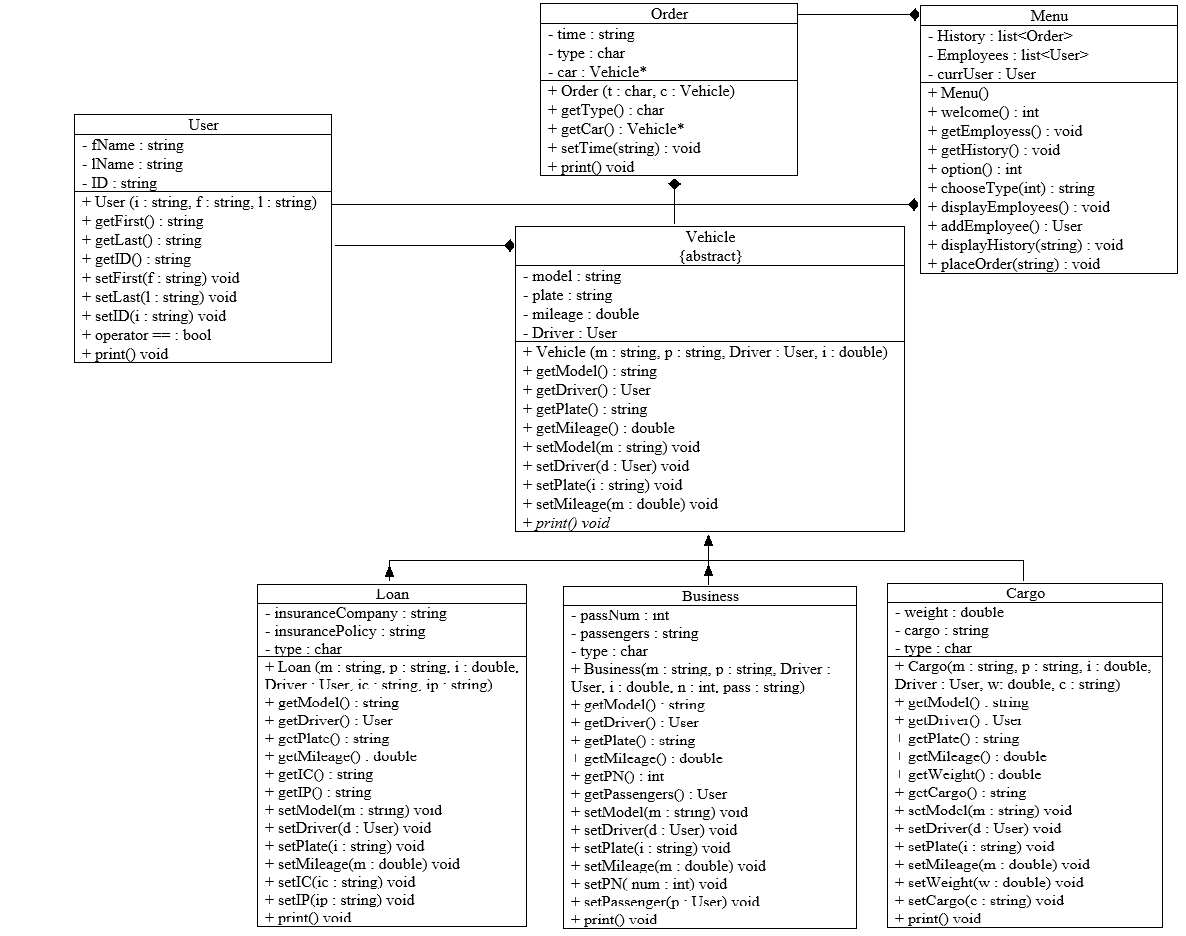
The program will run according to one of the following scenarios:



The data flow in the system will follow the chart displayed below, depending on the choice the user makes in the menu:



The system will consist of a list of classes, through which it will operate on data received from the user:



Class “Order” will be the main point of operation – it will include the type of the vehicle as a character, depending on which a Vehicle type object pointer will be created with the proper vehicle type (cargo, business, loan). The Order class is able to return the type of the vehicle set and the Vehicle contained within, as well as display all the data on the on the console – however, once the order is set through its constructor, you cannot change the vehicle or its type.

The “Vehicle” class is the abstract base class that connects the derived classes containing the data depending on the type of vehicle. It contains the model, plate, mileage of the trip, and a driver (of type User). Its subclasses (Cargo, Business, Loan) contain additional information (like cargo, passenger, or insurance information).

1. The Cargo class will inherit the functions and data from Vehicle class, with addition of two data members – the weight of cargo and the list of materials transported. It will also add mutators and accessors to operate with said data.
2. The Business class will operate similarly to Cargo class in terms of inheritance, but the additional data members will be the number of passengers and a container (list) with a list of passengers (User objects). It is implied that the passengers on the business trip are employees of the company.
3. The Loan class inherits the same data and member functions as the above classes and adds two private string data members containing insurance information of the user – the name of the company and policy.

The “User” class will contain the data about the user – first name, last name, and employee ID. This will be used to check if the user is already in the system, and recorded with each order placed in history.

The Main Menu of the program will operate using a list of Order objects and User objects. Initially, the system will collect the data from external files – the user and history databases. Those will be stored in the two lists. Depending on user actions, these two databases will either be altered or displayed to the user. The user will not be able to change any existing entries through the system – only by opening the .txt files. At the end of each program run, the data stored inside the Order and User objects lists will be updated within the files. The history and user list will be displayed/saved with the most recent entries on top.

|  |
| --- |
| Vehicle  {abstract} |
| - model : string  - plate : string  - mileage : double  - Driver : User |
| + Vehicle (m : string, p : string, Driver : User, i : double)  + getModel() : string  + getDriver() : User  + getPlate() : string  + getMileage() : double  + setModel(m : string) void  + setDriver(d : User) void  + setPlate(i : string) void  + setMileage(m : double) void  + *print() void* |

|  |
| --- |
| Cargo |
| - weight : double  - cargo : string  - type : char |
| + Cargo(m : string, p : string, i : double, Driver : User, w: double, c : string)  + getModel() : string  + getDriver() : User  + getPlate() : string  + getMileage() : double  + getWeight() : double  + getCargo() : string  + setModel(m : string) void  + setDriver(d : User) void  + setPlate(i : string) void  + setMileage(m : double) void  + setWeight(w : double) void  + setCargo(c : string) void  + print() void |

|  |
| --- |
| Business |
| - passNum : int  - passengers : string  - type : char |
| + Business(m : string, p : string, Driver : User, i : double, n : int, pass : string)  + getModel() : string  + getDriver() : User  + getPlate() : string  + getMileage() : double  + getPN() : int  + getPassengers() : User  + setModel(m : string) void  + setDriver(d : User) void  + setPlate(i : string) void  + setMileage(m : double) void  + setPN( num : int) void  + setPassenger(p : User) void  + print() void |

|  |
| --- |
| Loan |
| - insuranceCompany : string  - insurancePolicy : string  - type : char |
| + Loan (m : string, p : string, i : double, Driver : User, ic : string, ip : string)  + getModel() : string  + getDriver() : User  + getPlate() : string  + getMileage() : double  + getIC() : string  + getIP() : string  + setModel(m : string) void  + setDriver(d : User) void  + setPlate(i : string) void  + setMileage(m : double) void  + setIC(ic : string) void  + setIP(ip : string) void  + print() void |

|  |
| --- |
| User |
| - fName : string  - lName : string  - ID : string |
| + User (i : string, f : string, l : string)  + getFirst() : string  + getLast() : string  + getID() : string  + setFirst(f : string) void  + setLast(l : string) void  + setID(i : string) void  + operator == : bool  + print() void |

|  |
| --- |
| Order |
| - time : string  - type : char  - car : Vehicle\* |
| + Order (t : char, c : Vehicle)  + getType() : char  + getCar() : Vehicle\*  + setTime(string) : void  + print() void |

|  |
| --- |
| Menu |
| - History : list<Order>  - Employees : list<User>  - currUser : User |
| + Menu()  + welcome() : int  + getEmployess() : void  + getHistory() : void  + option() : int  + chooseType(int) : string  + displayEmployees() : void  + addEmployee() : User  + displayHistory(string) : void  + placeOrder(string) : void |