Travel Agency Model

Model Purpose:

The application’s main purpose is for the Travel Agency to be able to book flights from different airliners, for the customers.The travel agency has a travel office that maintains the customer directory.There is a Master Travel Schedule that has the flight schedules from different airliners in order to manage the customer ticket reservations.

Business Problems Addressed:

- Agency Application can add,delete or update the airliners.
- Agency Application can add,delete or update the customers.
- Agency can search for flights from different airliners and book the best available ticket, for the customer.
- Agency can also update or cancel the tickets, for the customer.
- Master Travel Schedule manages all the flight information from different airliners.
- New customers can be created using ID, name and contact details.
- New Airliners can be added to the application and then viewed . Also, new flights can be added to the specific airlines.

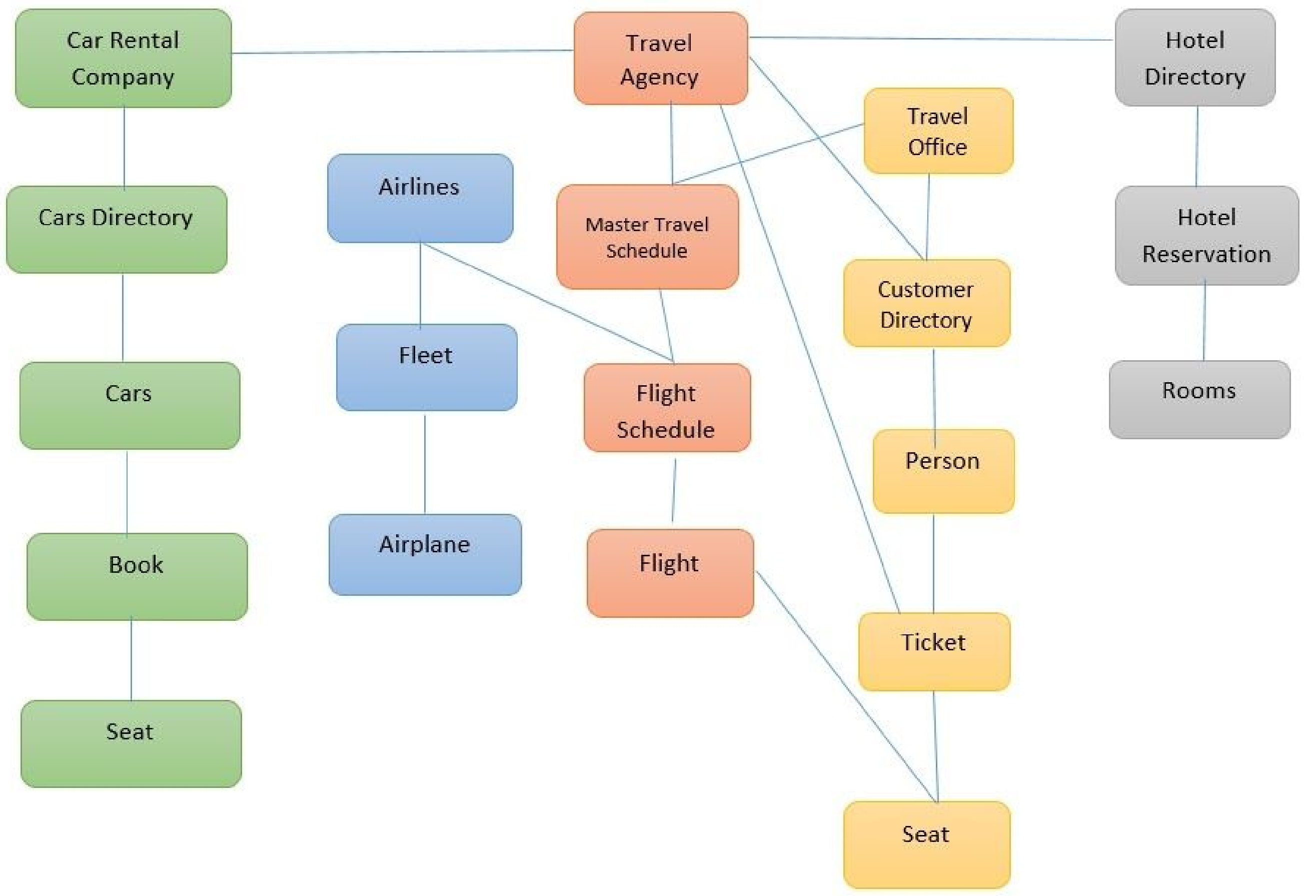
Business Rules:

- Any number of customers can be added.
- Any number of flights can be added to the airliners.
- Any number of airliners can be added.
- All the class level member variables would be defined as private.
- The class level member methods may be defined as public/private/protected as the case may be.

Design Requirements :

- User should have an option to navigate to a Panel which allows multi-field/multi-parameter search of flights. Results should be visible on a JTable.
- User should be able to reserve a seat for a Person/Customer.
- Dropdown has been used wherever possible as manual text entries can be error some.
- Exceptions have been handled.
- Data has been validated.

Object Model Diagram:



CAR RENTAL FEATURE:

- A Car Rental feature would be added to the existing application.
- **Business.CarRental package:**
- **Car.java** : Includes all the car related attributes along with its getters and setters.
- **CarDirectory.java** : Includes the necessary collections that would enable addition, deletion and updation of Cars.
- There would be **idleCarList** and **bookedCarList**, that would separate the idle cars from the booked cars.
- The Travel Office would manage the Car bookings for the Customer, as per the location of arrival, date and time of arrival. The car would be booked from the airport to the user given location.
- Car booking would be optional for the customer.

- The customer would have an option to choose the type of car that can be booked, from the list of available choices displayed.
- Only the customers who have booked the flight can book a car from the application.
- **UserInterface.CarRental** package would be created that would have three JPanels:
- **AddCarJPanel.java, ManageCarJPanel.java** and **ViewCarDetailsJPanel.java.**

HOTEL RESERVATIONS FEATURE:

- A Hotel Reservation feature would be added to the existing application.
- **Business.HotelReservation package:**
- **Hotels.java** : Includes all the hotel related attributes along with its getters and setters.
- **HotelDirectory.java** : Includes the necessary collections, that would maintaining the list of hotels, the availability and type of rooms present in the hotel and associated updation methods.
- The Travel Office would manage the Hotel reservations for the Customer, as per the location of arrival, date and time of arrival.
- Hotel reservation would be optional for the customer.
- The customer would be given an option to choose from the list of available hotels near his/her location of arrival.
- The user can choose the type of hotel rooms, the number of days for which reservation is required,the check-in and check-out time,etc.
- Only the customers who have booked the flight can reserve a hotel room, from the application.
- **UserInterface.CarRental** package would be created that would have three JPanels:
- **AddCarJPanel.java, ManageCarJPanel.java** and **ViewCarDetailsJPanel.java.**

SUPPORT FOR MULTIPLE STOPS IN A TRAVEL PLAN:

- A customer can take a halt of upto 2 days maximum after leaving departing from original location until reaching the final destination.
- A list, say **multipleStopList** would be added to the ManageCustomer.java class, that would store the multiple stops for the particular customer, based on the trip ids.
- Each subtrip would be booked similar to a normal trip and would be added to the **multipleStopList** for the user.
- In case of multiple stops, multiple **tripIds** would be associated with a single customer, and all of them would be appended to the list associated with the particular customer.