# Getting Ready: The Airline Management System

Understand the airline management system problem and learn the questions to simplify this problem further.

**We'll cover the following**

* [Problem definition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Expectations-from-the-interviewee)
  + [Flight reservation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Flight-reservation)
  + [Payment handling](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Payment-handling)
  + [Price variance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Price-variance)
  + [Flight cancellation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Flight-cancellation)
* [Design approach](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-approach)
* [Design pattern](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-pattern)

## Problem definition

An **airline management system** is a software used to manage all activities of the airline system efficiently. Nowadays, every airline has a management system to digitize the process of scheduling flights, managing staff, ticket reservations, and performing other necessary airline management tasks. Moreover, the system keeps track of the number of aircraft, pilots, and their availability at airports. The system provides customers the complete flight information so they can view the available flights and schedule them online. Similarly, the admin can manage all the airline activities with the help of this system. Therefore, the airline management system facilitates its customers as well as admins and controls all the operations of the airline company.

Airline management system

## Expectations from the interviewee

There are several components present in the airline management system, each with specific constraints and requirements. The following provides an overview of some of the main things that the interviewer will want to hear you discuss in more detail, during the interview.

### Flight reservation

Flight reservation is an essential part of the airline management system. The system has to ensure that no two people should be mapped to the same seat. The interviewer expects you to ask questions to identify how the system will complete flight reservations:

* How will the system ensure that multiple users do not have the same seat on the aircraft?
* Can one itinerary reserve multiple flights?
* Can the customer reserve the whole aircraft?

### Payment handling

One of the airline management system's most significant attributes is its payment structure for its customers. This can vary, so the interviewer would expect you to ask the questions listed below:

* What payment methods can the customer use, for example, credit card or cash?
* How does the customer make the payment? Does the customer pay online or in persone?
* Will the customer be able to pay in advance for a flight booking, or is a just-in-time (JIT) payment method available?

### Price variance

Now that we’ve discussed the payment methods of the airline management system, let’s ask the interviewer about the pricing model. You may ask the questions listed below:

* Is the price set manually, or does the system calculate the price for each flight?
* Does every seat has the same price, or is it calculated based on the seat type?
* Does the weekdays and weekends affect the price of the flight?
* Is the price of the flight affected by an increase in demand?
* How does the duration of the flight affect the payment?

### Flight cancellation

In the airline management system, the customer may want to cancel the flight after booking it. Therefore, you can ask the following questions:

* Can the customer cancel a flight?
* What is the time limit to cancel the flight?
* Which type of users are allowed to request a flight cancellation?

## Design approach

We’ll design this airline management system using the bottom-up approach. For this purpose, we will follow the steps below:

* Identify and design the smallest components first, including seat, flight, etc.
* Use these small components to design bigger components, including airport, aircraft that can be composed of multiple seats, etc.
* Repeat the steps above until we design the whole airline management system.

## Design pattern

It is always a good practice to discuss the design patterns that an airline management system falls under, during the interview. Stating the design patterns will give the interviewer a positive impression and shows that the interviewee is well-versed in the advanced concepts of object-oriented design.

The following design patterns can be used to design the airline management system:

* The Singleton design pattern
* The Observer design pattern

Let’s explore the requirements of the airline management system in the next lesson.

# Requirements for the Airline Management System

Learn about all requirements of the airline management system.

**We'll cover the following**

* [Requirements collection](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Requirements-collection)

In this lesson, we’ll list the requirements of the airline management system. This is a very crucial step, because requirements define the scope of a problem. Therefore, getting them right from the interviewer and understanding them well will make the design of the rest of the system smooth and easy.

We’ll use the notational convention to identify each requirement with a unique label, "Rn" where "R" is short for Requirement and "n" is a natural number.

## Requirements collection

The requirements for the airline management system are defined below:

**R1:** A customer should be able to search for flights by the date, departure, and destination airport.

**R2:** A customer should be able to reserve tickets for available flights. Customers should also be able to book multiple flights at once.

**R3:** The customer should be allowed to book multiple seats for a single flight.

**R4:** The system should allow the customer to check flight details, such as available seats, flight schedules, and departure/arrival times.

**R5:** The admin should be able to add new flights. The admin should be able to update or cancel scheduled flights.

**R6:** An airline should be able to own multiple aircrafts. The admin should be able to add these aircrafts to the system.

**R7:** An airline should be able to operate its flights from different airports.

**R8:** The admin should be able to assign pilots and crew members to flights effectively.

**R9:** The customer should be able to make payments against their flight reservations.

**R10:** The customer should be able to cancel their previous reservations.

**R11:** The front desk officer should be able to reserve tickets, create itineraries, and make flight payments for the customer.

**R12:** The flight crew should be able to view the schedule for their assigned flights.

**R13:** The system should send the customer a notification whenever a reservation has been made or canceled or when there is an update for their flight.

We've identified our requirements for the problem. In the next lesson, we will define different use cases of the airline management system.

# Use Case Diagram for the Airline Management System

Learn how to define use cases and create the corresponding use case diagram for the airline management system.

**We'll cover the following**

* [System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#System)
* [Actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actors)
  + [Primary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Primary-actors)
  + [Secondary actors](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Secondary-actors)
* [Use cases](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-cases)
  + [Customer/front desk officer](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Customer/front-desk-officer)
  + [Admin](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Admin)
  + [System](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#System)
  + [Crew](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Crew)
* [Relationships](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationships)
  + [Associations](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Associations)
  + [Include](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Include)
  + [Extend](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Extend)
* [Use case diagram](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Use-case-diagram)

Let's build the use case diagram of the airline management system and understand the relationship between its different components.

First, we’ll define the different elements of our airline, followed by the complete use case diagram of the system.

## System

Our system is an "airline."

## Actors

Now, we’ll define the main actors of the airline management system.

### Primary actors

* **Customer:** The customer is the airline's primary actor who can search for flights, create an itinerary, make a payment, and update or cancel the flight reservation.
* **Front desk officer:** This actor can perform all the actions that the customer can. The front desk officer can create an itinerary, make payment, update or cancel the flight reservation on behalf of customers, assign them seats, and search for flights.
* **Admin:** The admin is in charge of performing numerous operations, like adding aircraft to the system, adding or modifying flights, flight instances, and their schedules, canceling any flight, and assigning crew to the flight.

### Secondary actors

* **System:** This actor is responsible for sending notifications for flight status updates, itinerary changes, and reservation cancellations.
* **Crew:** This actor can view the schedules of the assigned flights.

## Use cases

In this section, we’ll define the use cases for an airline management system. We have listed the use cases according to their respective interactions with a particular actor.

**Note:** You’ll see some use cases occurring multiple times because they are shared among different actors in the system.

### Customer/front desk officer

* **Update/cancel reservation:**To update or cancel a flight reservation of the customer
* **Login/logout:**To log in or log out of the airline system
* **Reset password:** To reset the password of the account
* **Create itinerary:**To create an itinerary for the customer
* **Assign seat:**To assign a seat to the passenger for the flight
* **Search flights:** To search for flights in the airline management system
* **Make payment:**To pay for the itinerary or flight reservation
* **View itinerary:**To view the details of an itinerary
* **Cancel itinerary:**To cancel the itinerary for the customer

### Admin

* **Add aircraft:** To add a new aircraft to the airline management system
* **Add/modify flight:** To add a new flight or modify it
* **Assign crew:**To assign crew to the flight instance
* **Block/unblock user:** To block or unblock a user in an airline management system
* **Cancel flight:**To cancel the instance of a flight
* **Search flights:** To search for flights in the airline management system

### System

* **New itinerary notification:**To send a notification of a new itinerary to the customer
* **Flight status notification:**To send the flight status update notification to the customer
* **Reservation cancel notification:** To send a reservation cancellation notification to the customer

### Crew

* **View flight schedule:** To view the schedule of the assigned flights

## Relationships

We describe the relationships between and among actors and their use cases in this section.

### Associations

The below table shows the association relationship between actors and their use cases.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Customer** | **Front desk officer** | **Admin** | **System** | **Crew** |
| Cancel reservation | Cancel reservation | Add aircraft | New itinerary notification | View flight schedule |
| Login/logout | Login/logout | Add/modify flight | Reservation cancel notification |  |
| Reset password | Reset password | Block/unblock users | Flight status notification |
| Create itinerary | Create itinerary | Assign crew |  |
| Assign seat | Assign seat | Cancel flight |
| Search flights | Search flights | Search flights |
| Make payment | Make payment |  |
| View itinerary | View itinerary |
| Cancel itinerary | Cancel itinerary |

### Include

* To create an itinerary, one or more flight reservations should be created to add the passengers to the itinerary. Therefore, the “Create itinerary” use case has an include relationship with both “Create flight reservation” and “Add passenger” use cases.
* While creating a flight reservation, the seat should be assigned to the customer. Therefore, the “Create flight reservation” use case has an include relationship with the “Assign seat” use case.
* Whenever the flight is canceled, its related flight reservations should also be canceled. Hence the “Cancel flight” use case has an include relationship with the “Cancel reservation” use case.
* When an itinerary is created, a notification is sent to the customer. Therefore, the “Create itinerary” use case has an include relationship with the “New itinerary notification” use case.
* If an itinerary is canceled, the payment will be refunded, and a reservation cancellation notification will be sent to the customer. Hence, the “Cancel itinerary” use case has an include relationship with both “Refund payment” and “Reservation cancellation notification” use cases.
* When the flight is canceled, the payment will be refunded, and a reservation cancellation notification will be sent to the customer. Hence, the “Cancel reservation” use case has an include relationship with both “Refund payment” and “Reservation cancellation notification” use cases.

### Extend

* When modifying the flight instance, the admin can assign the crew to the flight instance. Therefore, the “Modify flight instance” use case has an extend relationship with the “Assign crew” use case.

## Use case diagram

Here’s the use case diagram of the airline management system:

A diagram of a flight

Description automatically generated

# Class Diagram for the Airline Management System

Understand how to create a class diagram for an airline management system using the bottom-up approach.

**We'll cover the following**

* [Components of an airline management system](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Components-of-an-airline-management-system)
  + [Account](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account)
  + [Person](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Person)
  + [Airline, airport, and aircraft](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Airline,-airport,-and-aircraft)
  + [Seat and flight seat](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Seat-and-flight-seat)
  + [Flight and flight instance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Flight-and-flight-instance)
  + [Flight reservation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Flight-reservation)
  + [Itinerary and passenger](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Itinerary-and-passenger)
  + [Search and catalog](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Search-and-catalog)
  + [Payment](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Payment)
  + [Notification](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Notification)
  + [Enumerations](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Enumerations)
  + [Custom data type](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Custom-data-type)
* [Relationship between classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Relationship-between-classes)
  + [Association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Association)
    - [One-way association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#One-way-association)
    - [Two-way association](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Two-way-association)
  + [Aggregation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Aggregation)
  + [Composition](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Composition)
  + [Inheritance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Inheritance)
* [Class diagram of the airline management system](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Class-diagram-of-the-airline-management-system)
* [Design pattern](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Design-pattern)

In the class diagram, we will first design and create the classes, abstract classes, and interfaces for the system. Then, we’ll identify the relationship between classes in accordance with all the requirements of the airline management system.

## Components of an airline management system

In this section, we’ll define the classes for an airline management system. As mentioned earlier, we will design the airline management system using a bottom-up approach. We will create the classes of small components first. Next, we will integrate these components and create the class diagram for the entire airline management system.

### Account

The Account class identifies the username and ID of an airline management system user. . The class definition is represented below:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a flight reservation

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## Relationship between classes

Now, we’ll discuss the relationships between the classes we have defined above in our airline management system.

### Association

The class diagram has the following association relationships:

#### One-way association

* Both FrontDeskOfficer and Customer classes have a one-way association with the Itinerary class.
* The FlightReservation class has a one-way association with the FlightInstance, Payment, and Notification classes.
* The FlightSeat class has a one-way association with the FlightReservation class.
* The Person class has a one-way association with the Search interface.

#### Two-way association

* The Airport class has a two-way association with the Flight class.
* Both the Aircraft and Crew classes have a two-way association with the FlightInstance class.
* The Airline class has a two-way association with the Crew class.

A screenshot of a diagram

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a diagram

Description automatically generated

A diagram of a flight management system

Description automatically generated

## Design pattern

The Airline class follows the Singleton design pattern because there will only be a single instance of the Airline class as it is the main organizational class.

The airline management system can also implement the Observer design pattern because all the passengers flying in a particular flight instance must be updated about any changes in their flight schedule. The passengers here will act as subscribers and be notified whenever the flight status changes.

We have successfully designed the class diagram for the airline management system. Let’s see how we can construct the sequence diagram for the system in our next lesson.

# Sequence Diagram for the Airline Management System

Create a sequence diagram for reserving a flight in the airline management system and solve a challenge.

**We'll cover the following**

* [Reserve a flight](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Reserve-a-flight)
* [Sequence challenge: Cancel a reservation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Sequence-challenge:-Cancel-a-reservation)

Sequence diagrams are a great way to understand the interactions between different entities and objects in the system. There can be different sequence diagrams that we can create for our airline management system. In this lesson, we will create sequence diagrams for the following two interactions:

* **Reserve a flight:** The customer reserves a flight online.
* **Sequence challenge:** The customer cancels a reservation.

## Reserve a flight

The sequence diagram for reserving a flight should have the following actors and objects that will interact with each other:

* **Actor:** Customer
* **Objects:** SearchCatalog, FlightReservation, FlightSeat, and Payment
* System

Here’re the steps in the reserve flight interaction:

1. The customer searches for flights flying from an airport on a particular date.
2. The catalog returns a list of flights that satisfy the search query.
3. The customer selects a flight.
4. If seats are available:
   1. The system requests the customer to select seats.
   2. The customer selects a seat.
   3. The system requests to add passenger details.
   4. The customer adds passenger details.
   5. The system creates a reservation for the customer.
   6. Payment is requested against the reservation.
   7. The customer initiates a transaction, and the payment is processed.
   8. If the payment is successful:
      1. The customer and system are informed that the payment is successful.
      2. The system updates the seat status to booked.
      3. The system updates the reservation status to confirmed.
      4. The reservation details are sent to the customer.
   9. Else, if the payment is unsuccessful:
      1. The customer is informed that the payment has failed.
5. Else, if seats are unavailable:
   1. The customer is informed that no seats are available.

**Note:** We assume that the customer performs a valid search that will result in a list of flights.

Based on the order above, the sequence diagram for reserving a flight in the airline management system is given below.

A screenshot of a computer

Description automatically generated

A screenshot of a diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

A screenshot of a diagram

Description automatically generated

# Activity Diagram for the Airline Management System

Create some activity diagrams for the airline management system.

**We'll cover the following**

* [Create an itinerary](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Create-an-itinerary)
  + [States](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#States)
  + [Actions](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Actions)
* [Activity challenge: The user receives confirmation notification after payment](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Activity-challenge:-The-user-receives-confirmation-notification-after-payment)

Activity diagrams are a great way to visualize the flow of messages from one activity to the other in the system. There can be different activity diagrams that we can create for the airline management system. In this lesson, we will create activity diagrams for the following two activities:

* Creating an itinerary
* **Activity challenge:**The user receives a confirmation notification after making the payment.

## Create an itinerary

The states and actions that will be involved in this activity diagram are provided below.

### States

**Initial state:**The customer chooses an option to create the itinerary.

**Final state:** A customer books their ticket.

### Actions

The customer selects an option to create an itinerary online themselves or through the front desk officer. The customer then selects their flights. Finally, the customer books their ticket.

A screenshot of a diagram

Description automatically generated

A diagram of a diagram

Description automatically generated

**Note:**If you’re unsure, click the “Show Solution” button to check the correct answer.

A screenshot of a diagram

Description automatically generated

# Code for the Airline Management System

Write the object-oriented code to implement the design of the airline management system problem.

**We'll cover the following**

* [The airline management system classes](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#The-airline-management-system-classes)
  + [Constants](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Constants)
  + [Account and passenger](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Account-and-passenger)
  + [Person](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Person)
  + [Seat and flight seat](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Seat-and-flight-seat)
  + [Flight and flight instance](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Flight-and-flight-instance)
  + [Itinerary and flight reservation](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Itinerary-and-flight-reservation)
  + [Payment](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Payment)
  + [Notification](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Notification)
  + [Search and catalog](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Search-and-catalog)
  + [Airport, aircraft, and airline](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Airport,-aircraft,-and-airline)
* [Wrapping up](https://www.educative.io/courses/grokking-the-low-level-design-interview-using-ood-principles/getting-ready-the-amazon-online-shopping-system#Wrapping-up)

We’ve reviewed different aspects of the airline management system and observed the attributes attached to the problem using various UML diagrams. Let’s explore the more practical side of things, where we will work on implementing the airline management system using multiple languages. This is usually the last step in an object-oriented design interview process.

We have chosen the following languages to write the skeleton code of the different classes present in the airline management system:

* Java
* C#
* Python
* C++
* JavaScript

## The airline management system classes

In this section, we’ll provide the skeleton code of the classes designed in the class diagram lesson.

**Note:**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 methods and modified only through their public methods function.

### Constants

The following code provides the definition of the various enums and custom data types being used in the airline management system:

**Note:** JavaScript does not support enumerations, so we will be using the Object.freeze() method as an alternative that freezes an object and prevents further modifications.

public class Address {

private int zipCode;

private String streetAddress;

private String city;

private String state;

private String country;

}

enum AccountStatus {

ACTIVE,

DISABLED,

CLOSED,

BLOCKED

}

enum SeatStatus {

AVAILABLE,

BOOKED,

CHANCE

}

enum SeatType {

REGULAR,

ACCESSIBLE,

EMERGENCY\_EXIT,

EXTRA\_LEG\_ROOM

}

enum SeatClass {

ECONOMY,

ECONOMY\_PLUS,

BUSINESS,

FIRST\_CLASS

}

enum FlightStatus {

ACTIVE,

SCHEDULED,

DELAYED,

LANDED,

DEPARTED,

CANCELED,

DIVERTED,

UNKNOWN

}

enum ReservationStatus {

REQUESTED,

PENDING,

CONFIRMED,

CHECKED\_IN,

CANCELED

}

enum PaymentStatus {

PENDING,

COMPLETED,

FAILED,

DECLINED,

CANCELED,

REFUNDED

}

A screenshot of a computer

Description automatically generated

Person is an abstract class that represents the various people or actors that can interact with the system. There are four types of persons: Admin, Crew, FrontDeskOfficer, and Customer. The implementation of the mentioned classes is shown below:

Java

C#

Python

public abstract class Person {

private String name;

private Address address;

private String email;

private String phone;

private Account account;

}

public class Admin extends Person {

public boolean addAircraft(Aircraft aircraft);

public boolean addFlight(Flight flight);

public boolean cancelFlight(Flight flight);

public boolean assignCrew(Flight flight);

public boolean blockUser(User user);

public boolean unblockUser(User user);

}

public class Crew extends Person {

public List<FlightInstance> viewSchedule();

}

public class FrontDeskOfficer extends Person {

public List<Itinerary> viewItinerary();

public boolean createItinerary();

public boolean createReservation();

public boolean assignSeat();

public boolean makePayment();

}

public class Customer extends Person {

private int customerId;

public List<Itinerary> viewItinerary();

public boolean createItinerary();

public boolean createReservation();

public boolean assignSeat();

public boolean makePayment();

}

A screenshot of a computer

Description automatically generated

public class Flight {

private String flightNo;

private int durationMin;

private Airport departure;

private Airport arrival;

private List<FlightInstance> instances;

}

public class FlightInstance {

private Flight flight;

private Date departureTime;

private String gate;

private FlightStatus status;

private Aircraft aircraft;

private List<FlightSeat> seats;

}

A screenshot of a computer program

Description automatically generated

### Payment

The Payment class is another abstract class with two child classes: Cash and CreditCard. This takes the PaymentStatus enum to keep track of the payment status. The definition of this class is provided below:

public abstract class Payment {

private int paymentId;

private double amount;

private PaymentStatus status;

private Date timestamp;

public abstract boolean makePayment();

}

public class Cash extends Payment {

public boolean makePayment() {

// functionality

}

}

public class CreditCard extends Payment {

private String nameOnCard;

private String cardNumber;

public boolean makePayment() {

// functionality

}

}

### Notification

The Notification class is another abstract class responsible for sending notifications with two child classes: SMSNotification and EmailNotification . The implementation of this class is shown below:

public abstract class Notification {

private int notificationId;

private Date createdOn;

private String content;

public abstract void sendNotification(Account account);

}

class SmsNotification extends Notification {

public void sendNotification(Account account) {

// functionality

}

}

class EmailNotification extends Notification {

public void sendNotification(Account account) {

// functionality

}

}

### Search and catalog

The SearchCatalog class contains the flight instance information and implements the Search interface class to enable the search functionality based on the criteria. Both classes are defined below:

public interface Search {

// Interface method (does not have a body)

public List<FlightInstance> searchFlight(Airport source, Airport dest, Date arrival, Date departure);

}

public class SearchCatalog implements Search {

private HashMap<Quartet<Airport, Airport, Date, Date>, List<FlightInstance>> flights;

public List<FlightInstance> searchFlight(Airport source, Airport dest, Date arrival, Date departure) {

// functionality

}

}

### Airport, aircraft, and airline

This section contains classes like Airport, Aircraft, and Airline that make up the infrastructure of our airline management system. Here, Airline is a Singleton class. The definition of these classes is given below:

public class Airport {

private String name;

private String code;

private Address address;

private List<Flight> flights;

}

public class Aircraft {

private String name;

private String code;

private String model;

private int seatCapacity;

private List<Seat> seats;

}

public class Airline {

private String name;

private String code;

private List<Flight> flights;

private List<Aircraft> aircrafts;

private List<Crew> crew;

// The Airline is a singleton class that ensures it will have only one active instance at a time

private static Airline airline = null;

// Created a static method to access the singleton instance of Airline class

public static Airline getInstance() {

if (airline == null) {

airline = new Airline();

}

return airline;

}

}

**Wrapping up**

We've explored the complete design of the airline management system in this chapter. We've looked at how a basic airline management system can be visualized using various UML diagrams and designed using object-oriented principles and design patterns.