GIC CINEMAS - SEAT BOOKING APP

Here's a concise summary on how to run the SeatBookingApp class to perform new bookings, modify bookings, and view existing bookings:

Running the Application

1. **Start the Application**:
    - Run the SeatBookingApp class. This initializes the application, prompting you to enter movie details and seating chart configuration.
    - Input format: [title] [rows] [seatsPerRow].
2. **New Booking**:
    - **Select Option**: Choose option [1] Book Tickets.
    - **Enter Number of Tickets**: Input the number of tickets you want to book. The application will attempt to find and book the specified number of seats.
    - **Review Booking**: After booking, you'll receive a booking ID. You can either accept the booking or modify the seat selection.
3. **Modify Booking**:
    - **Select Option**: After receiving the booking ID, if you want to modify it, you can enter a new starting position for the seat booking or leave it blank to accept the current selection.
    - **Provide New Starting Position**: Enter the new seat starting position (e.g., "B5"). The application will attempt to adjust the booking based on the new position.
    - **Confirm or Adjust**: Review the modified seat selection and confirm or make further adjustments.
4. **View Booking**:
    - **Select Option**: Choose option [2] Check Bookings.
    - **Enter Booking ID**: Input the booking ID you want to view.
    - **Review Booking**: The application will display the seating chart with the booked seats highlighted.

Summary of Actions

1. **Run Application**: SeatBookingApp.main()
2. **For Booking**: Select [1] Book Tickets, input the number of tickets.
3. **For Modification**: Provide new seat position if prompted after booking.
4. **For Viewing**: Select [2] Check Bookings, enter the booking ID

CLASS DESIGN

Here's a brief overview with a short description and purpose for each class:

1. **BookingException**:
    - **Description**: Custom runtime exception for booking errors.
    - **Purpose**: To handle and signal errors related to booking operations.
2. **BookingManagerImpl**:
    - **Description**: Implementation of the BookingManager interface.
    - **Purpose**: Manages bookings by creating, retrieving, and modifying them, using a ConcurrentHashMap for storage and an AtomicInteger for unique booking ID generation.
3. **Constants**:
    - **Description**: Utility class containing static final constants.
    - **Purpose**: Provides application-wide constants for error messages, booking ID formats, and user prompts to ensure consistency and ease of maintenance.
4. **Movie**:
    - **Description**: Record representing a movie.
    - **Purpose**: Holds the title of a movie to be used in booking contexts.
5. **Seat**:
    - **Description**: Represents a seat with row and column.
    - **Purpose**: To model individual seats in the cinema, including methods for equality and hashing.
6. **SeatManagerImpl**:
    - **Description**: Implementation of the SeatManager interface.
    - **Purpose**: Manages the seating chart, including booking and clearing seats, and providing available seat information.
7. **SeatBookingApp**:
    - **Description**: Main application class handling user interaction.
    - **Purpose**: Manages the application flow, including seat booking, viewing, and modification operations, and interacts with users through console input and output.
8. **SeatState**:
    - **Description**: Enum representing the state of a seat.

- **Purpose**: Defines seat states (AVAILABLE, BOOKED, TEMP_BOOKED) with associated symbols for visual representation in the seating chart.