Movie Ticket Booking System

→ The movie ticket booking system is a system that allows customer to book shows for their favorite movie.

→ Numerous movies in Database

→ Each movie has multiple shows

→ Customer can search & select a movie

Expectation from interviewee

→ How will the system make sure that multiple users do not book the same seat?

→ Will there be a timout session that reserves seat temporarily?

→ Will the system use a first come, first serve algorithm?

→ Will there be transaction locks involved in the system?

→ 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

→ How are we handling these instance, such as the same cinema having multiple cinama halls showing different movies simultaneously

→ Is the same movie being shown at different times in the same cinema/hall?

Requirement for the Movie Ticket Booking Ticket

R1: There exist multiple cinemas in the city, & 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 & 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 the ticket agent.

R7: Online Customers can only pay using a credit card, while walk-in

Customer can pay using cash or credit card through the ticket agent.

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

R9: Each seat type has a fixed cost. There are three types of silver, gold & plantinum

R10: No two customer should be able to reserve the same seat

R11: The admin can perform the following five actions on the Show time & the movie:

- Add a show
- Delete a show
- Update a show
- Add a movie
- Delete a movie

R13: The System should be able to diffentiate between available & booked seats

R14: The System should generate a notification for the following three cases.

- A new movie has been released
- A booking has made
- A booking is canceled

Actors

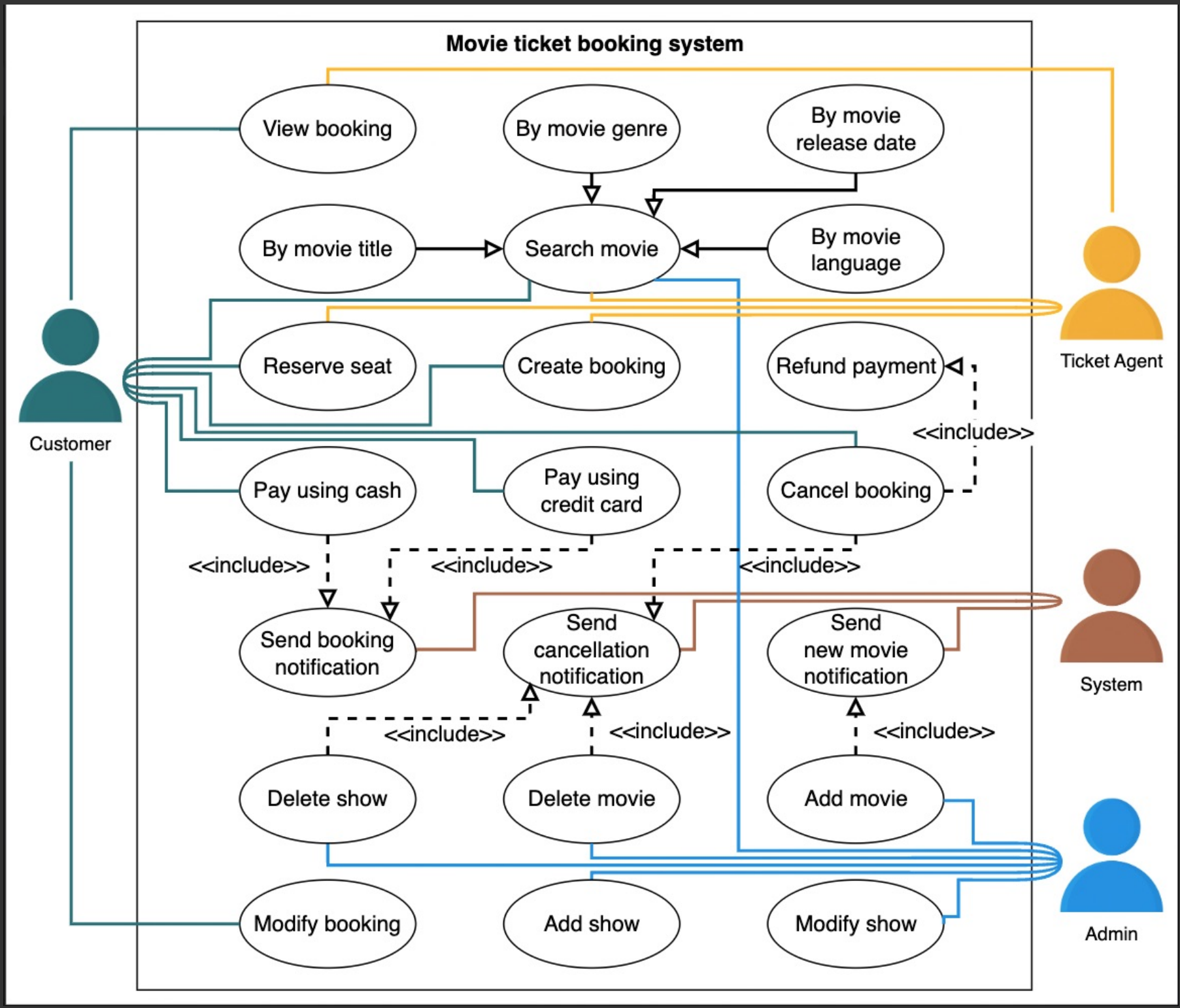
Primary Actors

- Customer → book one or more movie tickets
- Ticket Agent → assist customer

Secondary Actors

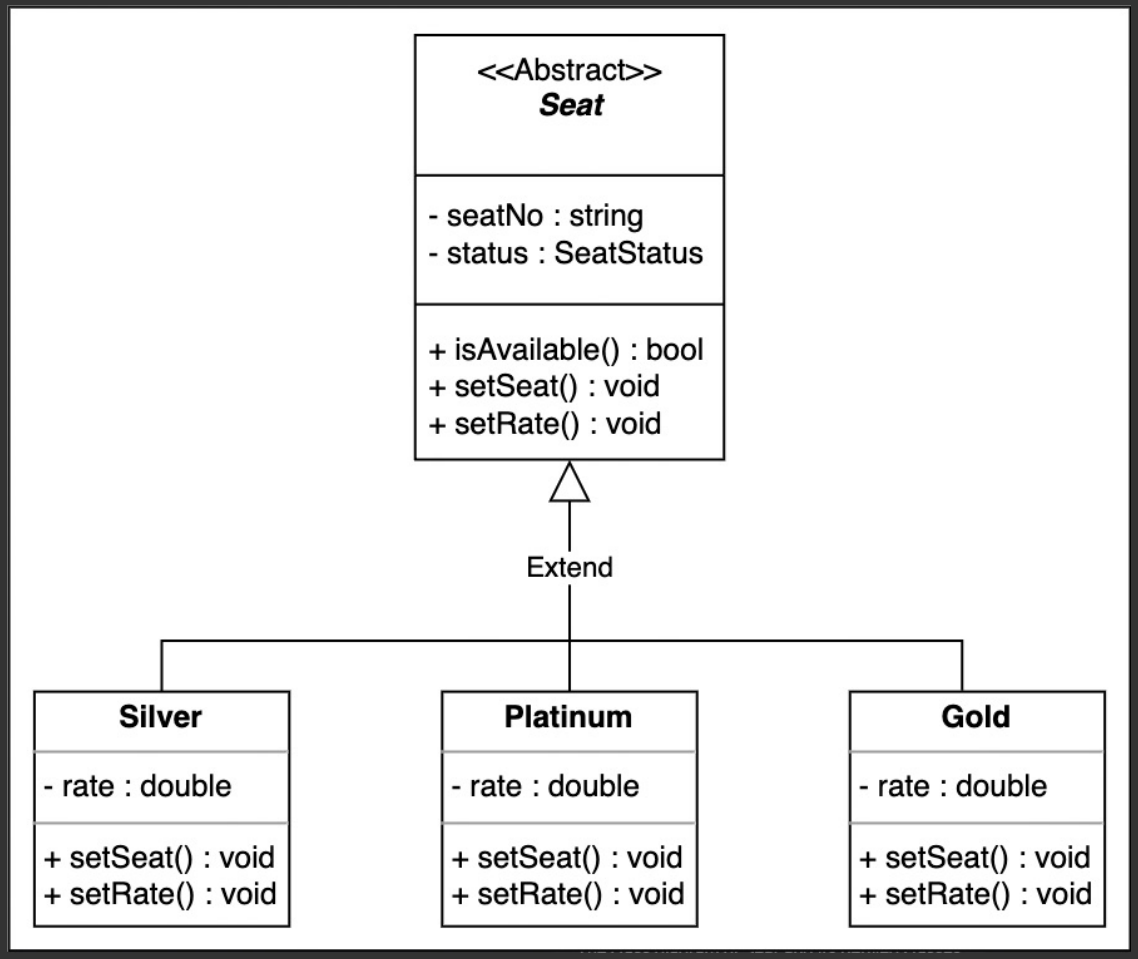
- Admin → add, remove & update → movie
- System → Responsible for sending Notification

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			


Movie ticket booking system
View booking
By movie genre
By movie release date
By movie title
Search movie
By movie language
Reserve seat
Create booking
Refund payment
<<include>>
Pay using cash
Pay using credit card
Cancel booking
<<include>>
<<include>>
<<include>>
Send booking notification
Send cancellation notification
Send new movie notification
<<include>>
<<include>>
<<include>>
Delete show
Delete movie
Add movie
Modify booking
Add show
Modify show
Customer
Ticket Agent
System
Admin

Class Diagram for the Movie Ticket Booking System

1. Seat



2. Show Time

ShowTime
- showID : int - startTime : date/time - date : date/time - duration : int - seats : Seat {list}
+ showAvailableSeats() : void

3. Hall

Hall
- hallID : int - shows : ShowTime {list}
+ findCurrentShows() : ShowTime {list}

4. Cinema

Cinema

- cinemaID : int
- halls : Hall {list}
- city : City

5. City

City

- name : string
- state : string
- zipCode : int
- cinemas : Cinema {list}

6. Movie

Movie

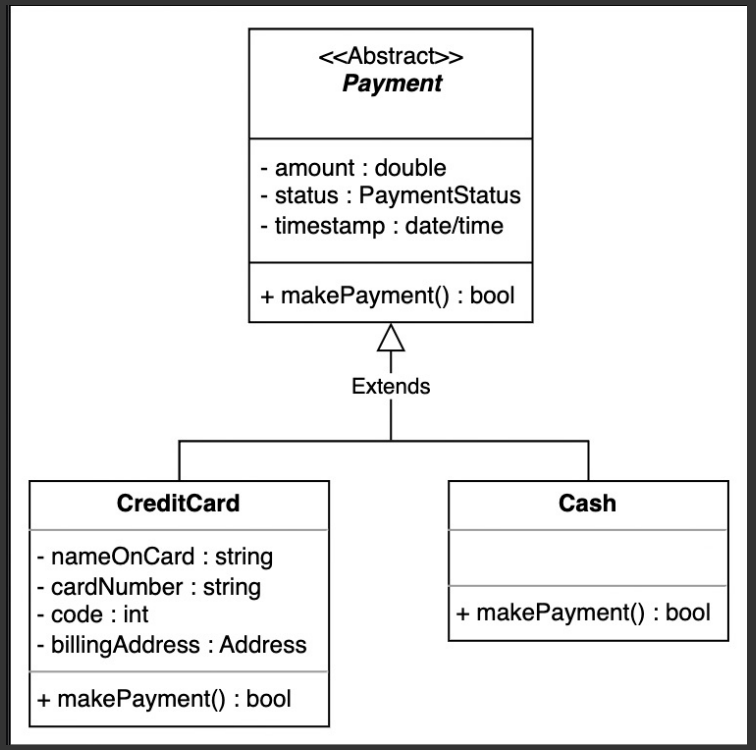
- title : string
- genre : string
- releaseDate : date/time
- language : string
- duration : int
- shows : ShowTime {list}

7. Movie Ticket

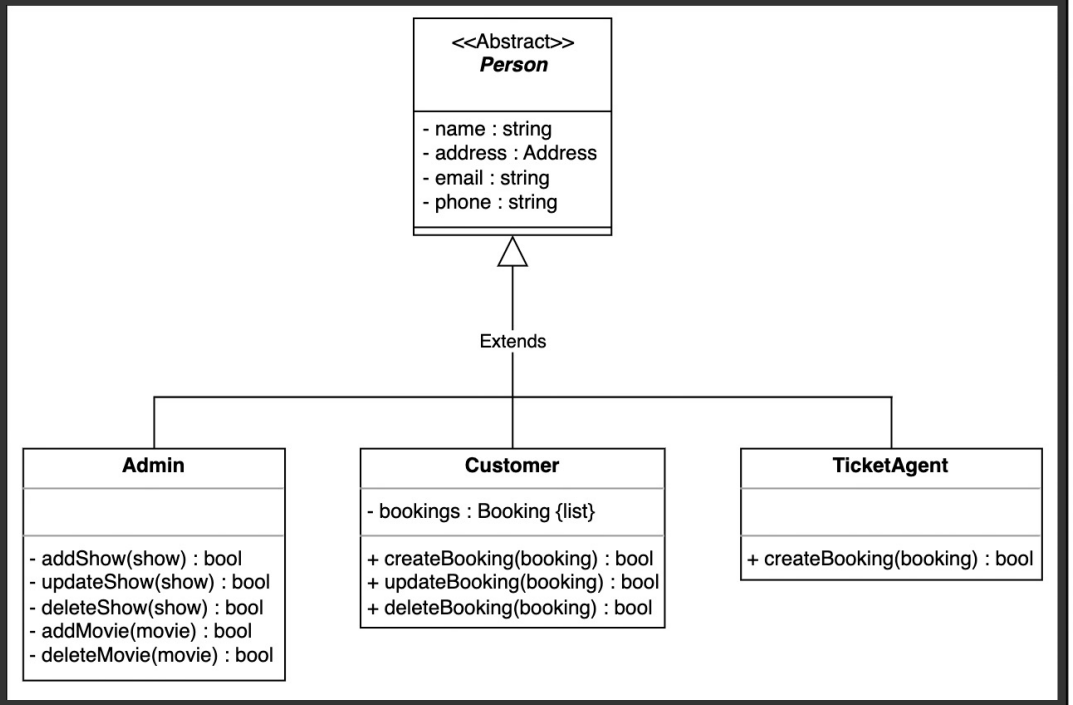
MovieTicket

- ticketID : int
- seat : Seat
- movie : Movie
- shows : ShowTime

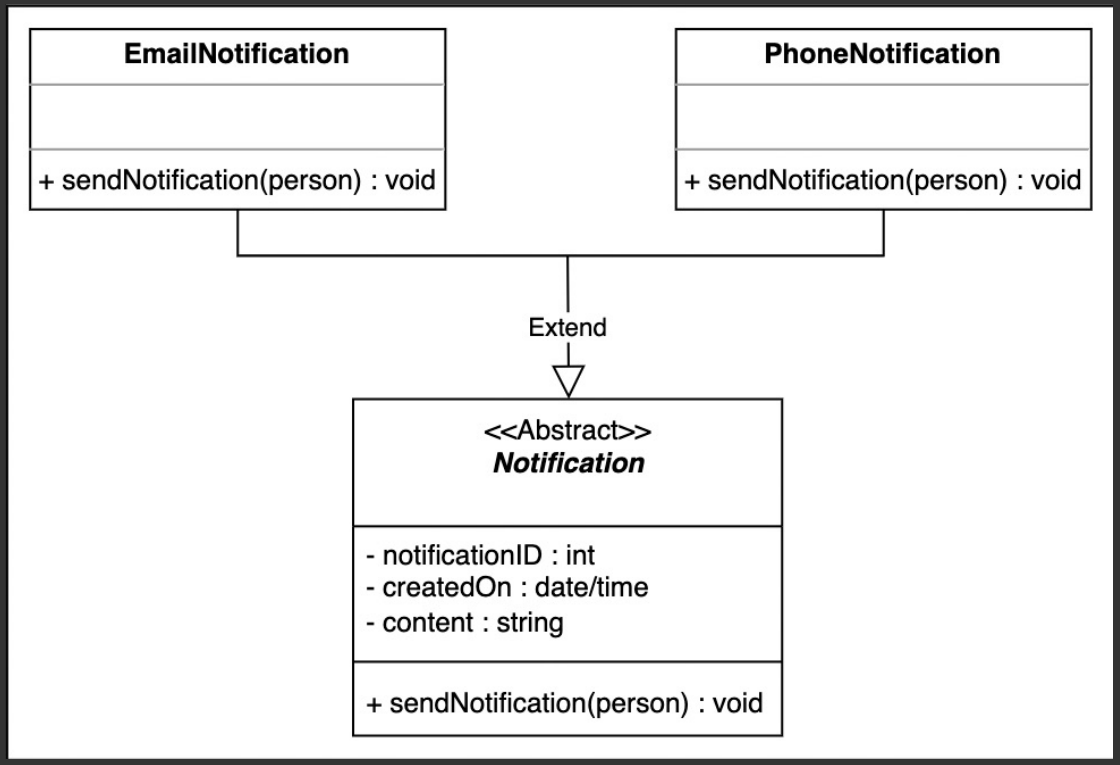
8. Payment



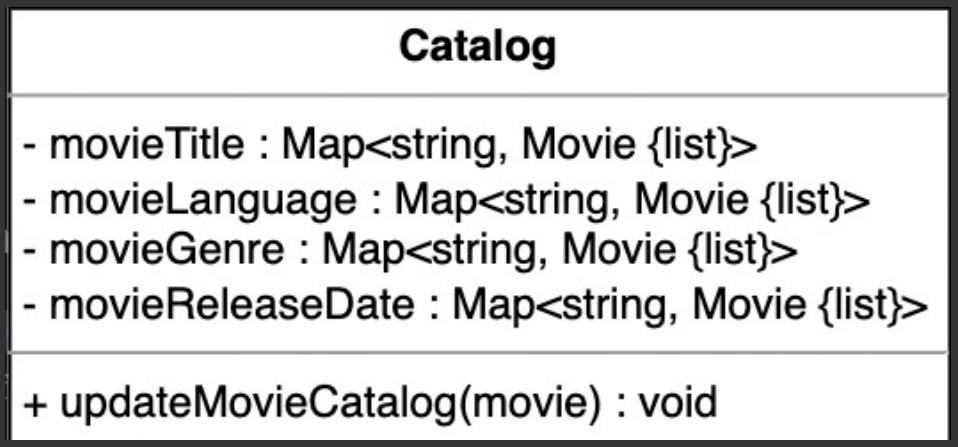
9. Person



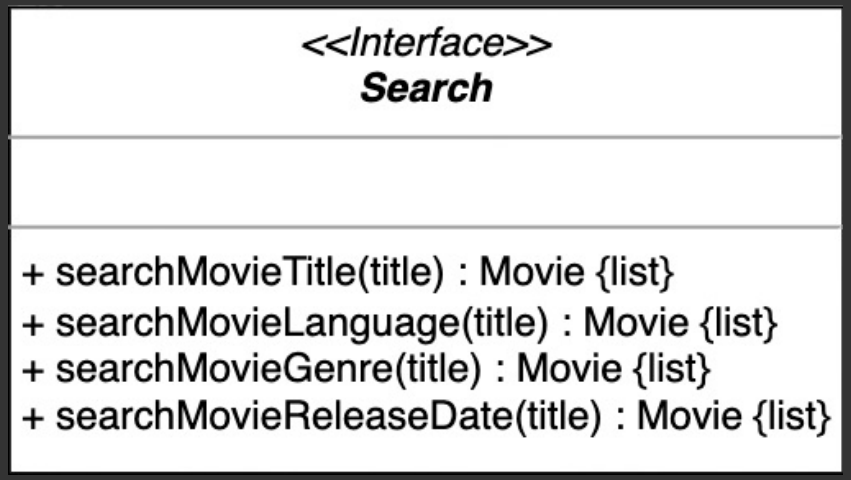
10. Notification



11. Catalog

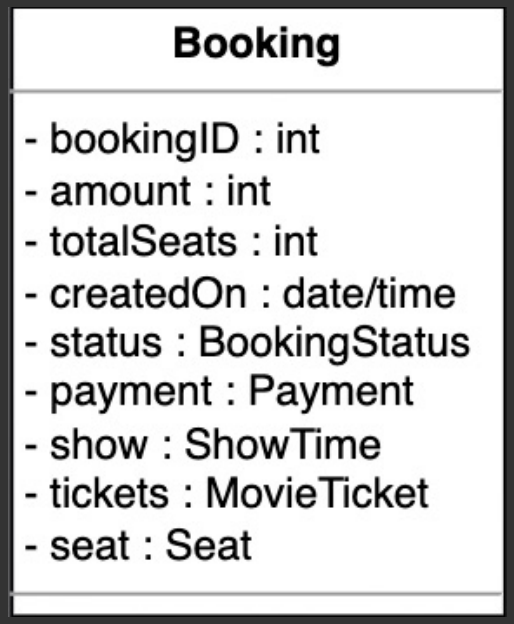


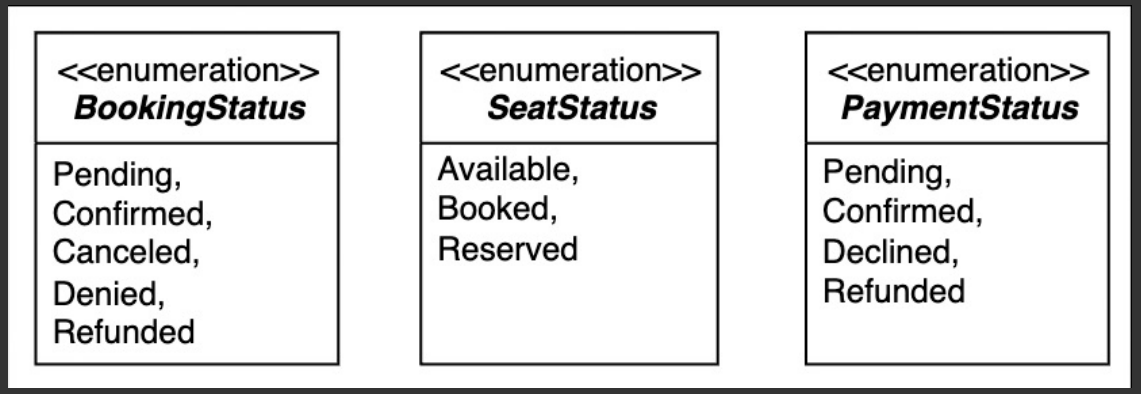
12. Search



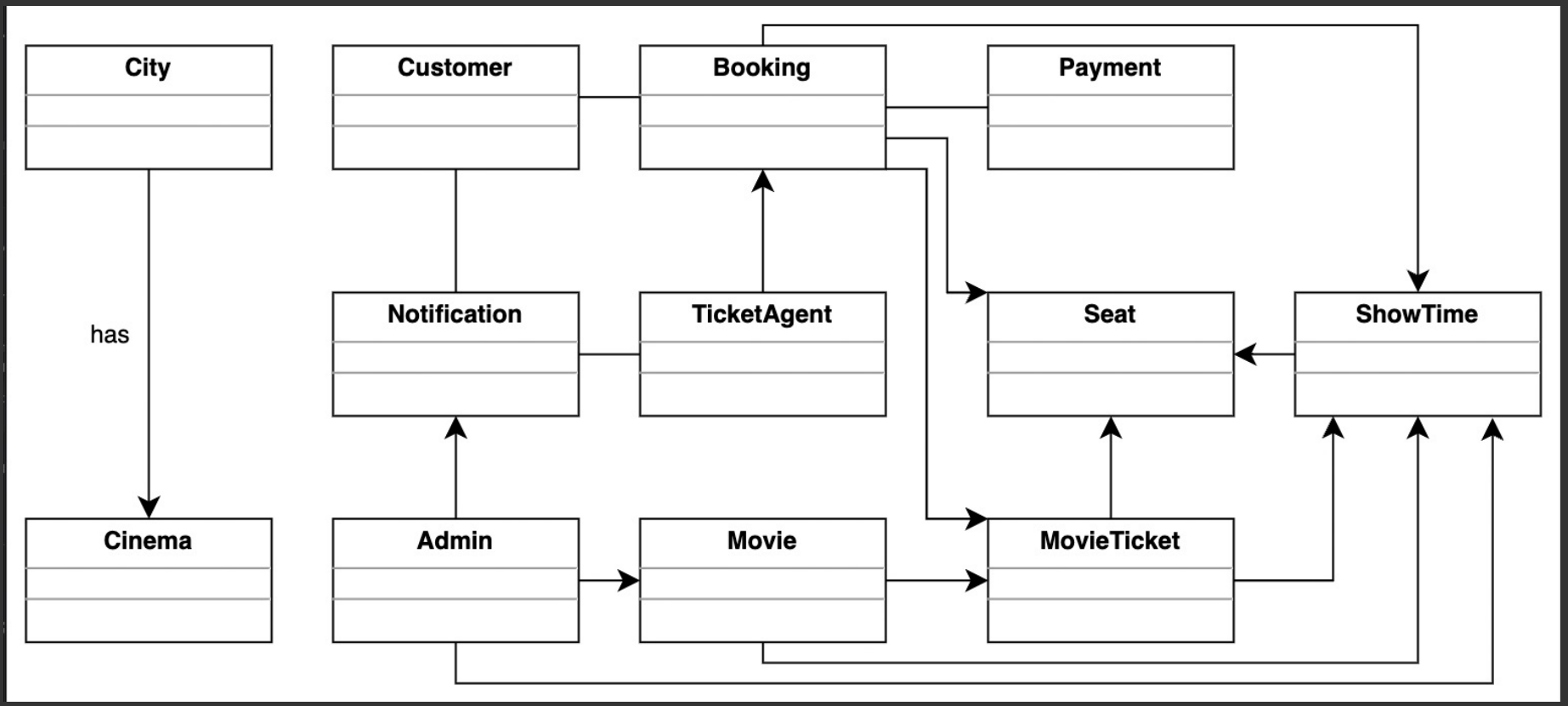
13. Booking

14. Enumerations

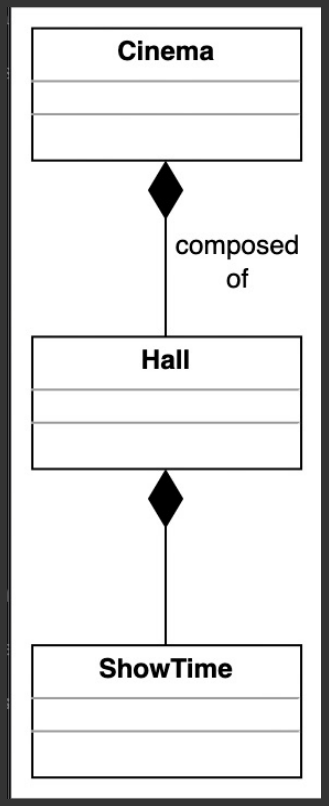




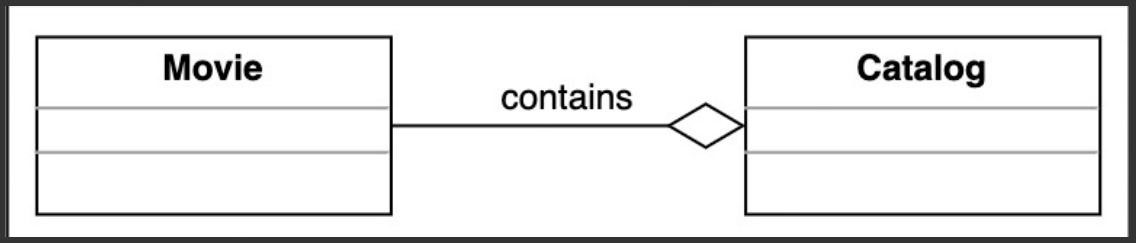
Associations



Composition

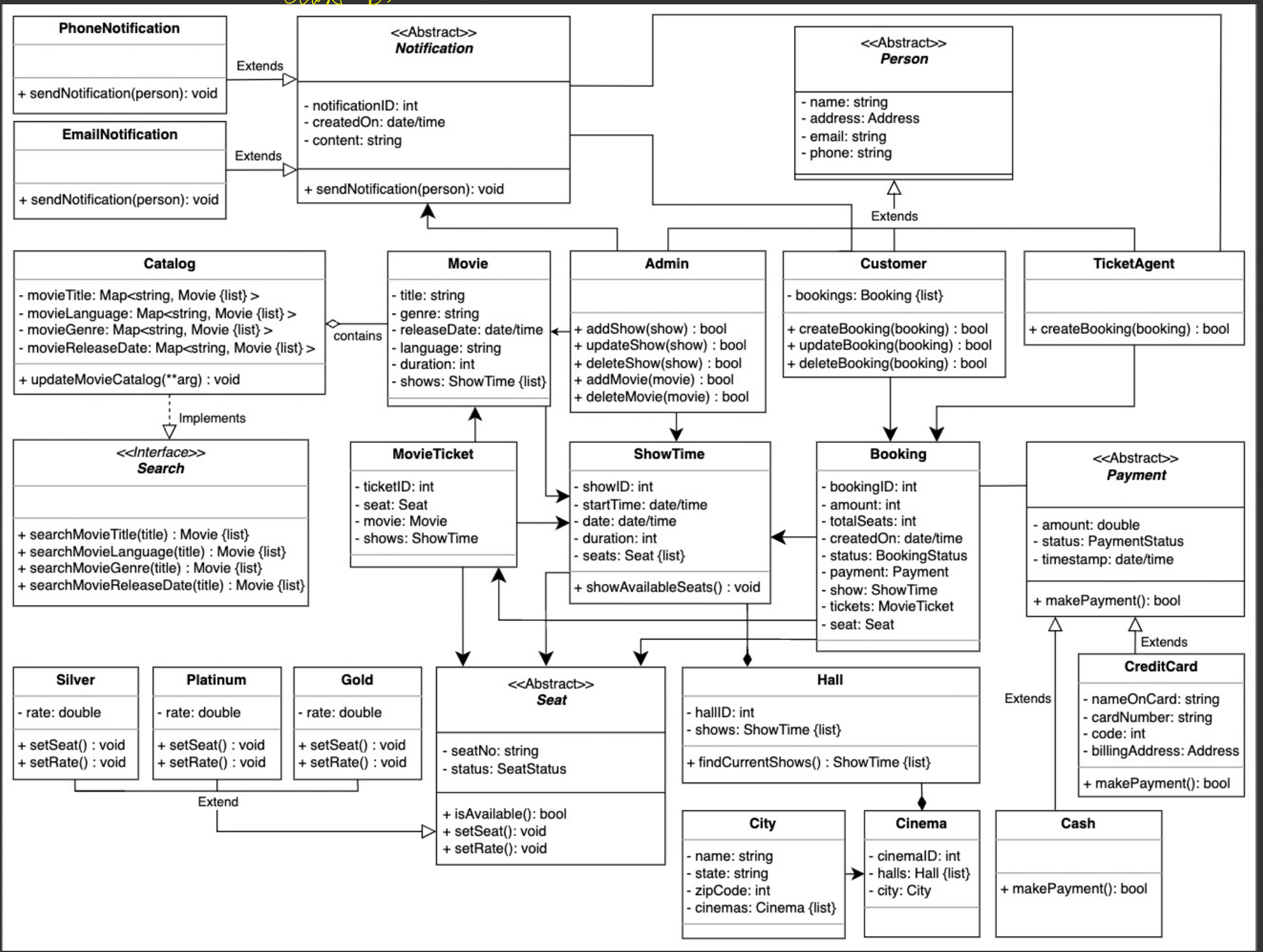


Aggregation



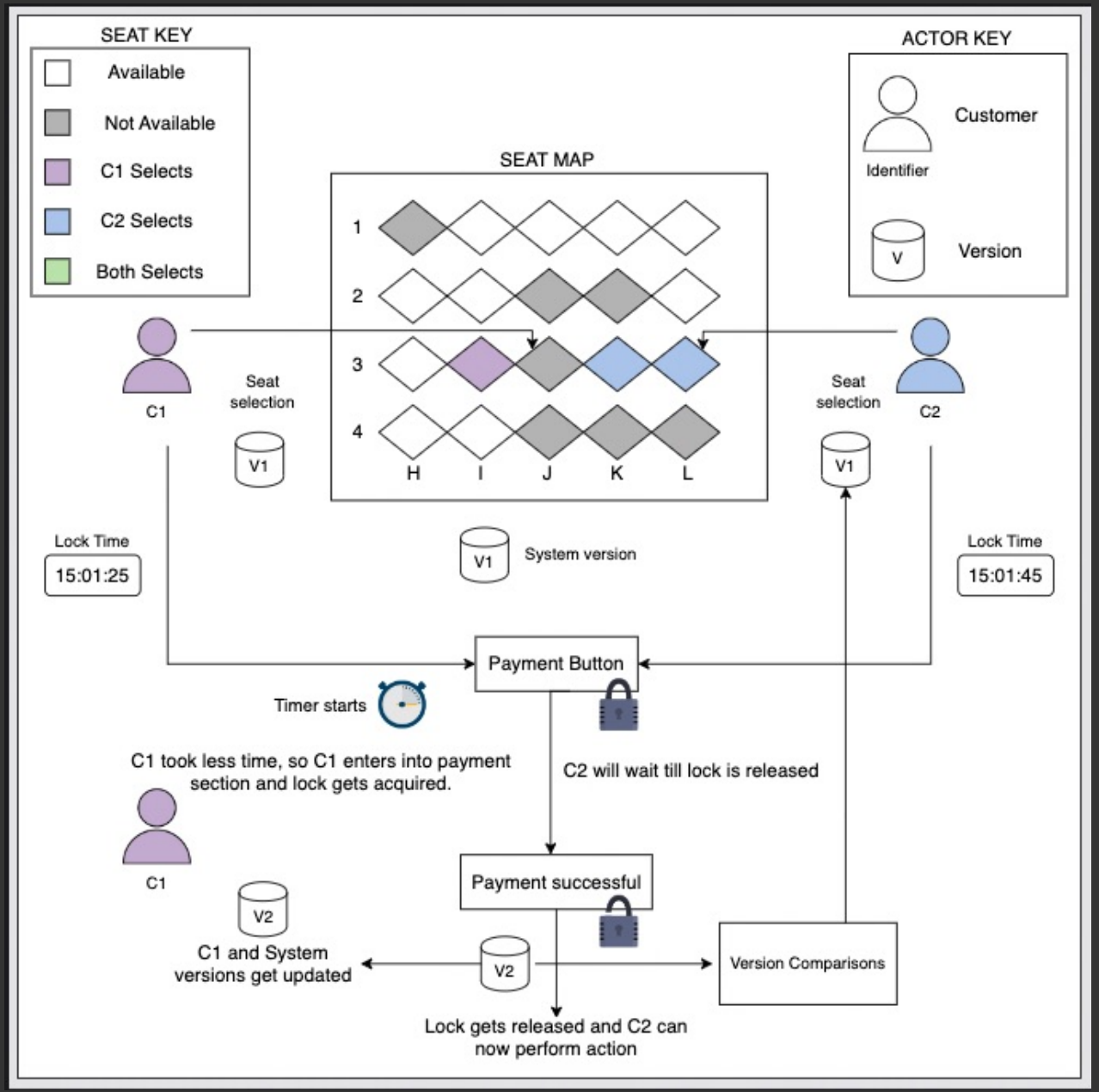
Generalization

Class Diagram



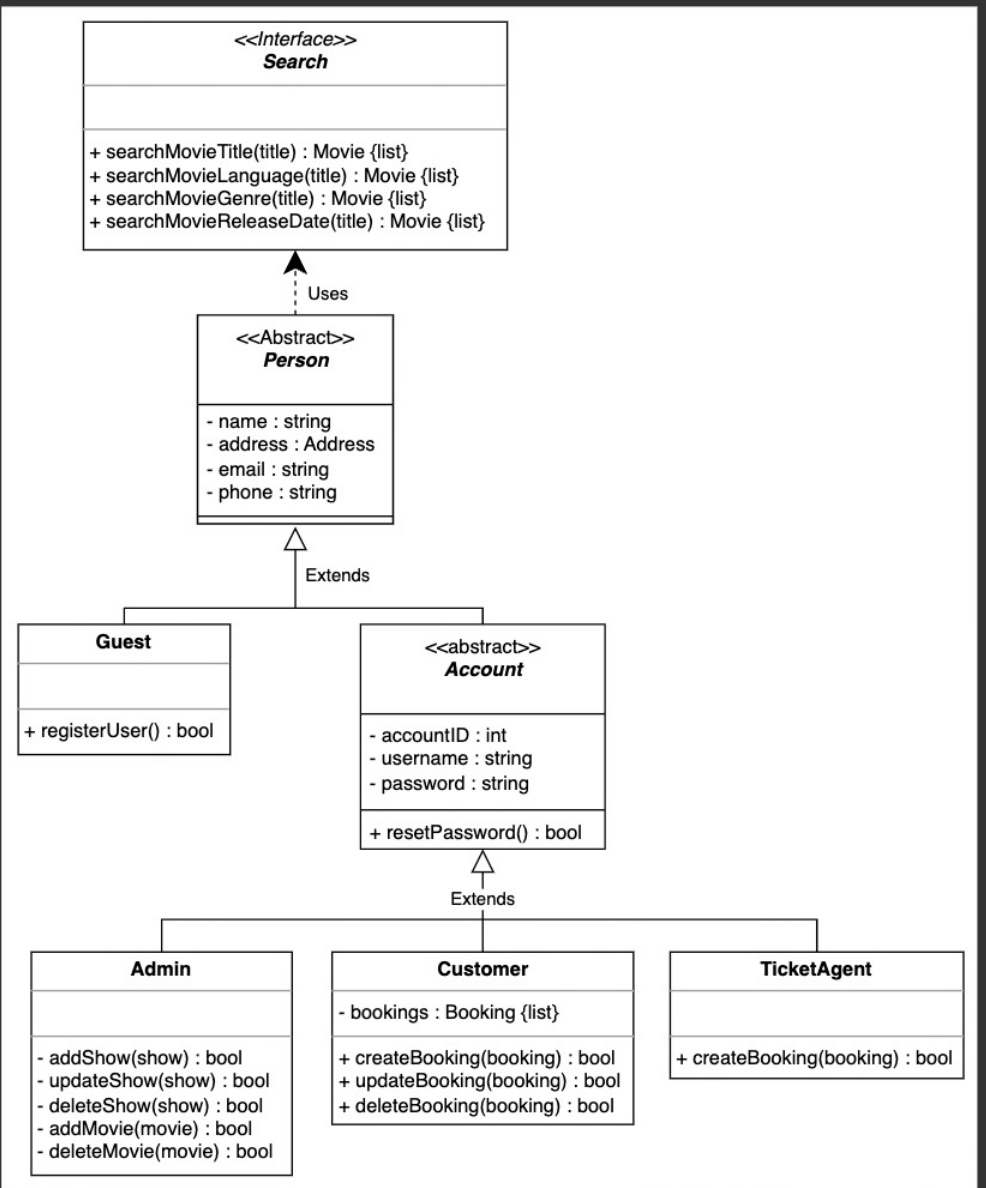
One of the Major Requirement no two customer can book the same seat

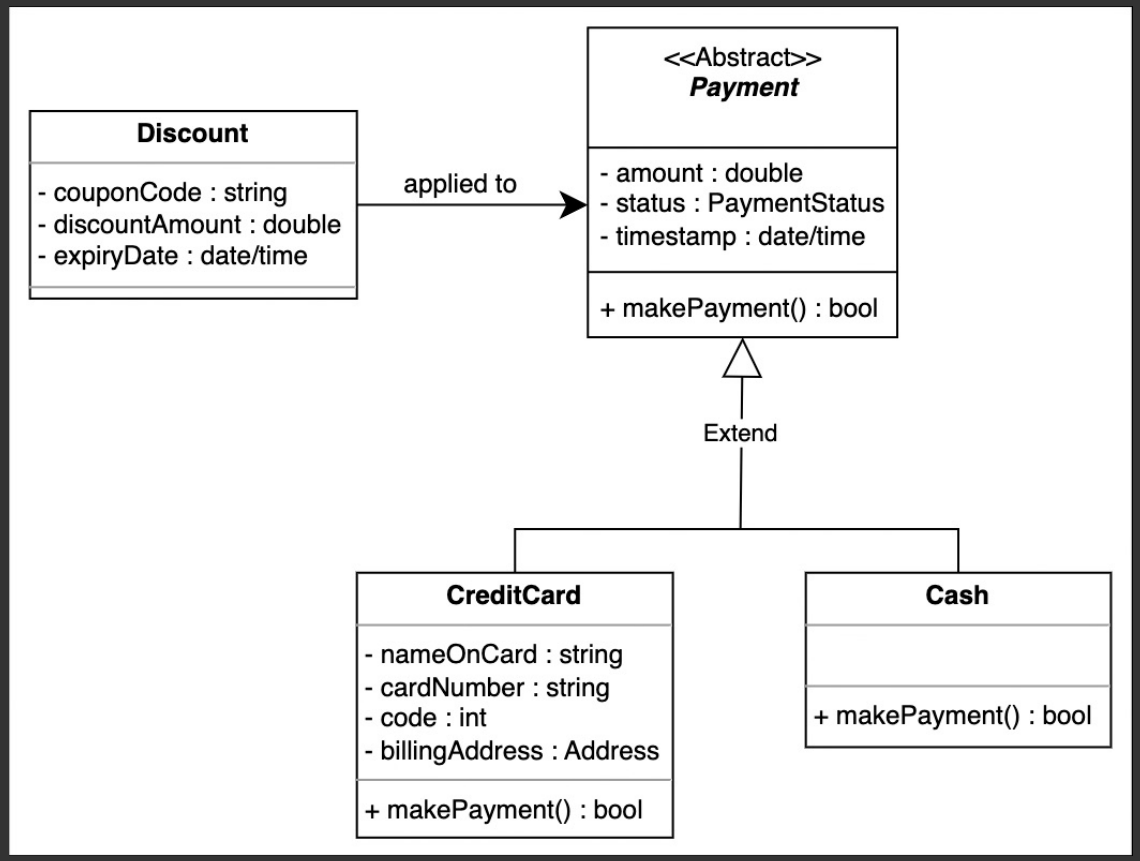
↳ To prevent such cases. → we will use lock.



Additional Requirement

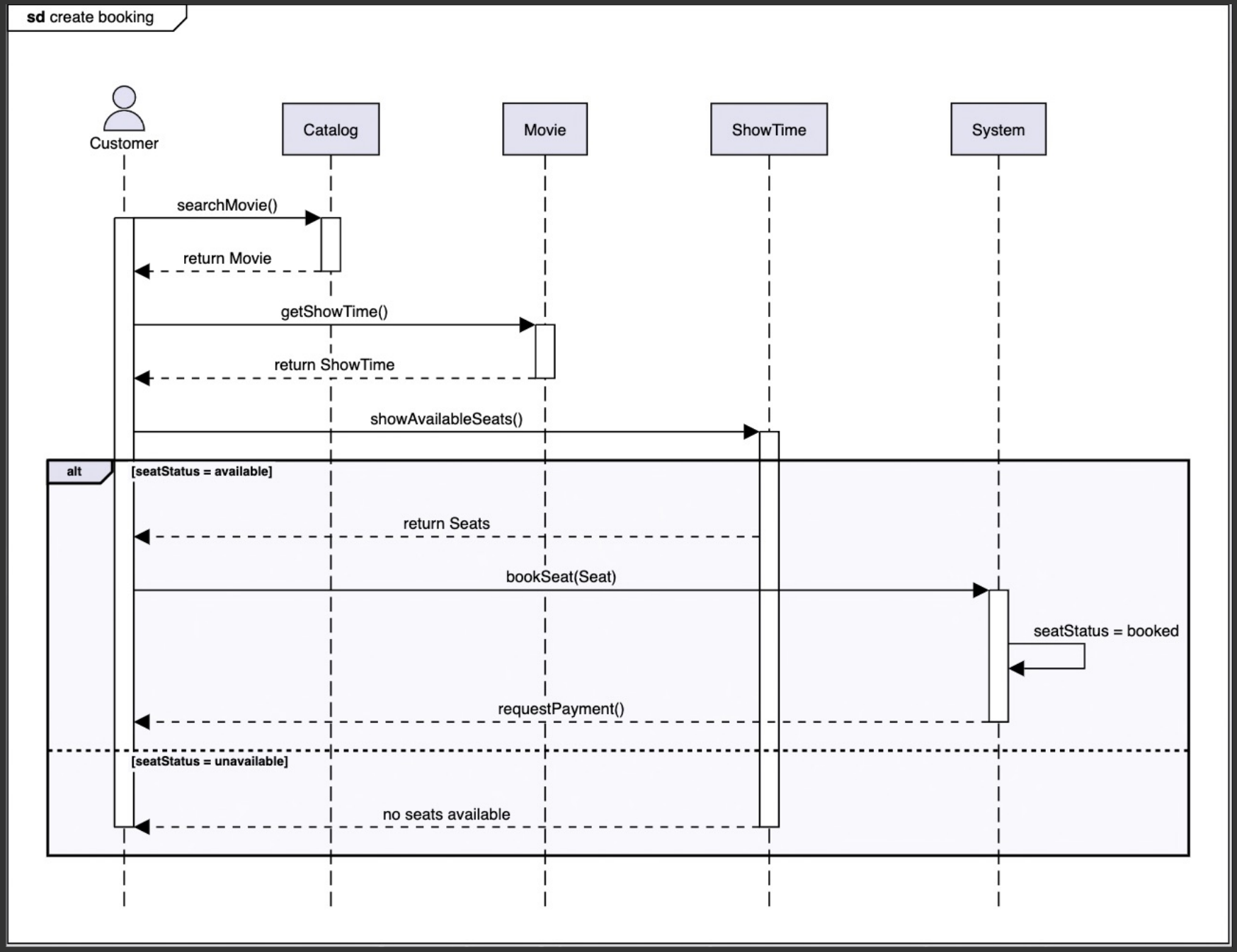
- Discount : add discount in payment



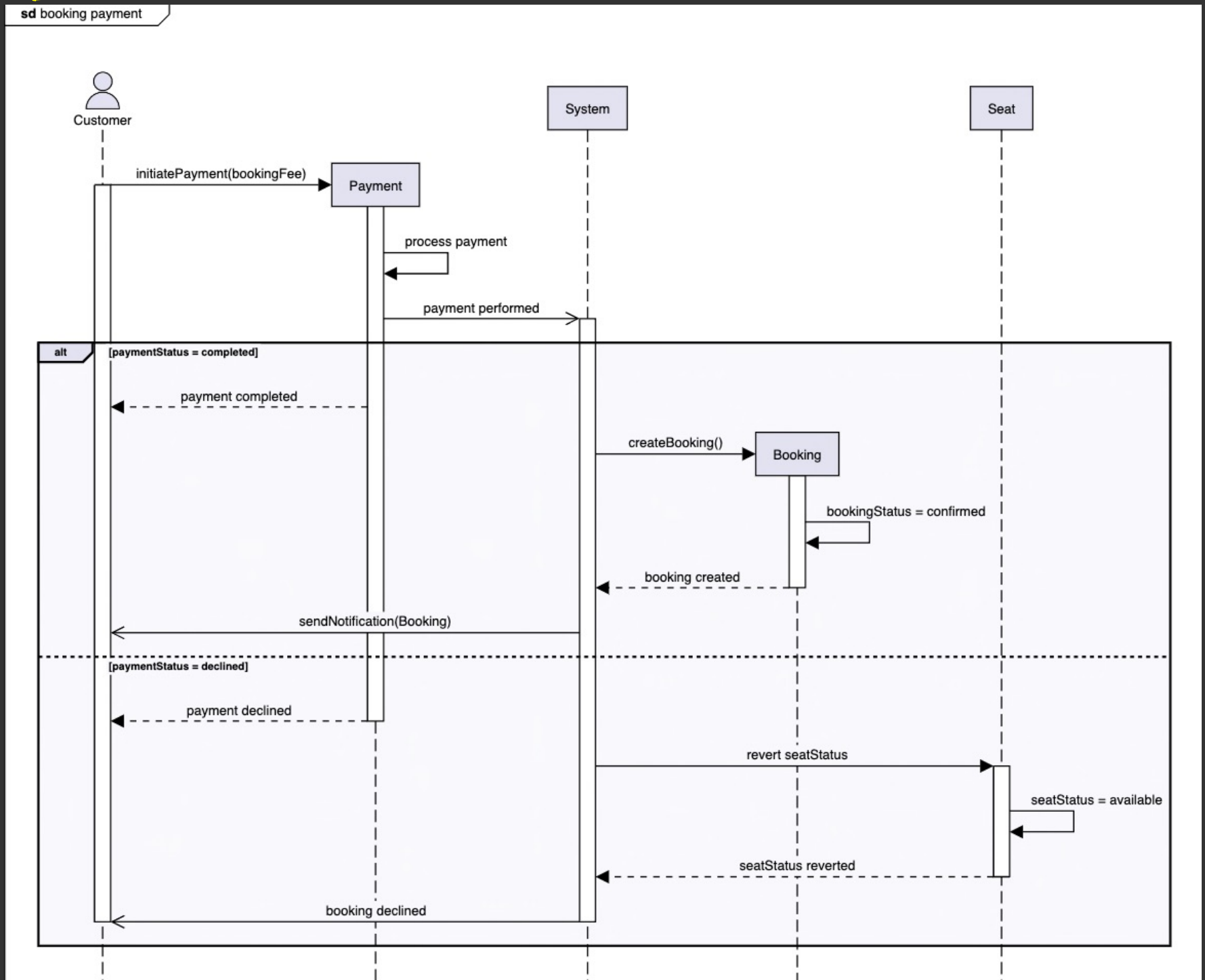


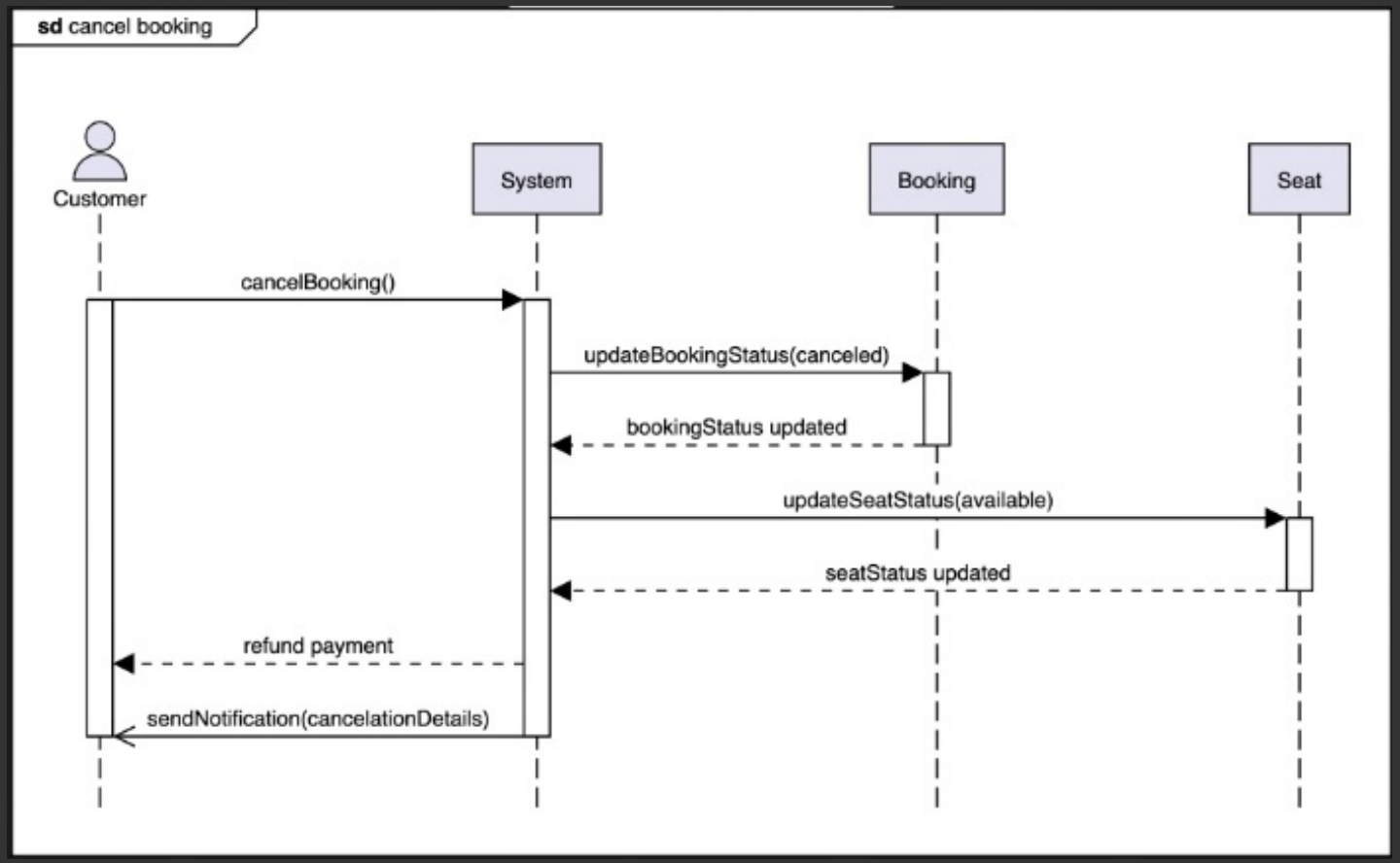
→ There can type of User guest who can search for movies & choose to register

Sequence Diagram of Create Booking

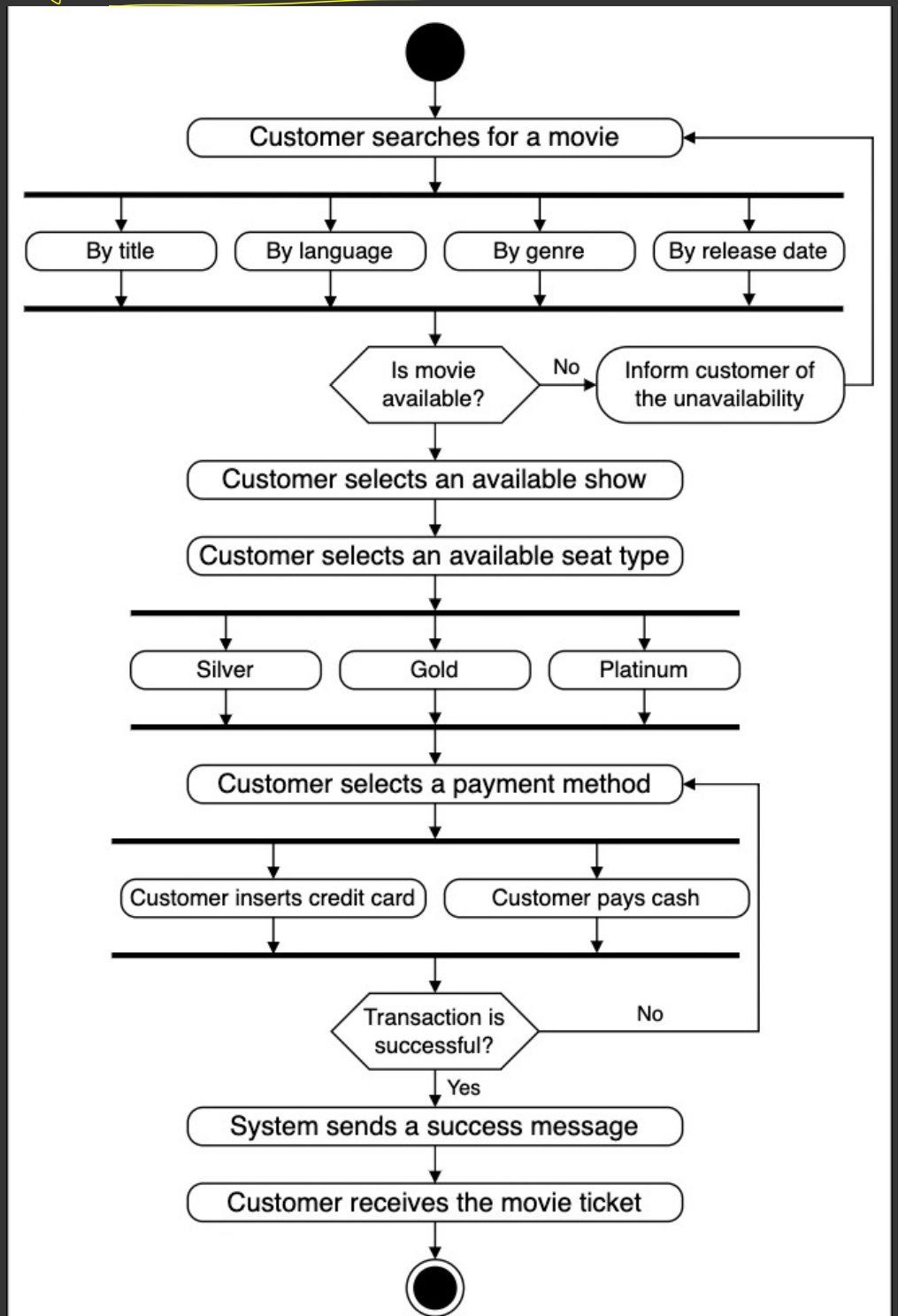


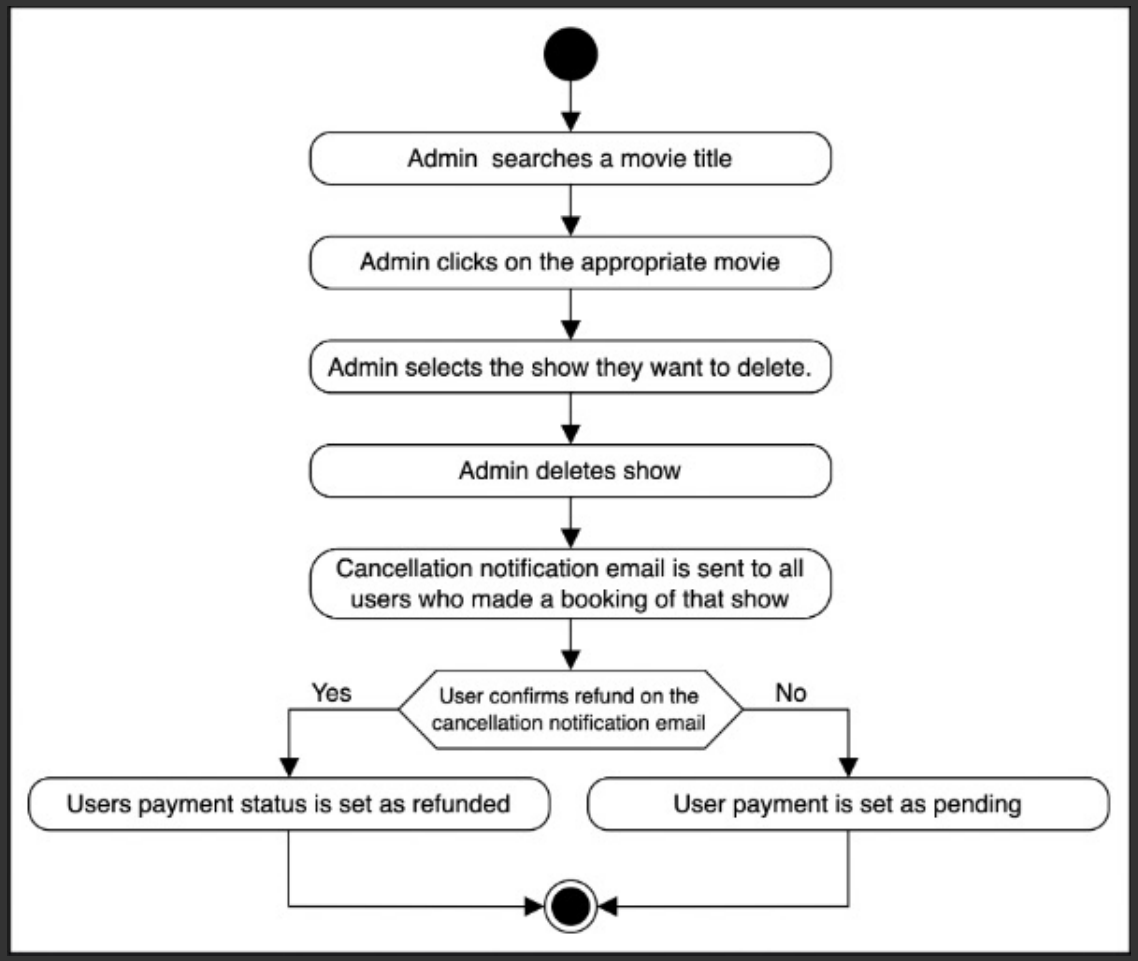
Sequence Diagram for booking Payment




sd cancel booking
Customer
System
Booking
Seat
cancelBooking()
updateBookingStatus(canceled)
bookingStatus updated
updateSeatStatus(available)
seatStatus updated
refund payment
sendNotification(cancelationDetails)

Activity Diagram of Customer making a booking




Admin searches a movie title
Admin clicks on the appropriate movie
Admin selects the show they want to delete.
Admin deletes show
Cancellation notification email is sent to all users who made a booking of that show
User confirms refund on the cancellation notification email
Yes
No
Users payment status is set as refunded
User payment is set as pending

Code for Movie Ticket Booking System

1. Enumerations

```
// Enumerations
enum PaymentStatus {
  PENDING,
  CONFIRMED,
  DECLINED,
  REFUNDED
}

enum BookingStatus {
  PENDING,
  CONFIRMED,
  CANCELLED,
  DENIED,
  REFUNDED
}

enum SeatStatus {
  AVAILABLE,
  BOOKED,
  RESERVED
}
```

2. Actors

```
// 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);
}
```

3. Seat

```
// 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
  }
}
```

4. Movie, Showtime & Movie Ticket

```
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;
}
```

5. City, Cinema & Hall

```
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();
}
```

6. Payment

```
// 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
    }
}
```

7. Notification Service

```
// 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
  }
}
```

8. Booking

```
public class Booking {
  // Data members
  private int bookingId;
  private int amount;
  private int totalSeats;

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

  // BookingStatus enum
  private BookingStatus
status;

  // Instances of classes
  private Payment payment;
  private ShowTime show;
  private List<MovieTicket>
tickets;
  private List<Seat> seats;
}
```

9. Search & Catlog

```
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
  }
}
```