.

# Getting Ready: Movie Ticket Booking System

Understand the movie ticket booking system problem and learn the questions to simplify this problem.

**We'll cover the following**

* [Problem definition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)
* [Expectations from the interviewee](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Expectations-from-the-interviewee)
  + [Seat selection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Seat-selection)
  + [Payment handling](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Payment-handling)
  + [Price variance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Price-variance)
  + [Duplication](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Duplication)
* [Design approach](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-approach)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)

## Problem definition[#](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Problem-definition)

The **movie ticket booking system** is a system that allows customers to book shows for their favorite movies. The booking system has information on all cinemas present in different cities and the hall information for each cinema. There are numerous movies stored in the system database. Each movie can have multiple shows playing in a particular cinema. The customer can search and select a movie for which they wish to book a show. The system displays a seating layout that identifies booked and available seats so that the customer can choose to reserve available seats. Once the customer completes the payment for the booking, the seat booking is confirmed and the customer receives an email notification.

Customers lining up to book tickets for a show

## Expectations from the interviewee

Numerous components are present in a typical movie ticket booking system, each with specific constraints and requirements placed on them. The following provides an overview of some of the main expectations that the interviewer will want to hear you discuss in more detail during the interview.

### Seat selection

Selecting a seat is an essential part of the movie ticket booking system. The system has to make sure that no two people can book the same seat. The interviewer expects you to ask questions to identify how the system will work in these situations:

* How will the system make sure that multiple users do not book the same seat?
* Will there be a timeout session that reserves seats temporarily? Will the system use a first come, first serve algorithm?
* Will there be transaction locks involved in the system?

### Payment handling

One of the most significant attributes of the movie ticket booking system is the payment structure that it provides to 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 is the payment performed? Does the customer pay themselves online or through a ticket agent on the location?

### Price variance

We touched upon the payment methods of the movie ticket booking system, now, the pricing model needs to be clarified by the interviewer, and therefore you may ask questions like these:

* How will the price of the booking be calculated? Will it vary based on the popularity of the show?
* Does the seat type affect the pricing?
* Will there be discount coupon codes?

### Duplication

There will be many duplicate instances in our system. The interviewer expects you to ask questions like these:

* How are we handling these instances, such as the same cinema having multiple cinema halls showing different movies simultaneously?
* Is the same movie being shown at different times in the same cinema/hall?

## Design approach

We’ll design this movie ticket booking system using the bottom-up design approach. For this purpose, we’ll follow the steps below:

* Identify and design the smallest components first, such as a cinema hall.
* Use these small components to design bigger components, such as building a cinema that can be composed of multiple halls.
* Repeat the above steps until we design the whole system.

## Design pattern

It is always a good practice to discuss the design patterns that a movie ticket booking 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 pattern can be used to design the movie ticket booking system:

* Strategy design pattern

Let’s explore the requirements of the movie ticket booking system in the next lesson.

Back

Code for the Meeting Scheduler

Mark As Completed

**Requirements for the Movie Ticket Booking System**

Learn about all requirements of the movie ticket booking system.

**We'll cover the following**

* [Requirement collection](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Requirement-collection)

In this lesson, we’ll list the requirements of our movie ticket booking system. This is a very crucial step as requirements define the scope of a problem, so 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.

**Requirement collection**

The following are the requirements that we have defined for the movie ticket booking problem:

**R1:** There exist multiple cinemas in the city, and the cinema has multiple halls.

**R2:** Each movie in the cinema can have multiple shows, however, one hall will only show one show at a time.

**R3:** The cinema displays all available showtimes of a movie.

**R4:** Users can search movies based on the following four criteria: title, language, genre, and release date.

**R5:** Users can make a booking at any cinema hall at the available showtime.

**R6:** The booking can either be made by the customer online or via a walk-in by the ticket agent.

**R7:** Online customers can only pay using a credit card, while walk-in customers can pay using cash or credit card through the ticket agent.

**R8:** Users can select multiple available seats for a show from a given seating arrangement.

**R9:** Each seat type has a fixed cost. There are three types of seats: silver, gold, and platinum.

**R10:** There can only be one ticket allocated per seat.

**R11:** No two customers should be able to reserve the same seat.

**R12:** The admin can perform the following five actions on the show times and the movie:

* Add a show
* Delete a show
* Update a show
* Add a movie
* Delete a movie

**R13:** The system should be able to differentiate between available and booked seats.

**R14:** The system should generate a notification for the following three cases:

* A new movie has been released.
* A booking has been made.
* A booking has been canceled.

We've identified our requirements for the problem, and in the next lesson, we will define different use cases of our movie booking system.

# Use Case Diagram for the Movie Ticket Booking System

Learn how to define use cases and create the corresponding use case diagram for the movie ticket booking system.

**We'll cover the following**

* [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Primary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Primary-actors)
  + [Secondary actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Secondary-actors)
* [Use Cases](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-Cases)
  + [Admin](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Admin)
  + [Customer](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Customer)
  + [Ticket agent](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Ticket-agent)
  + [System](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#System)
* [Relationships](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationships)
  + [Generalization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Generalization)
  + [Associations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Associations)
  + [Include](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Include)
* [Use case diagram](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Use-case-diagram)

Let’s build the use case diagram of the movie ticket booking system and understand the relationship between its different components.

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

## System

Our system is the “movie ticket booking system.”

## Actors

Now, we are going to define the main actors of our movie ticket booking system.

### Primary actors

* **Customer:**This actor can book one or more movie tickets for any movie as well as modify and cancel the booking. They also need to pay for the movie tickets that they booked.
* **Ticket agent:** The ticket agent will assist the customer and perform almost all the tasks that a customer can do, such as creating a booking of the movie ticket on behalf of the customer, except for modifying and canceling a booking.

### Secondary actors

* **Admin:** This can add, remove, or update a show and movie.
* **System:** This is responsible for sending out notifications for new movies, bookings, cancellations, etc.

## Use Cases

In this section, we will define the use cases for movie ticket booking systems. We have listed the use cases according to their respective interactions with a particular actor.

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

### Admin

* **Add show:**To add a new show for any particular movie
* **Modify show:**To modify a show
* **Delete show:**To delete or cancel a show
* **Add movie:**To add a new movie to the calendar
* **Search movie:**To search for any particular movie based on the given criteria (title, language, genre, release date)
* **Delete movie:**To delete any particular movie

### Customer

* **Search movie:**To search for any particular movie based on the given criteria (title, language, genre, release date)
* **Create/Modify/View/Cancel booking:**To create or cancel a booking for any show of a movie, and to view or modify the booking details for any show of a movie
* **Reserve a seat:**To reserve a seat from the available seats on a seating map for any show of a movie
* **Pay using credit card/cash:**To pay the movie ticket fee via credit card or cash

### Ticket agent

* **Search movie:**To search for any particular movie based on the given criteria (title, language, genre, release date)
* **Create/View booking:**To create a booking for any show of a movie and view its details
* **Reserve a seat:**To reserve a seat from the available seats on a seating map for any show of a movie

### System

* **Send new movie notification:**To send a notification of any new movie launched
* **Send booking notification:**To send a notification of the bookings made
* **Send cancellation notification:** To send a notification of any canceled bookings

## Relationships

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

### Generalization

We can search for a movie with the title, language, genre, or release date. This shows that the “Search movie” use case has a generalization relationship with the “By movie title,” “By movie language,” “By movie genre,” and “By movie release date” use cases.

### Associations

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Admin** | **Customer** | **Ticket** **Agent** | **System** |
| Add show | Search movie | Search movie | Send new movie notification |
| Modify show | Create/view/modify/cancel booking | Create/view booking | Send booking notification |
| Delete show | Reserve a seat | Reserve a seat | Send cancellation notification |
| Add movie | Pay using credit card/cash |  |  |
| Search movie |  |
| Delete movie |

### Include

* The “Create booking” use case has an include relationship with the “Reserve seat” use case, since the booking creation process includes reserving a seat on a seating map.
* The “Pay using cash” and “Pay using credit card” use cases have an include relationship with the “Send booking notification” use case, since once the payment has been made and verified, only then will the system generate a notification.
* The “Cancel booking” use case has an include relationship with the “Refund payment” use case, since this is part of the process.
* The “Add movie” use case has an include relationship with the “Send new movie notification” use case, since a notification will only be sent when an admin adds a new movie.
* The “Cancel booking,” “Delete movie,” and “Delete show” use cases have an include relationship with the “Send cancellation notification” use case, since a notification will only be sent when an admin cancels the show or the movie or a customer cancels their booking.

## Use case diagram

Here is the use case diagram of the movie ticket management system:

A diagram of movie tickets

Description automatically generated

# Class Diagram for the Movie Ticket Booking System

Learn to create a class diagram for the movie ticket booking system using the bottom-up approach.

**We'll cover the following**

* [Components of a movie ticket booking system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Components-of-a-movie-ticket-booking-system)
  + [Seat](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Seat)
  + [Show time](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Show-time)
  + [Hall](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Hall)
  + [Cinema](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Cinema)
  + [City](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#City)
  + [Movie](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Movie)
  + [Movie ticket](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Movie-ticket)
  + [Payment](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Payment)
  + [Person](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Person)
    - [Customer](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Customer)
    - [Admin](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Admin)
    - [Ticket agent](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Ticket-agent)
  + [Notification](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Notification)
  + [Catalog](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Catalog)
  + [Search](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Search)
  + [Booking](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Booking)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
* [Relationship between the classes](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Relationship-between-the-classes)
  + [Association](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Association)
  + [Composition](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Composition)
  + [Aggregation](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Aggregation)
  + [Generalization](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Generalization)
  + [Inheritance](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Inheritance)
* [Class diagram of the movie ticket booking system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Class-diagram-of-the-movie-ticket-booking-system)
* [Handle concurrency](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Handle-concurrency)
* [Design pattern](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Design-pattern)
* [Additional requirements](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Additional-requirements)

In this lesson, we are going to identify and design the classes, abstract classes, and interfaces based on the requirements that we have previously gathered from the interviewer in our movie ticket booking system.

## Components of a movie ticket booking system

As mentioned earlier, we should design the movie ticket booking system using a bottom-up approach.

### Seat

Our movie ticket booking system should have a Seat object that will be an abstract class and can be of three types: silver, gold, and platinum. Each seat type will have a fixed cost and differs from one another.

The visual representation of these classes is as follows:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a movie ticket system

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screen shot of a movie ticket

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

#### Customer

The Customer class refers to a user trying to create an online booking for any movie. A customer can also update the booking details and cancel the booking.

#### Admin

The Admin class is responsible for performing actions like adding, updating, and removing movies and shows.

#### Ticket agent

The TicketAgent class is responsible for creating bookings on behalf of walk-in customers. However, unlike the Customer class, a ticket agent cannot update or cancel a booking.

The class diagram of all these is provided below:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

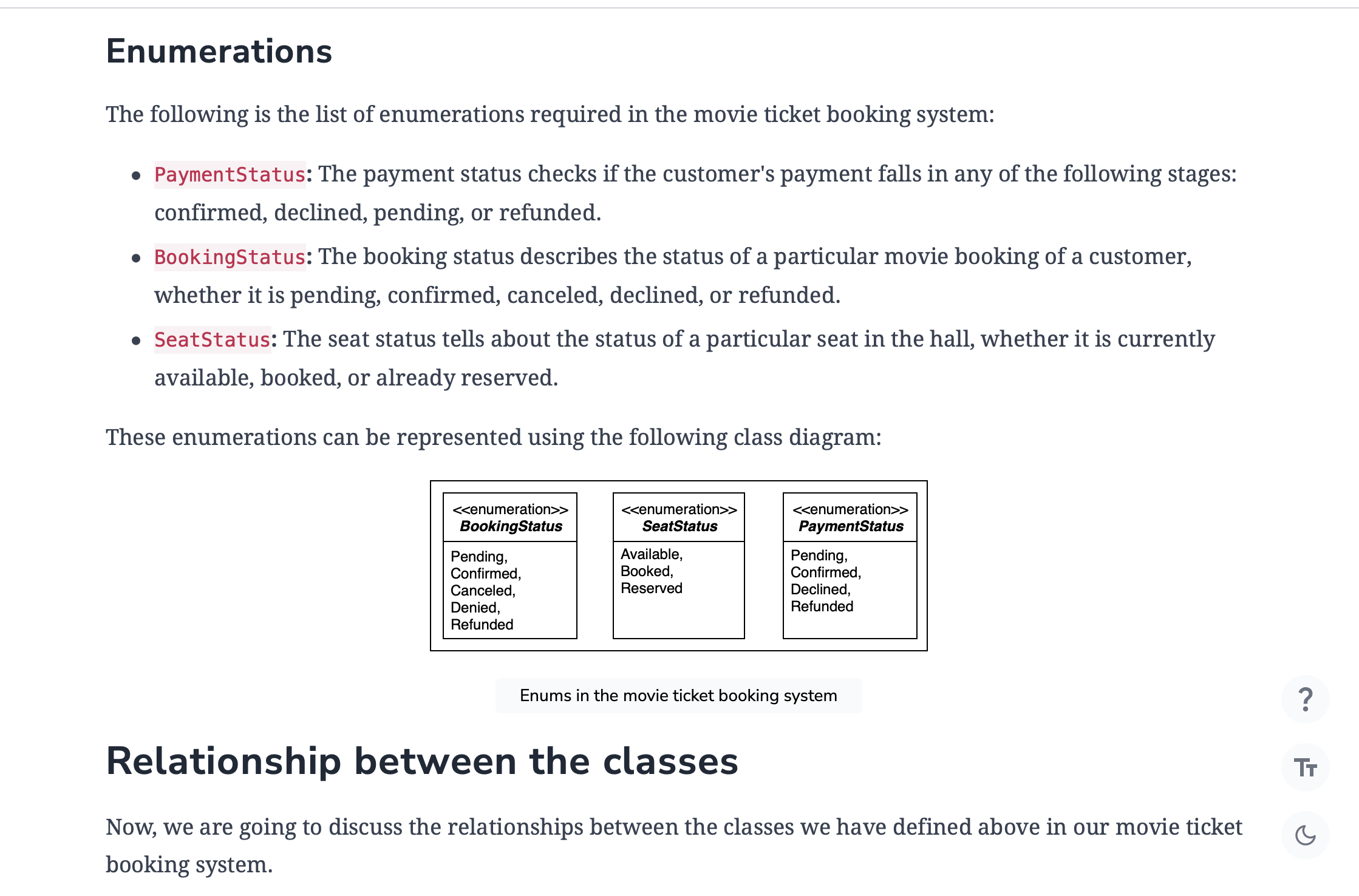
Description automatically generated

* Search movies by their genre.
* Search movies by their release date.

The UML representation of the class is shown below:

A screenshot of a computer

Description automatically generated



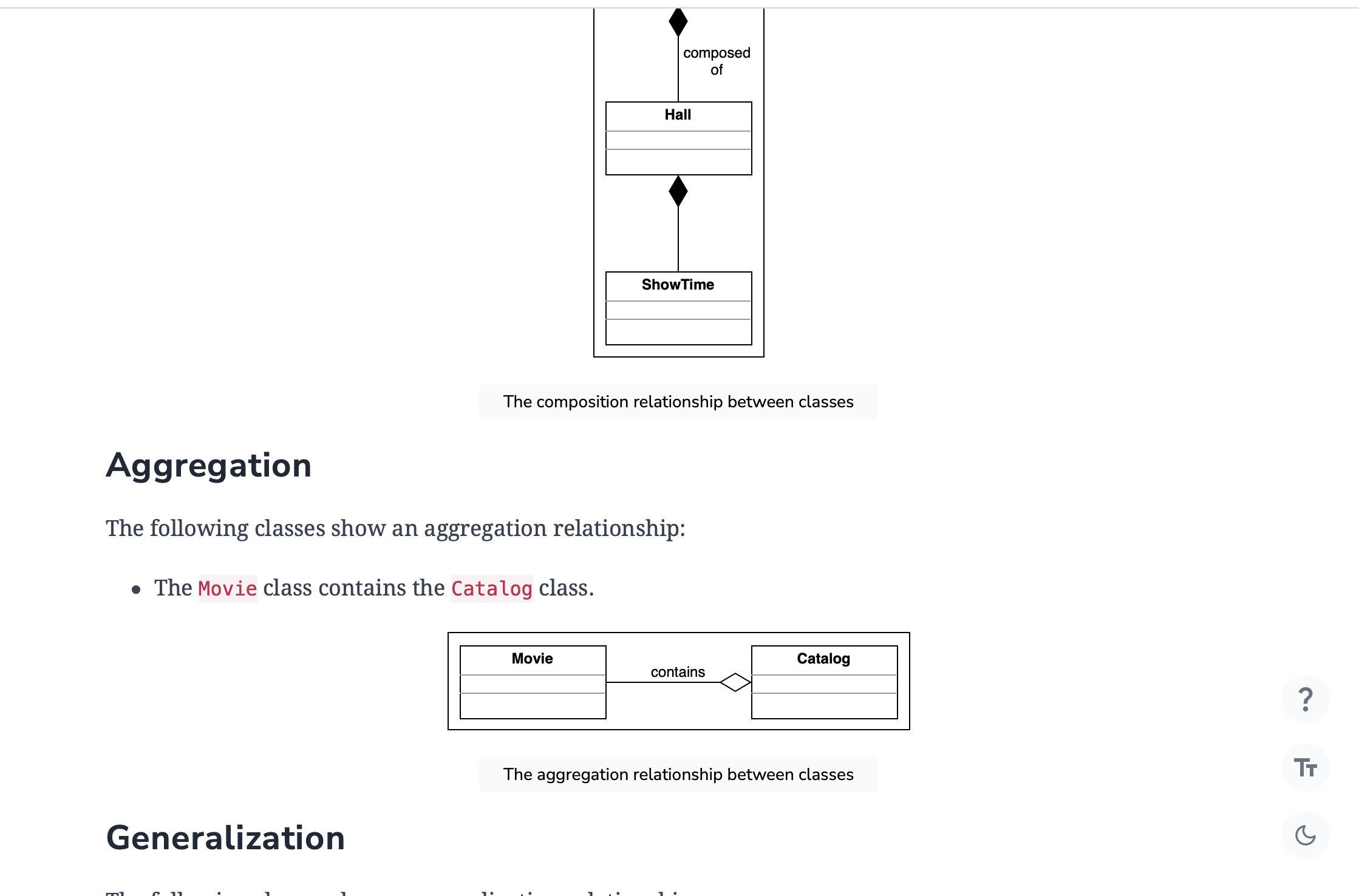
### Association

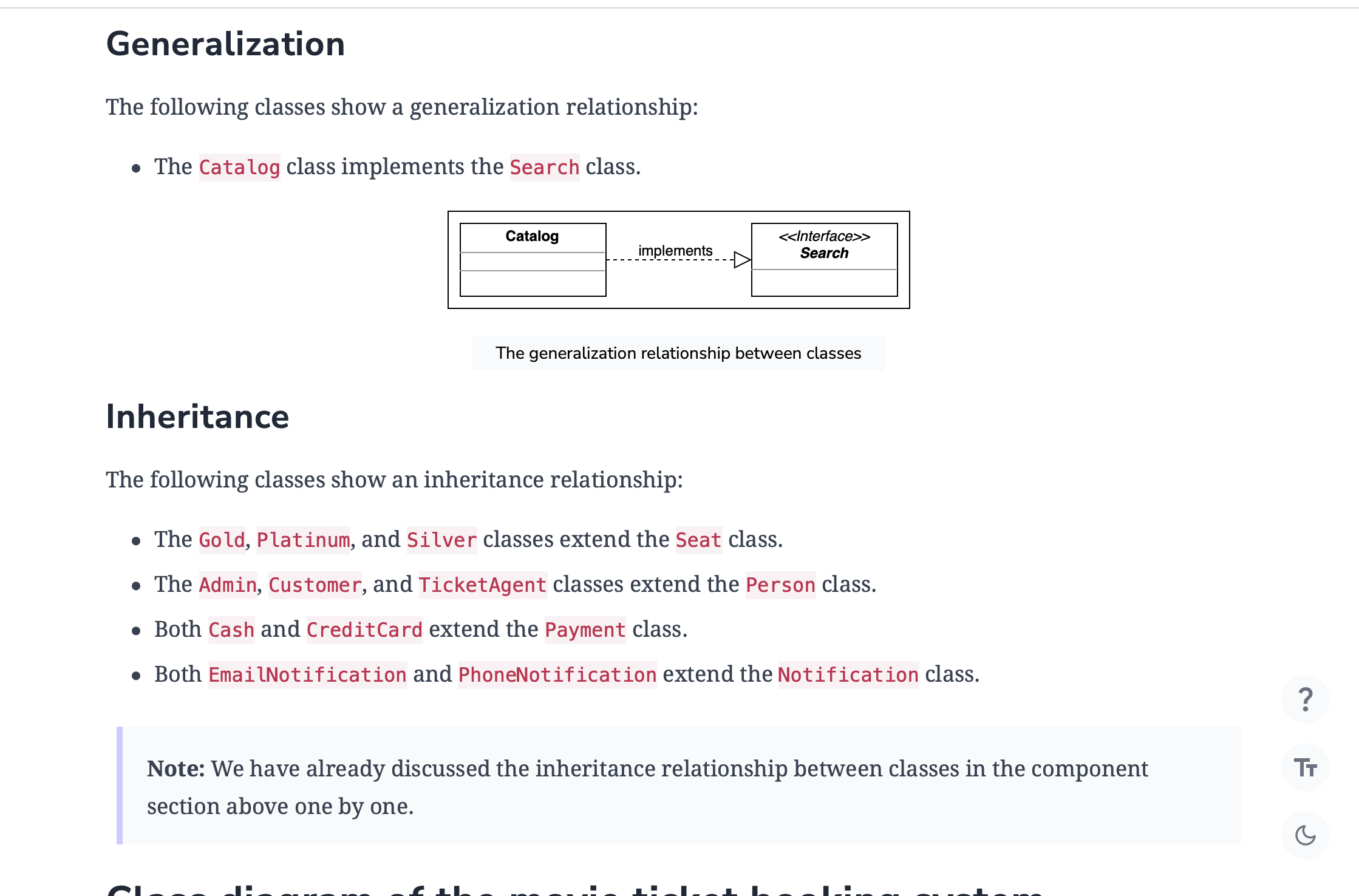
The class diagram has the following association relationships:

* The City class has a one-way association with Cinema.
* The Admin class has a one-way association with ShowTime, Movie, and Notification.
* Both Customer and TicketAgent have a two-way association with Notification.
* The Booking class has a one-way association with ShowTime, MovieTicket, and Seat.
* The Booking class has a two-way association with Payment.
* The ShowTime class has a one-way association with Seat and MovieTicket.
* The Movie class has a one-way association with ShowTime and MovieTicket

A diagram of a movie

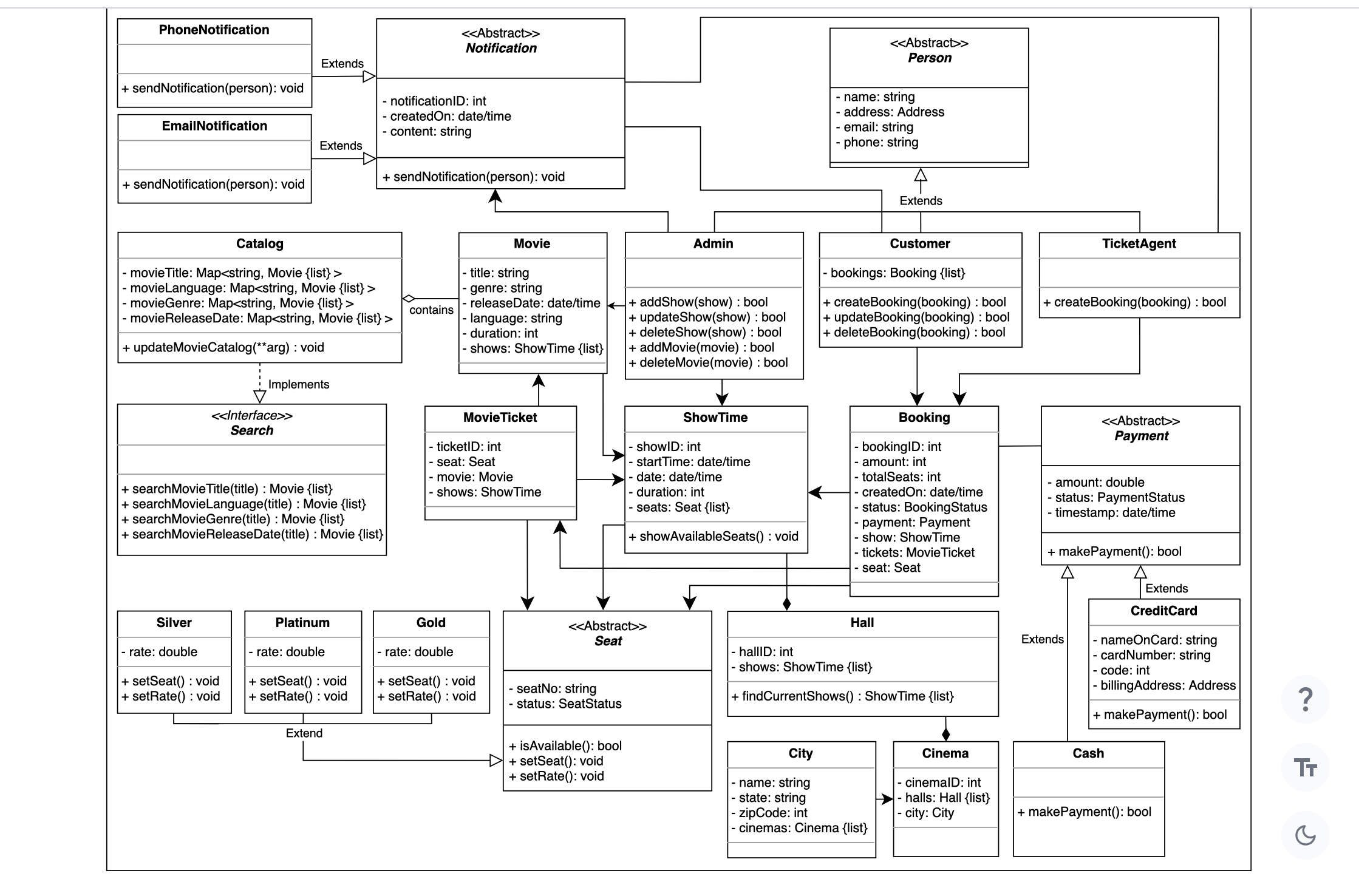
Description automatically generated





## Class diagram of the movie ticket booking system

Here’s the complete class diagram for our movie ticket booking system:



**Handle concurrency**

One of the major requirements of the movie ticket booking system is that no two customers can book the same seat in the same show of a movie. This is to prevent the case of double booking where two people have the same seats allocated to them.

To handle concurrent seat allocations, we can use the concept of locks, more specifically, optimistic locks or versioning.

The entire process has been summarized in the illustration given below:

* Customer 1 tries to select some seats on the seat selection map and gets assigned a version number V1.
* At the same time, customer 2 also tries to select some seats on the seat selection map and gets assigned a version number V1. However, both customers select the same seat.
* A lock time gets assigned that keeps track of the timestamps related to the customers entering in the payment section.
* The mutex lock gets acquired and the expiration time period starts. Since customer 1 took less time, it starts the payment process while customer 2 needs to wait.
* Customer 1 successfully completes the payment. The mutex lock also gets released.
* Versions of both customer 1 and the system get updated. Since the lock has been released, customer 2 can now perform the action of booking the tickets again.
* However, since the system version has been updated, customer 2 would need to perform the entire task from the beginning.
* The previous seat has become unavailable so customer 2 selects a new seat.

A screenshot of a diagram

Description automatically generated

## Design pattern

In the movie ticket booking system, there can be multiple seat types for a cinema hall, and each seat type will have its own formula to calculate the ticket fare. Therefore, the Strategy design pattern can be applied here, which will design a separate strategy or algorithm to calculate the price of each seat type. It is applied through the definition of a function that provides the implementation of the rates for each seat type using different strategies. This strategy can also differ based on a movie’s popularity or taxes

## Additional requirements

The interviewer can introduce some additional requirements in the movie ticket booking system, or they can ask some follow-up questions. Let’s see some examples of additional requirements:

**Discount:** Customers can use a coupon to add a discount to their payment. The class diagram provided below shows the relationship of Discount with the Payment class:

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

**Sequence Diagram for the Movie Ticket Booking System**

Visualize the sequence diagrams for the creation and payment of a booking, and practice the concepts with a challenge.

**We'll cover the following**

* [Create a booking](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Create-a-booking)
* [Payment of a booking](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Payment-of-a-booking)
* [Sequence challenge: Cancel booking](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Sequence-challenge:-Cancel-booking)

A sequence diagram is 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 movie ticket booking system. In this lesson, we will create sequence diagrams for the following three interactions:

* **Create a booking:** The customer creates a booking for a show.
* **Payment for the booking:** The customer pays for the booking.
* **Sequence challenge:** The customer cancels their booking.

**Create a booking**

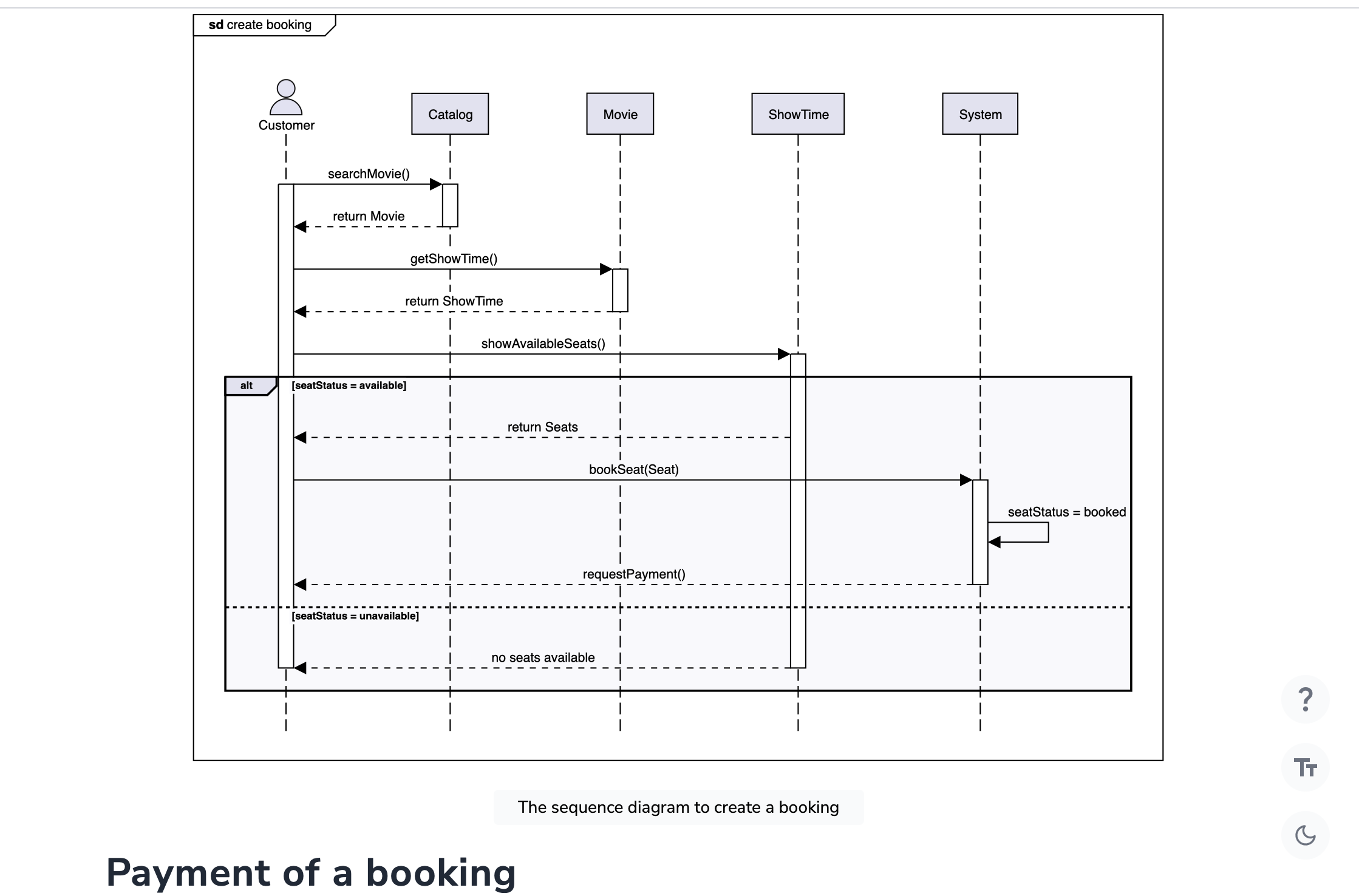
The sequence diagram for creating a booking should have the following actors and objects that will interact with each other:

* **Actor:** Customer
* **Objects:**Catalog, Movie, and ShowTime
* System

Here’re the steps of the interaction to create a booking interaction:

1. The customer searches for a movie from the catalog.
2. The catalog returns the required movie.
3. They request show times for the movie they wanted to book.
4. The available show times for the movie are returned.
5. The customer requests available seats for the show.
6. If seats are available:
   1. The customer receives a list of available seats.
   2. The customer books their desired seats using the system.
   3. The seat status is set to book.
   4. The system requests payment against the booked seats from the customer.
7. Else if the seats are not available:
   1. The customer is informed that no seats are available.

Based on the order above, the sequence diagram for creating a booking in the movie ticket booking system is given below.



The sequence diagram for payment of a booking should have the following actors and objects that will interact with each other:

* **Actor:** Customer
* **Object:** Payment, Seat
* System

Here are the steps in the payment of a booking interaction:

1. The customer initiates a payment against the booking fee.
2. The payment processes the payment and informs the system.
3. If the payment is complete:
   1. The payment informs the customer about the completed payment.
   2. The system creates an object for the booking.
   3. The booking status is updated and the system is informed.
   4. The system sends a notification to the customer containing the booking information.
4. Else if the payment is declined:
   1. The payment informs the customer about the declined payment.
   2. The system requests to revert the seat status.
   3. The seat status is updated to available and returned to the system.
   4. The system informs the customer that the booking is declined.

Based on the order above, the sequence diagram for payment of a booking in the movie ticket booking system is given below.

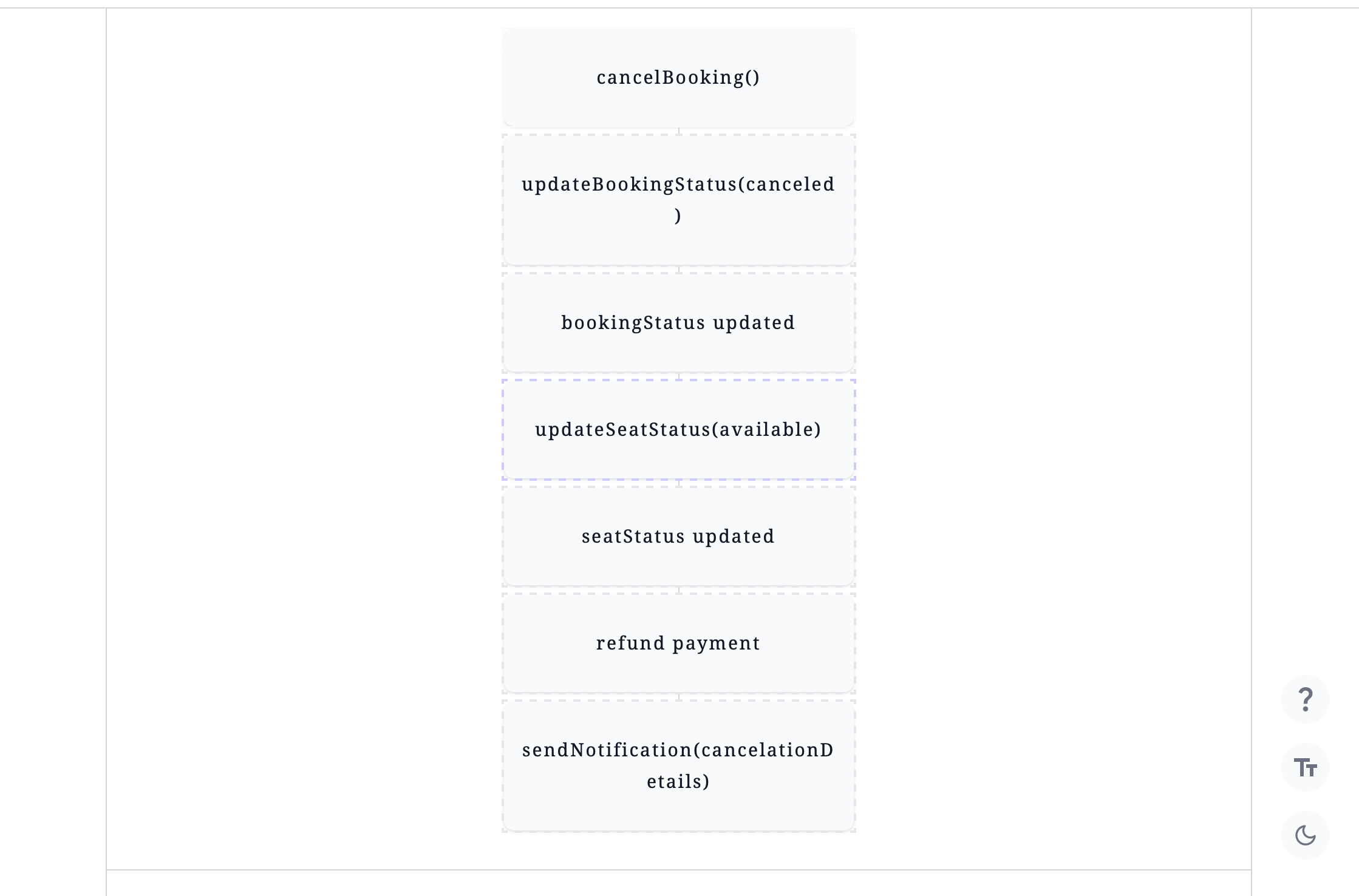
A screenshot of a computer screen

Description automatically generated

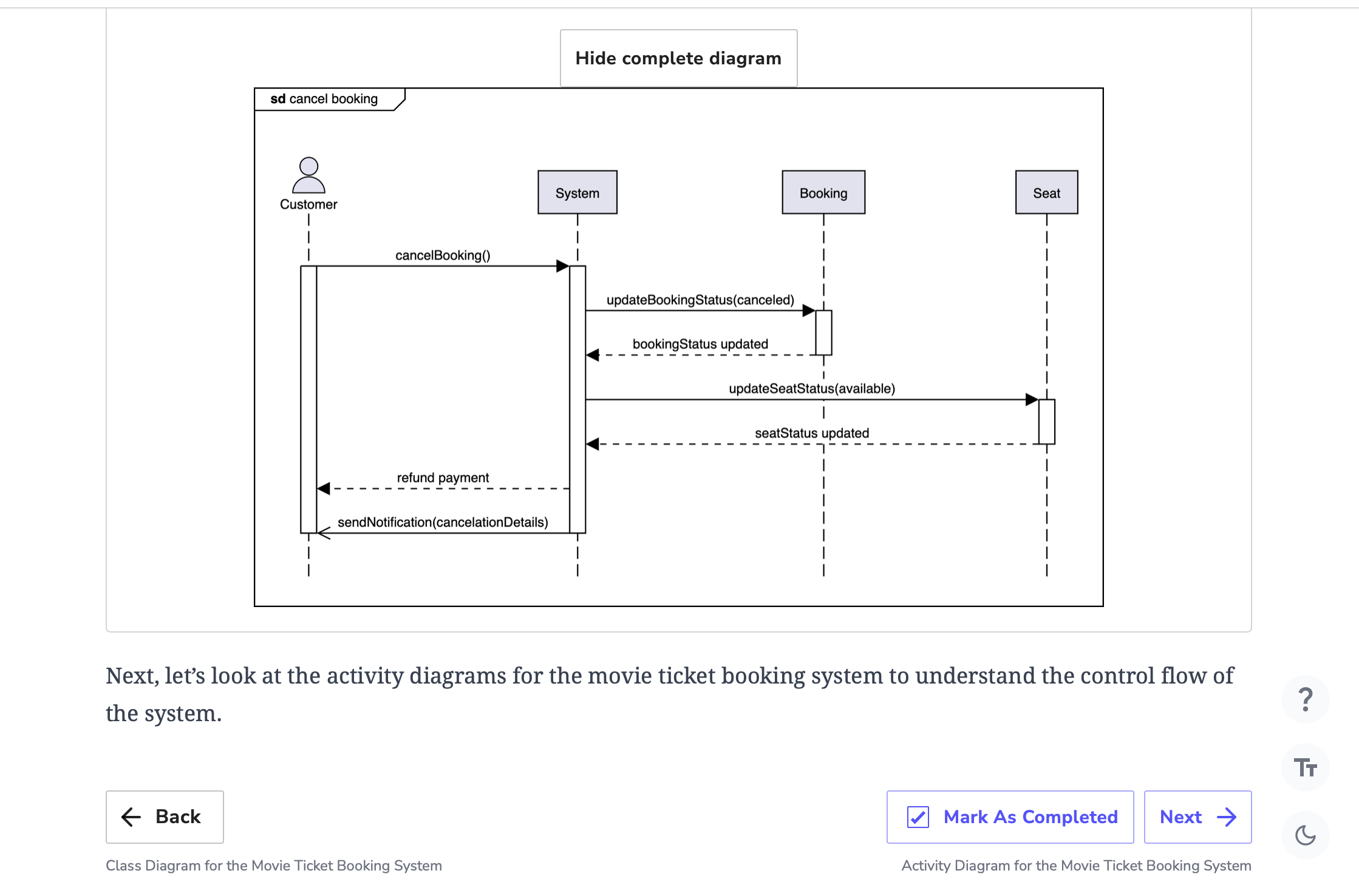


Notice that the arrows in the diagram above are numbered from 1 to 7. The message boxes shown below are the messages to be exchanged between the actor(s) and object(s). Can you rearrange the messages below in the correct sequence of order they should appear in the skeleton of the sequence diagram given above?

**Note:** If you get stuck, just click the “Show Solution” button to check the correct answer.



Alternatively, you can also click the "Show complete diagram" button below to see the complete sequence diagram for the cancel booking interaction.



# Activity Diagram for the Movie Ticket Booking System

Create some activity diagrams for the movie ticket booking system problem.

**We'll cover the following**

* [The customer makes a booking for the movie](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#The-customer-makes-a-booking-for-the-movie)
  + [States](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#States)
  + [Actions](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actions)
* [Activity challenge: Admin cancels a show](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Activity-challenge:-Admin-cancels-a-show)

An activity diagram is 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 our movie ticket booking system. In this lesson, we will create activity diagrams for the following two activities:

* The customer makes a booking for the movie.
* **Activity challenge:**The admin cancels a show.

## The customer makes a booking for the movie

The following are the states and actions that will be involved in this activity diagram.

### States

**Initial state:**The customer opens a search for a movie.

**Final state:**The customer receives a ticket for the movie.

### Actions

The customer searches for a movie by specific criteria and selects their movie. The customer then selects their required seat and pays according to their seat type. The payment is carried out either through cash or a credit card. After successful payment, the customer receives the movie ticket.

Based on the order above, the activity diagram of a customer making a booking for the movie is given below.

**Note:** Here we assume that the customer is purchasing a single cinema seat.

A diagram of a movie

Description automatically generated

## Activity challenge: Admin cancels a show

You will create an activity diagram of an admin canceling a show for a movie.

A skeleton of the activity diagram is given below.

## Activity challenge: Admin cancels a show

You will create an activity diagram of an admin canceling a show for a movie.

A skeleton of the activity diagram is given below.

A diagram of a diagram

Description automatically generated

Notice that the actions in the diagram above are numbered from 1 to 8. The slots shown below represent the activities and the arrows represent the flow from one activity to the other.

Can you rearrange the given slots in the correct order they should appear in the activity diagram above?

**Note:** If you get stuck, just click the “Show Solution” button to check out the correct answer.

Alternatively, you can also click the "Show complete diagram" button below to see the complete sequence diagram.

Show complete diagram

We've looked at some of the activity diagrams of our movie ticket booking system. In the next lesson, we will present the code for our designed classes in some of the most popular languages.

Back

# Code for the Movie Ticket Booking System

Write the object-oriented code to implement the design of the movie ticket booking problem.

**We'll cover the following**

* [Movie ticket booking system](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Movie-ticket-booking-system)
  + [Enumerations](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Enumerations)
  + [Actors](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Actors)
  + [Seat](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Seat)
  + [Movie, showtime, and movie ticket](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Movie,-showtime,-and-movie-ticket)
  + [City, cinema, and hall](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#City,-cinema,-and-hall)
  + [Payment](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Payment)
  + [Notification](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Notification)
  + [Booking](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Booking)
  + [Search and catalog](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Search-and-catalog)
* [Wrapping up](https://www.educative.io/order-confirmation/stripe/subscription-buy?payment_intent=pi_3O1FFeKhXp6R50hI1xakUX3g&payment_intent_client_secret=pi_3O1FFeKhXp6R50hI1xakUX3g_secret_EDla1wT2fgsm34I6up6bRJM7s&transaction_id=1d4a8dd5-516d-4906-aa95-9e9effdf447d#Wrapping-up)

We’ve gone over different aspects of the movie ticket booking 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 movie ticket booking 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 movie ticket booking system:

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

## Movie ticket booking system

In this section, we will 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 method functions.

### Enumerations

The following code provides the definition of the various enumerations used in the movie ticket booking 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.

Ja

// Enumerations

enum PaymentStatus {

PENDING,

CONFIRMED,

DECLINED,

REFUNDED

}

enum BookingStatus {

PENDING,

CONFIRMED,

CANCELLED,

DENIED,

REFUNDED

}

enum SeatStatus {

AVAILABLE,

BOOKED,

RESERVED

}

### Actors

This section contains the different people that will interact with our movie ticket systems, such as a Customer, Admins, and TicketAgents. All of these classes will inherit the properties of the Person class. The definition of these classes is given below:

// Person is an abstract class

public abstract class Person {

private String name;

private String address;

private String phone;

private String email;

}

public class Customer extends Person {

private List<Bookings> bookings; // List of bookings

// booking here refers to an instance of the Booking class

public boolean createBooking(Booking booking);

public boolean updateBooking(Booking booking);

public boolean deleteBooking(Booking booking);

}

public class Admin extends Person {

// show here refers to an instance of the ShowTime class

public boolean addShow(Show show);

public boolean updateShow(Show show);

public boolean deleteShow(Show show);

public boolean addMovie(Movie movie);

public boolean deleteMovie(Movie movie);

}

public class TicketAgent extends Person {

// booking here refers to an instance of the Booking class

public boolean createBooking(Booking booking);

}

### Seat

The Seat will be an abstract class, which serves as a parent for three different types of seats: Platinum, Gold, and Silver. The definition of the Seat and its child classes is given below:

// Seat is an abstract class

public abstract class Seat {

// Data members

private String seatNo;

private SeatStatus status; // Refers to the SeatStatus enum

// Member functions

public boolean isAvailable();

public abstract void setSeat();

public abstract void setRate();

}

public class Platinum extends Seat {

private double rate;

public void setSeat() {

// definition

}

public void setRate() {

// definition

}

}

public class Gold extends Seat {

private double rate;

public void setSeat() {

// definition

}

public void setRate() {

// definition

}

}

public class Silver extends Seat {

private double rate;

public void setSeat() {

// definition

}

public void setRate() {

// definition

}

}

### Movie, showtime, and movie ticket

Next, we will explore the ShowTime, Movie, and MovieTicket classes that provide the details of the movie to the customer. Th

public class Movie {

// Data members

private String title;

private String genre;

private Date releaseDate;

private String language;

private int duration;

private List<ShowTime> shows;

}

public class ShowTime {

// Data members

private int showId;

// The Date datatype represents and deals with both date and time

private Date startTime;

private Date date;

private int duration;

private List<Seat> seats;

// Displays the list of available seats

public void showAvailableSeats();

}

public class MovieTicket {

// Data members

private int ticketId;

private Seat seat;

private Movie movie;

private ShowTime show;

}

### City, cinema, and hall

This section contains classes like Hall, Cinema, and City that make up the infrastructure of our movie ticket system. The definition of these classes is given below:

public class City {

// Data members

private String name;

private String state;

private int zipCode;

private List<Cinema> cinemas;

}

public class Cinema {

// Data members

private int cinemaId;

private List<Hall> halls;

private City city;

}

public class Hall {

// Data members

private int hallId;

private List<ShowTime> shows;

// Returns list of shows

public List<ShowTime> findCurrentShows();

}

### Payment

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

// Payment is an abstract class

public abstract class Payment {

// Data members

private double amount;

// The Date datatype represents and deals with both date and time.

private Date timestamp;

private PaymentStatus status;

public abstract boolean makePayment();

}

public class Cash extends Payment {

public boolean makePayment() {

// functionality

}

}

public class CreditCard extends Payment {

// Data members

private String nameOnCard;

private String cardNumber;

private String billingAddress;

private int code;

public boolean makePayment() {

// functionality

}

}

### Notification

The Notification class is an abstract class that is responsible for sending notifications via email or phone/SMS after actions performed by either the admin and/or customer. Its definition is given below:

Java

C#

Python

C++

// Notification is an abstract class

public abstract class Notification {

private int notificationId;

// The Date datatype represents and deals with both date and time.

private Date createdOn;

private String content;

// person here refers to an instance of the Person class

public abstract void sendNotification(Person person);

}

public class EmailNotification extends Notification {

// person here refers to an instance of the Person class

public void sendNotification(Person person) {

// functionality

}

}

public class PhoneNotification extends Notification {

// person here refers to an instance of the Person class

public void sendNotification(Person person) {

// functionality

}

}

A screenshot of a computer program

Description automatically generated

### Search and catalog

The Catalog class contains the movie information and implements the Search interface class to enable the search functionality based on the given criteria (title, language, genre, and release date). The definition of these two classes is given below:

public interface Search {

public List<Movie> searchMovieTitle(String title);

public List<Movie> searchMovieLanguage(String language);

public List<Movie> searchMovieGenre(String genre);

public List<Movie> searchMovieReleaseDate(Date date);

}

public class Catalog implements Search {

HashMap<String, List<Movie>> movieTitles;

HashMap<String, List<Movie>> movieLanguages;

HashMap<String, List<Movie>> movieGenres;

// The Date datatype represents and deals with both date and time.

HashMap<Date, List<Movie>> movieReleaseDates;

public List<Movie> searchMovieTitle(String title) {

// functionality

}

public List<Movie> searchMovieLanguage(String language) {

// functionality

}

public List<Movie> searchMovieGenre(String genre) {

// functionality

}

public List<Movie> searchMovieReleaseDate(Date date) {

// functionality

}

}

**Wrapping up**

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

Back