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

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.

```
// Enumerations
enum PaymentStatus {
  UNPAID,
  PENDING,
  COMPLETED,
  DECLINED,
  REFUNDED,
  CANCELLED
}

enum BookingStatus {
  REQUESTED,
  PENDING,
  CONFIRMED,
  CANCELLED,
  REFUNDED,
  NONE
}

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

Enum definitions

Actors

This section contains the different people that will interact with our movie ticket systems, such as a `Customer`, `Admin`, and `TicketAgent`. 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<Booking> bookings; // List of bookings

  // booking here refers to an instance of the Booking class
  public boolean addNewBooking(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);
}
```

Actors involved in the movie ticket booking system

Seat

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

```
// Seat is an abstract class
public abstract class Seat {
  private int number;
  private String rowNo;
  private SeatStatus status; // Refers to the SeatStatus enum

  // Member Function
  public boolean isAvailable();
  public abstract void setSeat();
  public abstract void setRate();
}

public class Platinum extends Seat {
  public void setSeat() {
    // definition
  }
  public void setRate() {
    // definition
  }
}

public class Gold extends Seat {
  private double rate;
  public void setSeat() {
    // definition
  }
  public void setRate() {
    // definition
  }
}
```

Seat and its derived classes

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. The definition of these classes is given below:

```
public class Movie {
  // Data members
  private String title;
  private String genre;
  private Date releaseDate;
  private String language;
  private int duration;
  private List<Show> shows;
}

public class ShowTime {
  // Data members
  private int showId;

  // The Data 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;
}
```

The Movie, ShowTime, and MovieTicket classes

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 address;
  private int cityCode;
  private List<Cinema> cinemas;
}

public class Cinema {
  // Data members
  private String name;
  private String address;
  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();
}
```

The City, Cinema, and Hall classes

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 zip;

  public boolean makePayment() {
    // functionality
  }
}
```

Payment class and its child classes

Notification

The `NotificationLog` 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:

```
// Notification is an abstract class
public abstract class NotificationLog {
  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
  }
}
```

Notification class and its child classes

Booking

The `Booking` class is the main class of our movie ticket booking system and will display the information relating to a particular customer's booking. The definition of this class is given below:

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

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

  // BookingStatus enum
  private BookingStatus status;

  // Instance of classes
  private Payment payment;
  private List<Seat> reservedSeats;
  private List<Seat> seats;
}
```

The Booking class

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> searchMoviesByTitle(String title);
  public List<Movie> searchMoviesByLanguage(String language);
  public List<Movie> searchMoviesByGenre(String genre);
  public List<Movie> searchMoviesByReleaseDate(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> searchMoviesByTitle(String title) {
    // functionality
  }

  public List<Movie> searchMoviesByLanguage(String language) {
    // functionality
  }

  public List<Movie> searchMoviesByGenre(String genre) {
    // functionality
  }

  public List<Movie> searchMoviesByReleaseDate(Date date) {
    // functionality
  }
}
```

The Search interface and Catalog class

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.