TRAVEL AGENCY

DESCRIPTION OF THE PROJECT ("THE STORY")

In this project I am going to create a kind of software representing the website off a travel agency where clients can book different travel. Employee user is created by the "***SuperUser***" off the website who is basically the computer scientist of the company.

"***TravelAgency***" is the main class. It contains attributes about the basic's information and some table where each line represents an object. So, there is different table to list all the employee, all the travel and all the payment. The class contains the following methods: **showAgencyDetails**, **addSuperUser** and **bankrupt**.

"***User***" represents the people who are using the app. Once again, attributes represent the basic information for one person. User can only **updateAccount** to modify eventual error or to modify the password.

Moreover, "***User***" has several inherited classes which are "***Client***", "***Employee***" and "***SuperUser***". With this we can define some specificity for each of them according to their name. For example, Employee can **createTravel**, **removeTravel**, **updateTravel**, **createClient** and **removeClient**. "***SuperUser***" can **addEmployee** and **removeEmployee**.

"***SuperUser***" inherits from class "***Employee***" because a "***SuperUser***" is obviously an employee of the travel agency.

"***Payment***" is made to list the customer payment. So, this class contains attribute about *client_id* and all that is needed to identify each payment easily. This class can also **searchPayment**, **refundPayment**, and **addPayment**.

"***Booking***" class allows clients to book some travel according to the availability. Once again, like the "Payment" class attributes represent the basic useful information about each booking. This class allow to **bookTravel**, **updateBooking** and **deleteBooking**.

"***Travel***" class permit employee to create travel including the number of: "***Hotel***", "***Plane***" and "***Train***". This class can **create**, **delete** and **update** each "***Travel***" object. The attributes are the following: *travel_id*, *start_date*, *end_date*, *duration*, *place_available*, *price*, *hotel_number*, *plane_number*, *train_number*, *tab_hotel*, *tab_plane* and *tab_train*.

"***Hotel***", "***Train***" and "***Plane***" include in "***Travel***" tables as previously stated.

"***Hotel***" contains the following attributes: hotel_id, address, name and price.

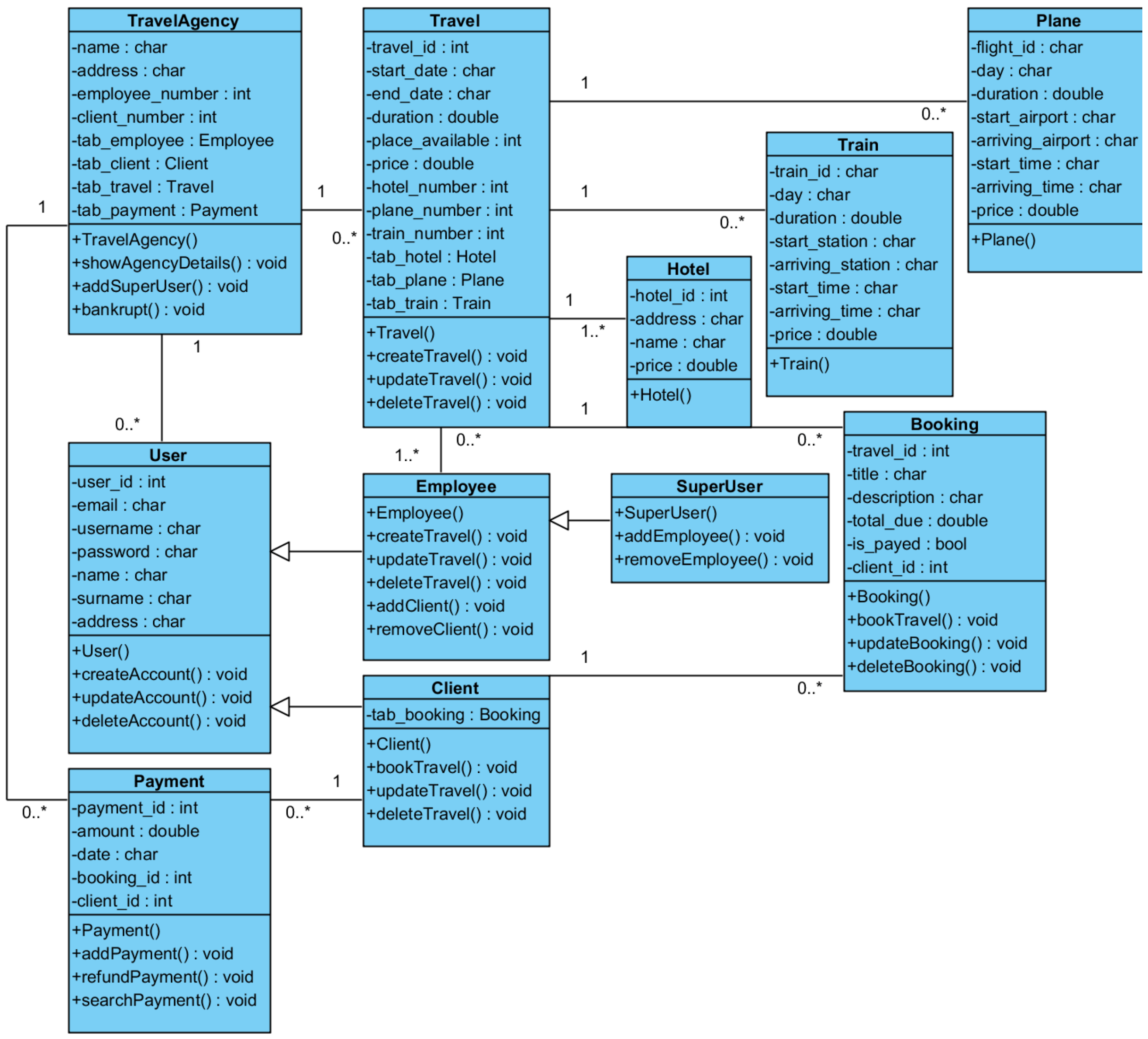
"***Train***" and "***Plane***" contain these attributes: (*train*|*plane*)*_id*, *day*, *duration*, *start_*(*station*|*airport*), *arriving_*(*station*|*airport*), *start_time*, *arriving_time* and *price*.

These four classes have several restrictions:

1. One Travel must have at least one hotel
2. An employee can't book any travel
3. If a client wishes to book, he must pay at least half the total price of the travel

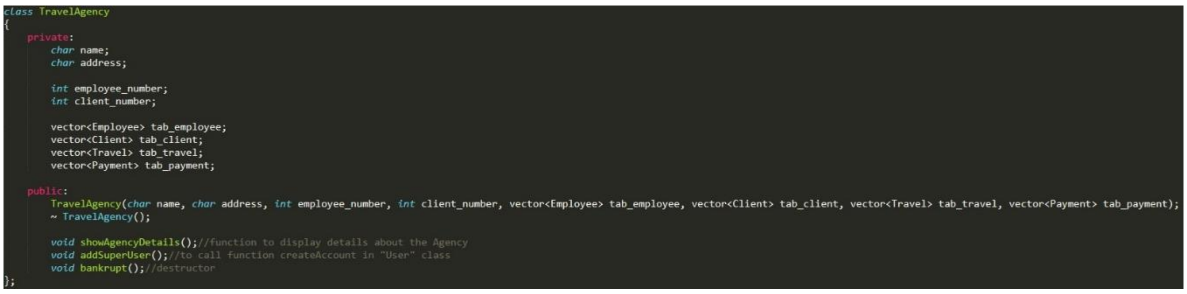
NB: All the tables will be vectors.

CASE STUDY (MEMORY MAP)



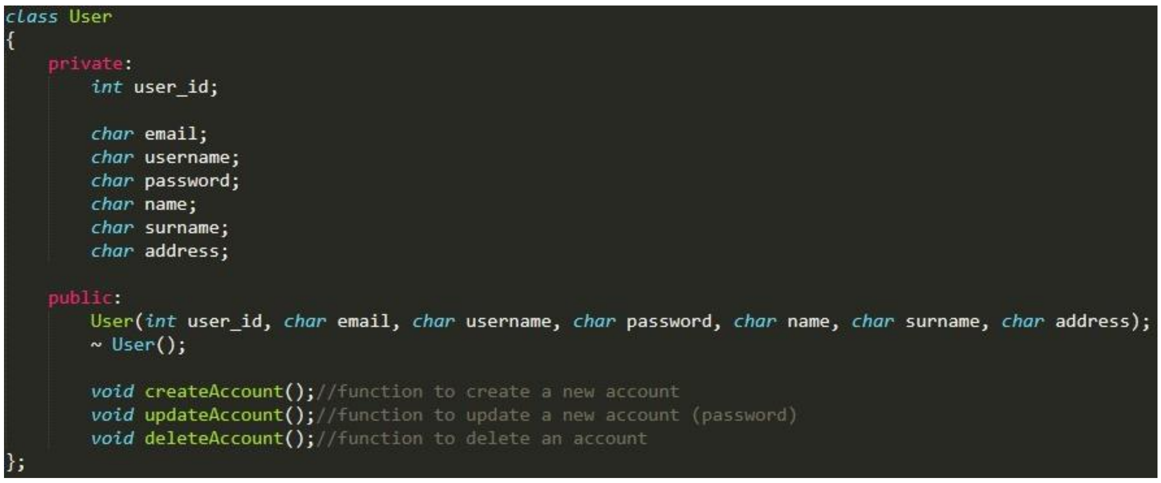
Declaration of the classes

"TravelAgency"



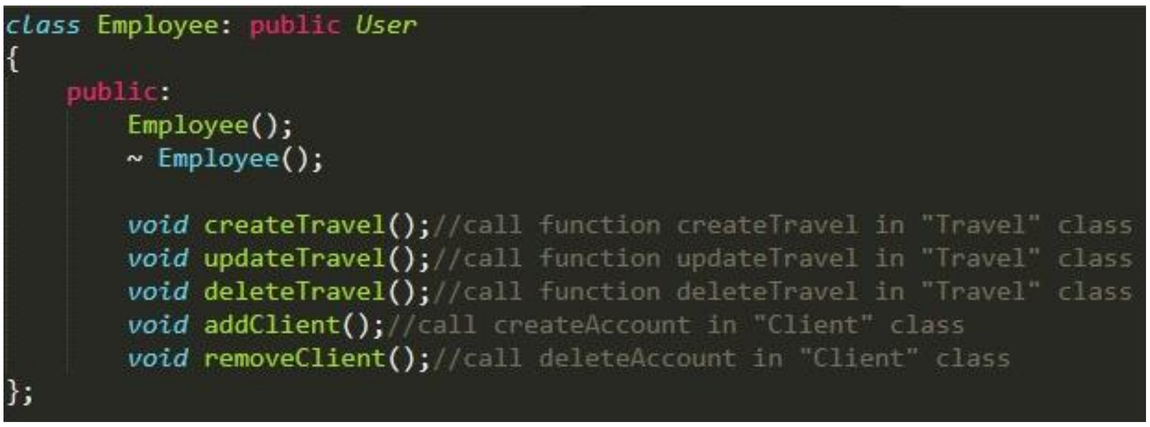
```
class TravelAgency
{
    private:
        char name;
        char address;

        int employee_number;
        int client_number;

        vector<Employee> tab_employee;
        vector<Client> tab_client;
        vector<Travel> tab_travel;
        vector<Payment> tab_payment;

    public:
        TravelAgency(char name, char address, int employee_number, int client_number, vector<Employee> tab_employee, vector<Client> tab_client, vector<Travel> tab_travel, vector<Payment> tab_payment);
        ~ TravelAgency();

        void showAgencyDetails();//function to display details about the Agency
        void addSuperUser();//to call function createAccount in "User" class
        void bankrupt();//destructor
};
```

"User"



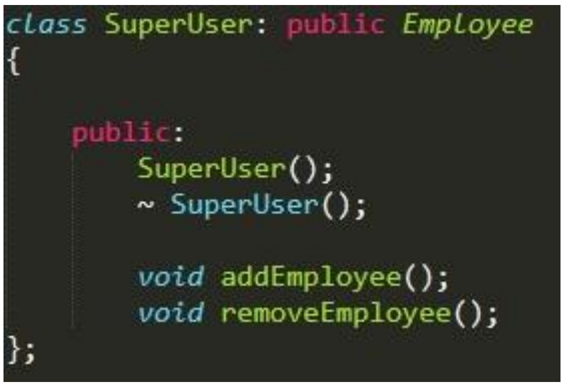
```
class User
{
    private:
        int user_id;

        char email;
        char username;
        char password;
        char name;
        char surname;
        char address;

    public:
        User(int user_id, char email, char username, char password, char name, char surname, char address);
        ~ User();

        void createAccount();//function to create a new account
        void updateAccount();//function to update a new account (password)
        void deleteAccount();//function to delete an account
};
```

"Employee"



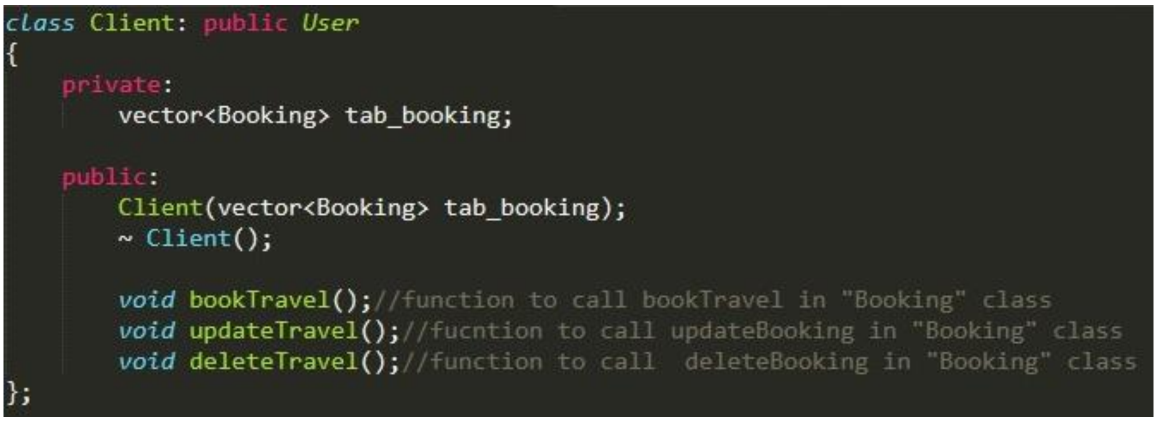
```
class Employee: public User
{
    public:
        Employee();
        ~ Employee();

        void createTravel();//call function createTravel in "Travel" class
        void updateTravel();//call function updateTravel in "Travel" class
        void deleteTravel();//call function deleteTravel in "Travel" class
        void addClient();//call createAccount in "Client" class
        void removeClient();//call deleteAccount in "Client" class
};
```

"***SuperUser***"



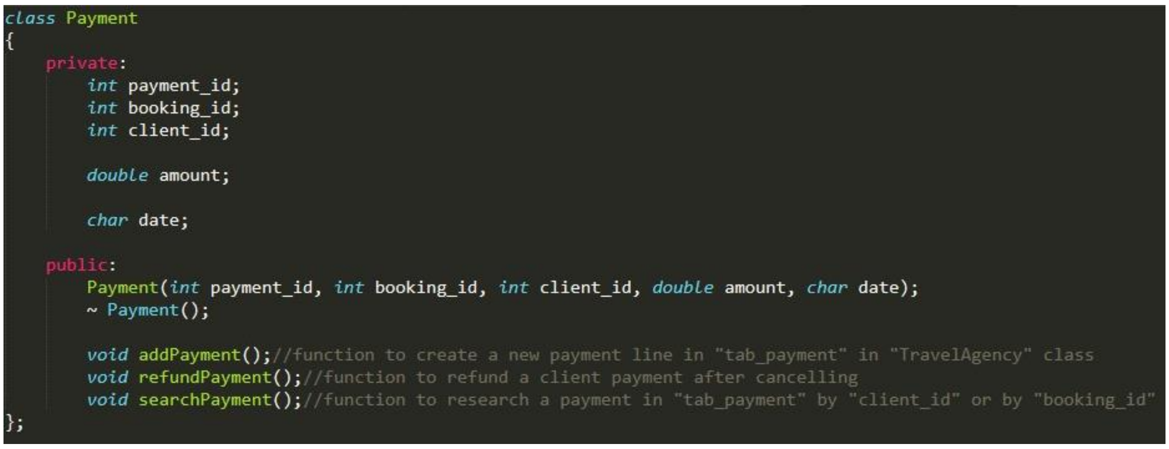
```
class SuperUser: public Employee
{

    public:
        SuperUser();
        ~ SuperUser();

        void addEmployee();
        void removeEmployee();
};
```

"***Client***"



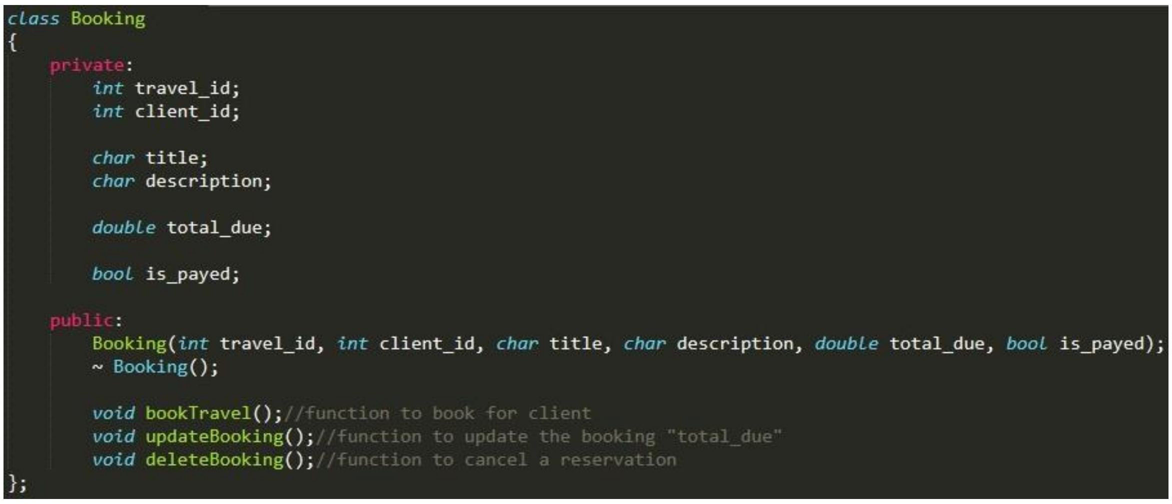
```
class Client: public User
{
    private:
        vector<Booking> tab_booking;

    public:
        Client(vector<Booking> tab_booking);
        ~ Client();

        void bookTravel();//function to call bookTravel in "Booking" class
        void updateTravel();//fucntion to call updateBooking in "Booking" class
        void deleteTravel();//function to call  deleteBooking in "Booking" class
};
```

"***Payment***"



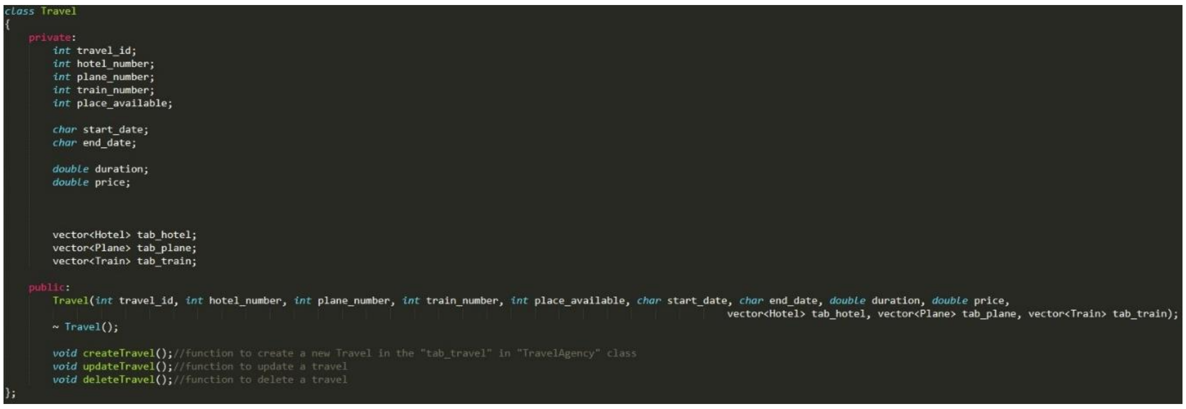
```
class Payment
{
    private:
        int payment_id;
        int booking_id;
        int client_id;

        double amount;

        char date;

    public:
        Payment(int payment_id, int booking_id, int client_id, double amount, char date);
        ~ Payment();

        void addPayment();//function to create a new payment line in "tab_payment" in "TravelAgency" class
        void refundPayment();//function to refund a client payment after cancelling
        void searchPayment();//function to research a payment in "tab_payment" by "client_id" or by "booking_id"
};
```

"Booking"



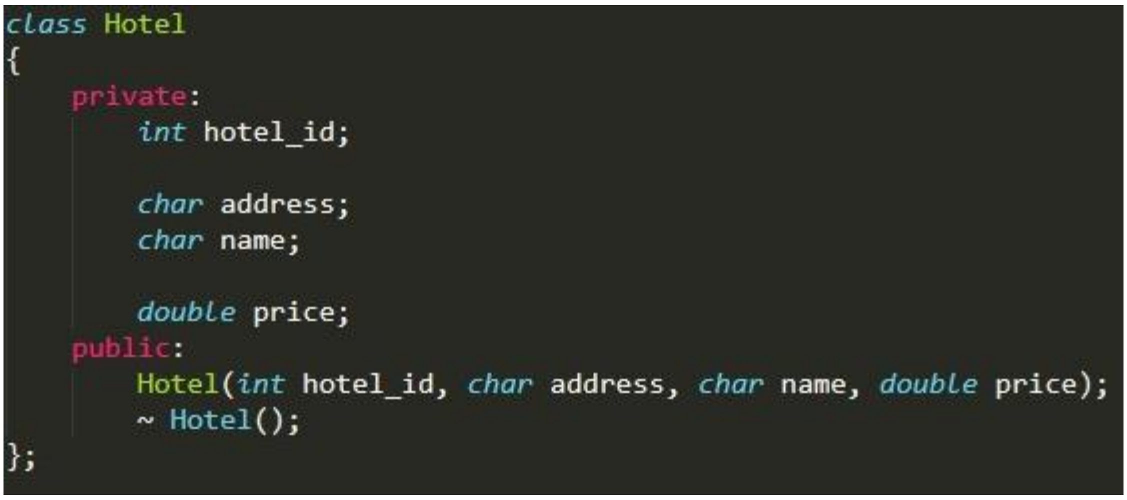
```cpp
class Booking
{
    private:
        int travel_id;
        int client_id;

        char title;
        char description;

        double total_due;

        bool is_payed;

    public:
        Booking(int travel_id, int client_id, char title, char description, double total_due, bool is_payed);
        ~ Booking();

        void bookTravel();//function to book for client
        void updateBooking();//function to update the booking "total_due"
        void deleteBooking();//function to cancel a reservation
};
```

"Travel"



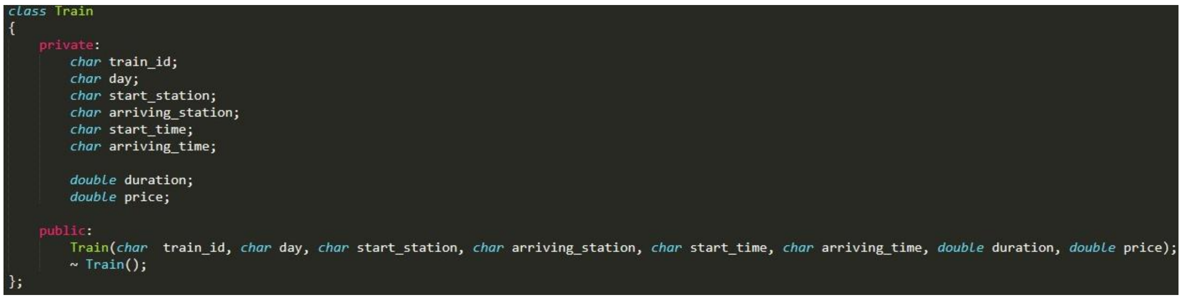
```cpp
class Travel
{
    private:
        int travel_id;
        int hotel_number;
        int plane_number;
        int train_number;
        int place_available;

        char start_date;
        char end_date;

        double duration;
        double price;



        vector<Hotel> tab_hotel;
        vector<Plane> tab_plane;
        vector<Train> tab_train;

    public:
        Travel(int travel_id, int hotel_number, int plane_number, int train_number, int place_available, char start_date, char end_date, double duration, double price,
                                                                                                          vector<Hotel> tab_hotel, vector<Plane> tab_plane, vector<Train> tab_train);
        ~ Travel();

        void createTravel();//function to create a new Travel in the "tab_travel" in "TravelAgency" class
        void updateTravel();//function to update a travel
        void deleteTravel();//function to delete a travel
};
```

"Hotel"



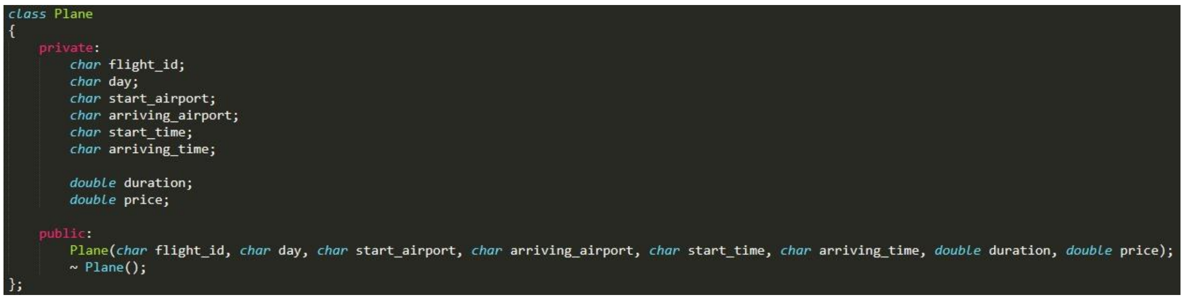
```cpp
class Hotel
{
    private:
        int hotel_id;

        char address;
        char name;

        double price;
    public:
        Hotel(int hotel_id, char address, char name, double price);
        ~ Hotel();
};
```

“Train”



```
class Train
{
    private:
        char train_id;
        char day;
        char start_station;
        char arriving_station;
        char start_time;
        char arriving_time;

        double duration;
        double price;

    public:
        Train(char  train_id, char day, char start_station, char arriving_station, char start_time, char arriving_time, double duration, double price);
        ~ Train();
};
```

“Plane”



```
class Plane
{
    private:
        char flight_id;
        char day;
        char start_airport;
        char arriving_airport;
        char start_time;
        char arriving_time;

        double duration;
        double price;

    public:
        Plane(char flight_id, char day, char start_airport, char arriving_airport, char start_time, char arriving_time, double duration, double price);
        ~ Plane();
};
```

TESTING

For the testing part I will make the following points:

1. Interact with the program
2. Create Travel ad
3. Create test function in the main to be certain that functions are doing the right action

Then with the different tests and especially with the third point I will be able to ameliorate my code. It will help me to make my program more and more efficient.