Back To Course Home

Grokking the Object Oriented Design Interview

0% completed

Search Course

Object-Oriented Design and UML

- Object-Oriented Basics
- OO Analysis and Design
- What is UML?
- Use Case Diagrams
- Class Diagram
- Sequence diagram
- Activity Diagrams

Object Oriented Design Case Studies

- Design a Library Management System
- Design a Parking Lot
- Design Amazon - Online Shopping System
- Design Stack Overflow
- Design a Movie Ticket Booking System
- Design an ATM
- Design an Airline Management System
- Design Blackjack and a Deck of Cards
- Design a Hotel Management System
- Design a Restaurant Management system
- Design Chess
- Design an Online Stock Brokerage System
- Design a Car Rental System
- Design LinkedIn
- Design Cricinfo
- Design Facebook - a social network

Appendix

- Contact us
- Other courses

Course Certificate

Mark Course as Completed

Design an Airline Management System

We'll cover the following

- System Requirements
- Use case diagram
- Class diagram
- Activity diagrams
- Code

An Airline Management System is a managerial software which targets to control all operations of an airline. Airlines provide transport services for their passengers. They carry or hire aircraft for this purpose. All operations of an airline company are controlled by their airline management system.

This system involves the scheduling of flights, air ticket reservations, flight cancellations, customer support, and staff management. Daily flights updates can also be retrieved by using the system.

System Requirements

We will focus on the following set of requirements while designing the Airline Management System:

1. Customers should be able to search for flights for a given date and source/destination airport.
2. Customers should be able to reserve a ticket for any scheduled flight. Customers can also build a multi-flight itinerary.
3. Users of the system can check flight schedules, their departure time, available seats, arrival time, and other flight details.
4. Customers can make reservations for multiple passengers under one itinerary.
5. Only the admin of the system can add new aircrafts, flights, and flight schedules. Admin can cancel any pre-scheduled flight (all stakeholders will be notified).
6. Customers can cancel their reservation and itinerary.
7. The system should be able to handle the assignment of pilots and crew members to flights.
8. The system should be able to handle payments for reservations.
9. The system should be able to send notifications to customers whenever a reservation is made/modified or there is an update for their flights.

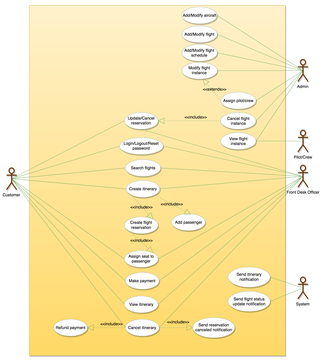
Use case diagram

We have five main Actors in our system:

- **Admin:** Responsible for adding new flights and their schedules, canceling any flight, maintaining staff-related work, etc.
- **Front desk officer:** Will be able to reserve/cancel tickets.
- **Customer:** Can view flight schedule, reserve and cancel tickets.
- **Pilot/Crew:** Can view their assigned flights and their schedules.
- **System:** Mainly responsible for sending notifications regarding itinerary changes, flight status updates, etc.

Here are the top use cases of the Airline Management System:

- **Search Flights:** To search the flight schedule to find flights for a suitable date and time.
- **Create/Modify/View reservation:** To reserve a ticket, cancel it, or view details about the flight or ticket.
- **Assign seats to passengers:** To assign seats to passengers for a flight instance with their reservation.
- **Make payment for a reservation:** To pay for the reservation.
- **Update flight schedule:** To make changes in the flight schedule, and to add or remove any flight.
- **Assign pilots and crew:** To assign pilots and crews to flights.

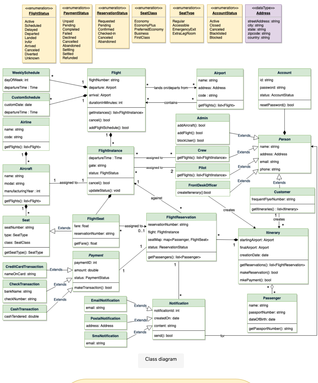


Use case diagram

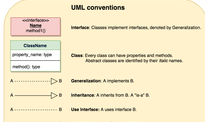
Class diagram

Here are the main classes of our Airline Management System:

- **Airline:** The main part of the organization for which this software has been designed. It has attributes like 'name' and an airline code to distinguish the airline from other airlines.
- **Airport:** Each airline operates out of different airports. Each airport has a name, address, and a unique code.
- **Aircraft:** Airlines own or hire aircraft to carry out their flights. Each aircraft has attributes like name, model, manufacturing year, etc.
- **Flight:** The main entity of the system. Each flight will have a flight number, departure and arrival airport, assigned aircraft, etc.
- **FlightInstance:** Each flight can have multiple occurrences; each occurrence will be considered a flight instance in our system. For example, if a British Airways flight from London to Tokyo (flight number: BA212) occurs twice a week, each of these occurrences will be considered a separate flight instance in our system.
- **WeeklySchedule and CustomSchedule:** Flights can have multiple schedules and each schedule will create a flight instance.
- **FlightReservation:** A reservation is made against a flight instance and has attributes like a unique reservation number, list of passengers and their assigned seats, reservation status, etc.
- **Itinerary:** An itinerary can have multiple flights.
- **FlightSeat:** This class will represent all seats of an aircraft assigned to a specific flight instance. All reservations of this flight instance will assign seats to passengers through this class.
- **Payment:** Will be responsible for collecting payments from customers.
- **Notification:** This class will be responsible for sending notifications for flight reservations, flight status update, etc.

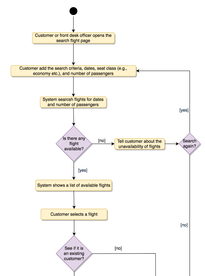


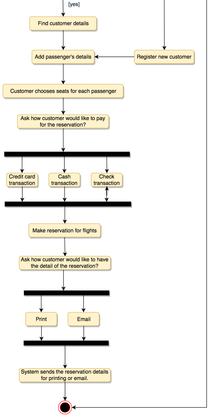
Class diagram



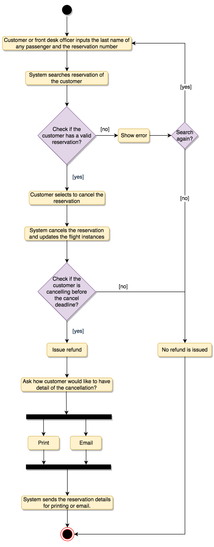
Activity diagrams

- **Reserve a ticket:** Any customer can perform this activity. Here are the steps to reserve a ticket:





- **Cancel a reservation:** Any customer can perform this activity. Here are the set of steps to cancel a reservation:



Code

Here is the code for major classes.

Enums and Constants: Here are the required enums, data types, and constants:

Java | Python

```java
public enum FlightStatus{
  ACTIVE,
  SCHEDULED,
  DELAYED,
  DEPARTED,
  LANDED,
  IN_AIR,
  ARRIVED,
  CANCELLED,
  DIVERTED,
  UNKNOWN
}

public enum PaymentStatus{
  UNPAID,
  PENDING,
  COMPLETED,
  FILLED,
  DECLINED,
  CANCELLED,
  ABANDONED,
  SETTLING,
  SETTLED,
  REFUNDED
}

public enum ReservationStatus{
  REQUESTED,
```

Account, Person, Customer and Passenger: These classes represent the different people that interact with our system:

Java | Python

```java
// For simplicity, we are not defining getter and setter functions. The reader can
// assume that all class attributes are private and accessed through their respective
// public getter method and modified only through their public setter method.

public class Account {
  private String id;
  private String password;
  private AccountStatus status;

  public boolean resetPassword();
}

public abstract class Person {
  private String name;
  private Address address;
  private String email;
  private String phone;

  private Account account;
}

public class Customer extends Person {
  private String frequentFlyerNumber;

  public List<Itinerary> getItineraries();
}

public class Passenger {
```

Airport, Aircraft, Seat and FlightSeat: These classes represent the top-level classes of the system:

Java | Python

```java
public class Airport {
  private String name;
  private Address address;
  private String code;

  public List<Flight> getFlights();
}

public class Aircraft {
  private String name;
  private String model;
  private int manufacturingYear;
  private List<Seat> seats;

  public List<FlightInstance> getFlights();
}

public class Seat {
  private String seatNumber;
  private SeatType type;
  private SeatClass _class;
}

public class FlightSeat extends Seat {
  private double fare;
  public double getFare();
}
```

Flight Schedule classes, Flight, FlightInstance, FlightReservation, Itinerary: Here are the classes related to flights and reservations:

Java | Python

```java
public class WeeklySchedule {
  private int dayOfWeek;
  private Time departureTime;
}

public class CustomSchedule {
  private Date customDate;
  private Time departureTime;
}

public class Flight {
  private String flightNumber;
  private Airport departure;
  private Airport arrival;
  private int durationInMinutes;

  private List<WeeklySchedules> weeklySchedules;
  private List<CustomSchedules> customSchedules;
  private List<FlightInstance> flightInstances;
}

public class FlightInstance {
  private Date departureTime;
  private String gate;
  private FlightStatus status;
  private Aircraft aircraft;

  public bool cancel();
```