BOOK M SHOW/ Ticket master

```
// Enums
enum SeatType { DELUX, VIP, ECONOMY, OTHER }
enum SeatStatus { BOOKED, AVAILABLE, RESERVED, NOT_AVAILABLE }
enum Genre { SCI_FI, DRAMA, ROM_COM, FANTASY }
enum BookingStatus { REQUESTED, PENDING, CONFIRMED, CANCELLED }
enum PaymentStatus { UNPAID, PENDING, COMPLETED, DECLINED, CANCELLED,
REFUNDED }

// Address
class Address {
    int pinCode;
    String street, city, state, country;
    public Address(int pinCode, String street, String city, String state,
String country) {
        this.pinCode = pinCode;
        this.street = street;
        this.city = city;
        this.state = state;
        this.country = country;
    }
}

// Movie
class Movie {
    String movieName;
    int movieId;
    int durationInMins;
    String language;
    Genre genre;
    Date releaseDate;
    Map<String, List<Show>> cityShowMap = new HashMap<>();
    public Movie(int movieId, String movieName, int durationInMins, String
language, Genre genre, Date releaseDate) {
        this.movieId = movieId;
        this.movieName = movieName;
        this.durationInMins = durationInMins;
        this.language = language;
        this.genre = genre;
        this.releaseDate = releaseDate;
    }
}

// Seat
class Seat {
    int seatId;
    SeatType seatType;
    SeatStatus seatStatus;
    Double price;
    public Seat(int seatId, SeatType seatType, Double price) {
        this.seatId = seatId;
        this.seatType = seatType;
        this.price = price;
        this.seatStatus = SeatStatus.AVAILABLE;
    }
}
```

```
// Show
class Show {
    int showId;
    Movie movie;
    Date startTime, endTime;
    CinemaHall cinemaPlayedAt;
    List<Seat> seats;
    public Show(int showId, Movie movie, Date startTime, Date endTime,
CinemaHall cinemaPlayedAt, List<Seat> seats) {
        this.showId = showId;
        this.movie = movie;
        this.startTime = startTime;
        this.endTime = endTime;
        this.cinemaPlayedAt = cinemaPlayedAt;
        this.seats = seats;
    }
}

// Audi
class Audi {
    int audiId;
    String audiName;
    int totalSeats;
    List<Show> shows = new ArrayList<>();
    public Audi(int audiId, String audiName, int totalSeats) {
        this.audiId = audiId;
        this.audiName = audiName;
        this.totalSeats = totalSeats;
    }
}

// CinemaHall
class CinemaHall {
    int cinemaHallId;
    String cinemaHallName;
    Address address;
    List<Audi> audiList = new ArrayList<>();
    public CinemaHall(int cinemaHallId, String cinemaHallName, Address
address) {
        this.cinemaHallId = cinemaHallId;
        this.cinemaHallName = cinemaHallName;
        this.address = address;
    }
    public Map<Date, Movie> getMovies(List<Date> dateList) {
        Map<Date, Movie> map = new HashMap<>();
        for (Audi audi : audiList) {
            for (Show show : audi.shows) {
                if (dateList.contains(show.startTime)) {
                    map.put(show.startTime, show.movie);
                }
            }
        }
        return map;
    }
    public Map<Date, Show> getShows(List<Date> dateList) {
        Map<Date, Show> map = new HashMap<>();
        for (Audi audi : audiList) {
            for (Show show : audi.shows) {
                if (dateList.contains(show.startTime)) {
```

```
                    map.put(show.startTime, show);
                }
            }
        }
        return map;
    }
}

// Account
class Account {
    String userName, password;
    public Account(String userName, String password) {
        this.userName = userName;
        this.password = password;
    }
}

// Search
class Search {
    List<Show> shows;
    List<Movie> movies;
    public Search(List<Show> shows, List<Movie> movies) {
        this.shows = shows;
        this.movies = movies;
    }
    public List<Show> searchMoviesByNames(String name) {
        List<Show> result = new ArrayList<>();
        for (Show show : shows) {
            if (show.movie.movieName.equalsIgnoreCase(name)) {
                result.add(show);
            }
        }
        return result;
    }
    public List<Movie> searchMoviesByGenre(Genre genre) {
        List<Movie> result = new ArrayList<>();
        for (Movie m : movies) {
            if (m.genre == genre) result.add(m);
        }
        return result;
    }
    public List<Movie> searchMoviesByLanguage(String language) {
        List<Movie> result = new ArrayList<>();
        for (Movie m : movies) {
            if (m.language.equalsIgnoreCase(language)) result.add(m);
        }
        return result;
    }
    public List<Movie> searchMoviesByDate(Date releaseDate) {
        List<Movie> result = new ArrayList<>();
        for (Movie m : movies) {
            if (m.releaseDate.equals(releaseDate)) result.add(m);
        }
        return result;
    }
}
```

```
// User
class User {
    int userId;
    Search searchObj;
    public User(int userId, Search searchObj) {
        this.userId = userId;
        this.searchObj = searchObj;
    }
}

// SystemMember
class SystemMember extends User {
    Account account;
    String name, email;
    Address address;
    public SystemMember(int userId, Search searchObj, Account account,
String name, String email, Address address) {
        super(userId, searchObj);
        this.account = account;
        this.name = name;
        this.email = email;
        this.address = address;
    }
}

// Member
class Member extends SystemMember {
    List<Booking> bookings = new ArrayList<>();
    public Member(int userId, Search searchObj, Account account, String
name, String email, Address address) {
        super(userId, searchObj, account, name, email, address);
    }
    public Booking makeBooking(Booking booking) {
        bookings.add(booking);
        return booking;
    }
    public List<Booking> getBooking() {
        return bookings;
    }
}

// Admin
class Admin extends SystemMember {
    public Admin(int userId, Search searchObj, Account account, String name,
String email, Address address) {
        super(userId, searchObj, account, name, email, address);
    }
    public boolean addMovie(Movie movie) { return true; }
    public boolean addShow(Show show) { return true; }
}

// Payment
class Payment {
    double amount;
    Date paymentDate;
    int transactionId;
    PaymentStatus paymentStatus;
    public Payment(double amount, Date paymentDate, int transactionId,
PaymentStatus paymentStatus) {
```

```
        this.amount = amount;
        this.paymentDate = paymentDate;
        this.transactionId = transactionId;
        this.paymentStatus = paymentStatus;
    }
}

// Booking
class Booking {
    String bookingId;
    Date bookingDate;
    Member member;
    Audi audi;
    Show show;
    BookingStatus bookingStatus;
    double totalAmount;
    List<Seat> seats;
    Payment paymentObj;
    public Booking(String bookingId, Date bookingDate, Member member, Audi
audi, Show show, List<Seat> seats) {
        this.bookingId = bookingId;
        this.bookingDate = bookingDate;
        this.member = member;
        this.audi = audi;
        this.show = show;
        this.seats = seats;
        this.bookingStatus = BookingStatus.REQUESTED;
        this.totalAmount = seats.stream().mapToDouble(s -> s.price).sum();
    }
    public boolean makePayment(Payment payment) {
        this.paymentObj = payment;
        if (payment.paymentStatus == PaymentStatus.COMPLETED) {
            this.bookingStatus = BookingStatus.CONFIRMED;
            for (Seat seat : seats) seat.seatStatus = SeatStatus.BOOKED;
            return true;
        }
        return false;
    }
}

// BMSService1
class BMSService2 {
    List<CinemaHall> cinemas = new ArrayList<>();
    List<Movie> movies = new ArrayList<>();
    public List<Movie> getMovies(Date date, String city) {
        List<Movie> result = new ArrayList<>();
        for (CinemaHall hall : cinemas) {
            if (hall.address.city.equalsIgnoreCase(city)) {
                for (Audi audi : hall.audiList) {
                    for (Show show : audi.shows) {
                        if (sameDay(show.startTime, date)) {
                            if (!result.contains(show.movie))
result.add(show.movie);
                        }
                    }
                }
            }
        }
        return result;
```

```
    }
    public List<CinemaHall> getCinemalHalls(String city) {
        List<CinemaHall> result = new ArrayList<>();
        for (CinemaHall hall : cinemas) {
            if (hall.address.city.equalsIgnoreCase(city)) result.add(hall);
        }
        return result;
    }
    private boolean sameDay(Date d1, Date d2) {
        SimpleDateFormat fmt = new SimpleDateFormat("yyyyMMdd");
        return fmt.format(d1).equals(fmt.format(d2));
    }
}


/*
 *  End-to-End Use Case Demo
 */

public class Main {
    public static void main(String[] args) throws Exception {
        // Setup
        BMSService1 bms = new BMSService1();

        // Create CinemaHalls
        Address addr1 = new Address(110001, "Connaught Place", "Delhi",
"Delhi", "India");
        Address addr2 = new Address(400001, "Marine Drive", "Mumbai",
"Maharashtra", "India");
        CinemaHall delhiHall = new CinemaHall(1, "PVR Plaza", addr1);
        CinemaHall mumbaiHall = new CinemaHall(2, "INOX Marine", addr2);

        // Add Audis
        Audi delhiAudi1 = new Audi(1, "Delhi Audi 1", 20);
        Audi delhiAudi2 = new Audi(2, "Delhi Audi 2", 15);
        Audi mumbaiAudi1 = new Audi(3, "Mumbai Audi 1", 25);
        delhiHall.audiList.add(delhiAudi1);
        delhiHall.audiList.add(delhiAudi2);
        mumbaiHall.audiList.add(mumbaiAudi1);

        bms.cinemas.add(delhiHall);
        bms.cinemas.add(mumbaiHall);

        // Create Movies
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm");
        Movie movie1 = new Movie(1, "Interstellar", 169, "English",
Genre.SCI_FI, sdf.parse("2024-07-01 00:00"));
        Movie movie2 = new Movie(2, "Yeh Jawaani Hai Deewani", 140, "Hindi",
Genre.ROM_COM, sdf.parse("2013-05-31 00:00"));
        bms.movies.add(movie1);
        bms.movies.add(movie2);
 // Create Shows
        List<Seat> seatsDelhi1 = new ArrayList<>();
        for (int i = 1; i <= delhiAudi1.totalSeats; i++)
            seatsDelhi1.add(new Seat(i, SeatType.ECONOMY, 200.0));
        Show show1 = new Show(1, movie1, sdf.parse("2025-07-05 18:00"),
sdf.parse("2025-07-05 21:00"), delhiHall, seatsDelhi1);
        delhiAudi1.shows.add(show1);
```

```
        List<Seat> seatsDelhi2 = new ArrayList<>();

        for (int i = 1; i <= delhiAudi2.totalSeats; i++)
            seatsDelhi2.add(new Seat(i, SeatType.VIP, 400.0));
        Show show2 = new Show(2, movie2, sdf.parse("2025-07-05 20:00"),
sdf.parse("2025-07-05 22:30"), delhiHall, seatsDelhi2);
        delhiAudi2.shows.add(show2);

        // Create Member
        Account acc = new Account("john_doe", "password123");
        Member member = new Member(1, new Search(Arrays.asList(show1,
show2), bms.movies), acc, "John Doe", "john@example.com", addr1);


 // Member books 2 tickets for show1
        List<Seat> seatsToBook = new ArrayList<>();
        seatsToBook.add(show1.seats.get(0)); // Seat 1
        seatsToBook.add(show1.seats.get(1)); // Seat 2
        Booking booking = new Booking("BKG1", new Date(), member,
delhiAudi1, show1, seatsToBook);

        // Make payment
        Payment payment = new Payment(400.0, new Date(), 1001,
PaymentStatus.COMPLETED);
        booking.makePayment(payment);

        // Add booking to member
        member.makeBooking(booking);

        // Output
        System.out.println("Booking Details:");
        System.out.println("Member: " + member.name);
        System.out.println("Movie: " + show1.movie.movieName);
        System.out.println("Show Time: " + sdf.format(show1.startTime));
        System.out.println("Seats Booked: " + seatsToBook.get(0).seatId + ",
" + seatsToBook.get(1).seatId);
        System.out.println("Booking Status: " + booking.bookingStatus);
        System.out.println("Payment Status: " +
booking.paymentObj.paymentStatus);
        System.out.println("Total Amount: " + booking.totalAmount);
    }
}

/*output Example
 * Booking Details:
Member: John Doe
Movie: Interstellar
Show Time: 2025-07-05 18:00
Seats Booked: 1, 2
Booking Status: CONFIRMED
Payment Status: COMPLETED
Total Amount: 400.0
 */
```