Alexandru Majeru

CSC 330

Project #2

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 – personal ID and password. If the data 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 to call an administrator, who has access to registering a new system user, whose data will then be saved for future use. The main menu itself will present the user with a number of options:

1. Book a flight
2. See client information
3. See flights
4. Print boarding pass

For higher status users (administration) the list of options is extended:

1. Reschedule a flight
2. Add new flight
3. Change client information
4. Change user information (see employee list, add new employee)
5. See flight information

The menu is built using C# Windows Forms, which will be interchanging depending on what the user decides to do in the menu. Choosing any of the above options will open a new form, while closing the previous one – with an option to return to it by pressing “Back” button.

The system will be built using a number of classes:

1. Employee
2. ID
3. Password
4. Full Name
5. Status (booking agent or administrator)
6. Client
7. Unique ID
8. Full Name
9. Address
10. Phone Number
11. Email
12. Flight Number
13. Flight Class
14. Seat Preference
15. Meal Preference
16. Special needs

Aside from the data about the users/clients, the system will also have a “flight” class, containing:

1. Flight Number
2. Vessel model
3. Flight Destination
4. Total number of seats (economy and business)
5. Number of reserved seats
6. Time and date of arrival/departure

It is important to note a number of details about the data stored. First, a unique client ID will be assigned by the program itself as a way to distinguish unique clients in the database created. Said database will also act as a form of history of client reservations – with assumption that a single customer is limited to a single flight and cannot create any more bookings if one is still active.

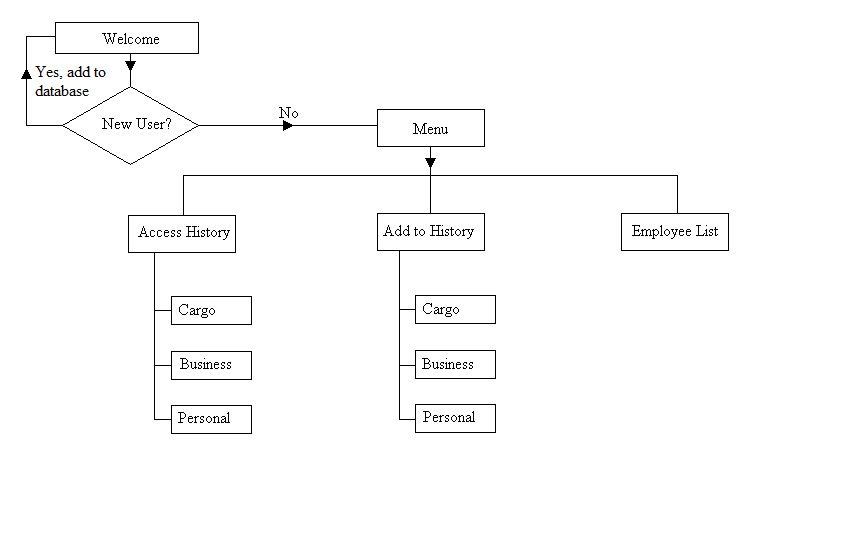
Another assumption is that all the flights are direct and connect two cities – Flin Flon and any airport available in the system. The system will have a check for whether the flight is departing or arriving to Flin Flon.

Furthermore, the system will account for overbooking the flights. When inputting data, the administrator will assign the number of seats to an aircraft – with the program automatically increasing it by 15% for potential overbooking. The only seats overbooked will be in economy class.

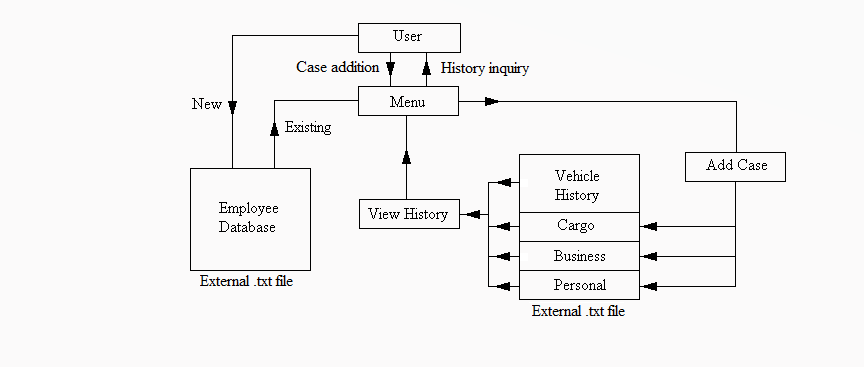
Additionally, the system will not work with any monetary transactions or calculations – any refund options, class change, ticket price information and processing will have to be conducted in a different program.

As mentioned previously, the system GUI will be built using C# Windows Forms for ease of readability and use. Additionally, all the data regarding flights, clients, and employees will be stored in SQL database tables, from which the information will be pulled for further processing or display. Three tables will be created, containing data needed to define objects of classes described above.

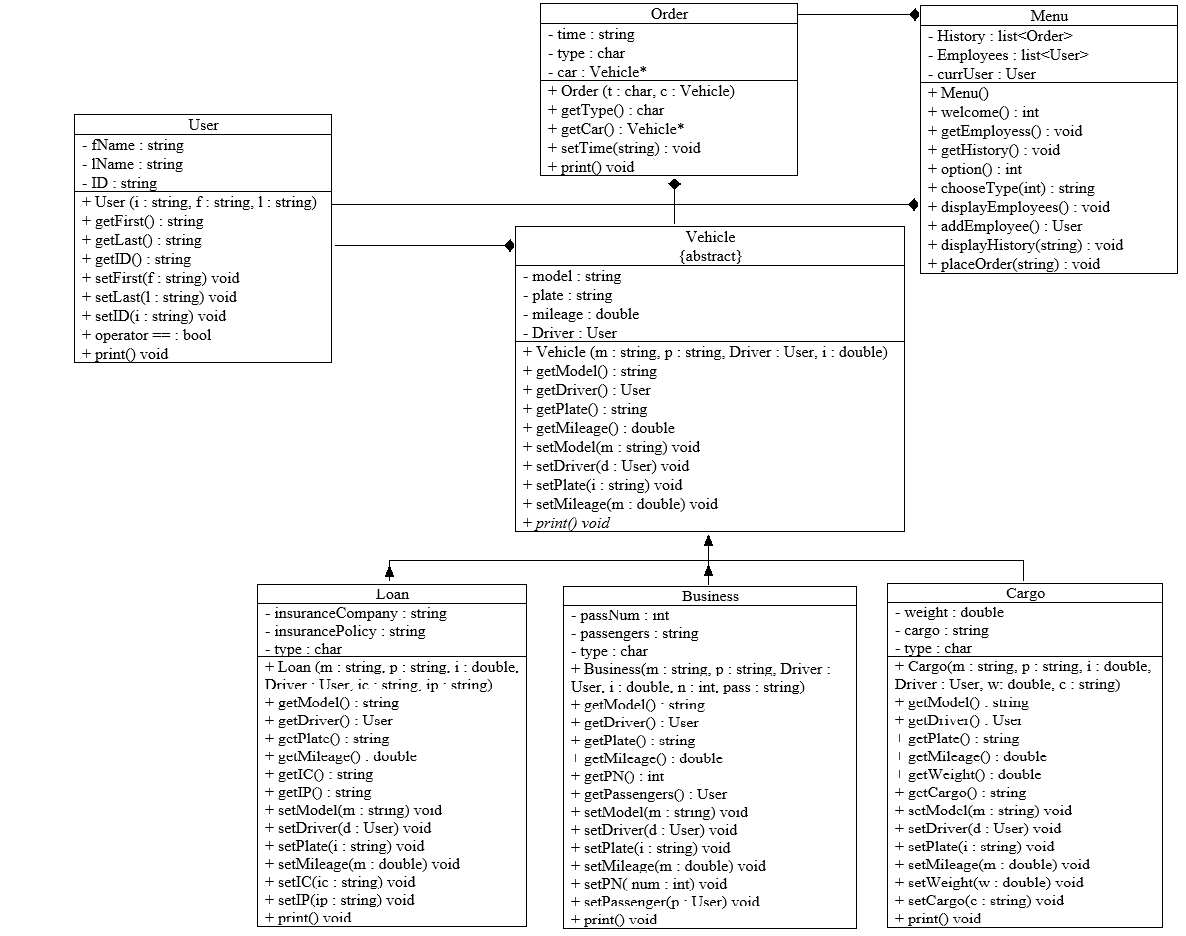
The program will run according to one of the following scenarios:



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.