**Project Title 1: Dynamic Ticket Reservation and Cancellation System**

**1. Problem Description:**

The project aims to develop a dynamic ticket reservation and cancellation system in Java that can efficiently manage seat bookings, cancellations, and undo actions. Seats are categorized (e.g., VIP, Economy) and stored in a searchable structure for quick allocation.

Key features include:

**1.1:Seat booking** and **cancellation** based on seat number and category.

**1.2:Undo functionality** for the last booking or cancellation

**1.3:Fast lookup** of seat details like availability and customer name.

1. The system must support **scalability** for thousands of seats, provide **real-time updates**, and ensure **data integrity** using appropriate data structures such as **AVL Trees**, **HashMaps**, and **Stacks**.
2. **Data Structures Justification:**

**2.1:**  For **seat catogry** , An Binary Serach tree is Used. Each node contains catogry and catogryNode that has price and No. Of available seats.

> BST enables us for Log(N) complexity for inseart, check seat Catogry.

**2.2:** For **Seat Object** , Hash Map is used for each seat No. Eacg hash map Node store hashcode of SeatNo. And seat object

>  **Hash Map** enables Us to have **Constant Time Access** to Seat Lookup, it Availabiltiy, it’s details.

**2.3** For **Booking Seat**, Two separate stacks manage undo operations. BookingStack: Records each booking for undoing reservations. CancellationStack: Records each cancellation for undoing a cancellation.

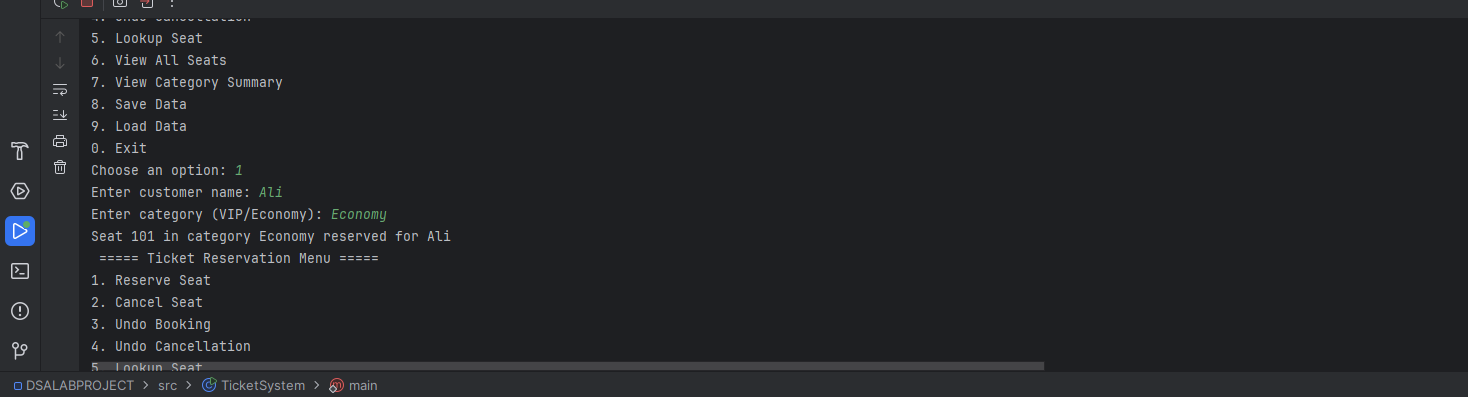
**Stack** work on the principal of **LIFO** , it enables us to **efficiently pop and push** cancelations and Booking.

**4. Full Explanation of Algorithms**

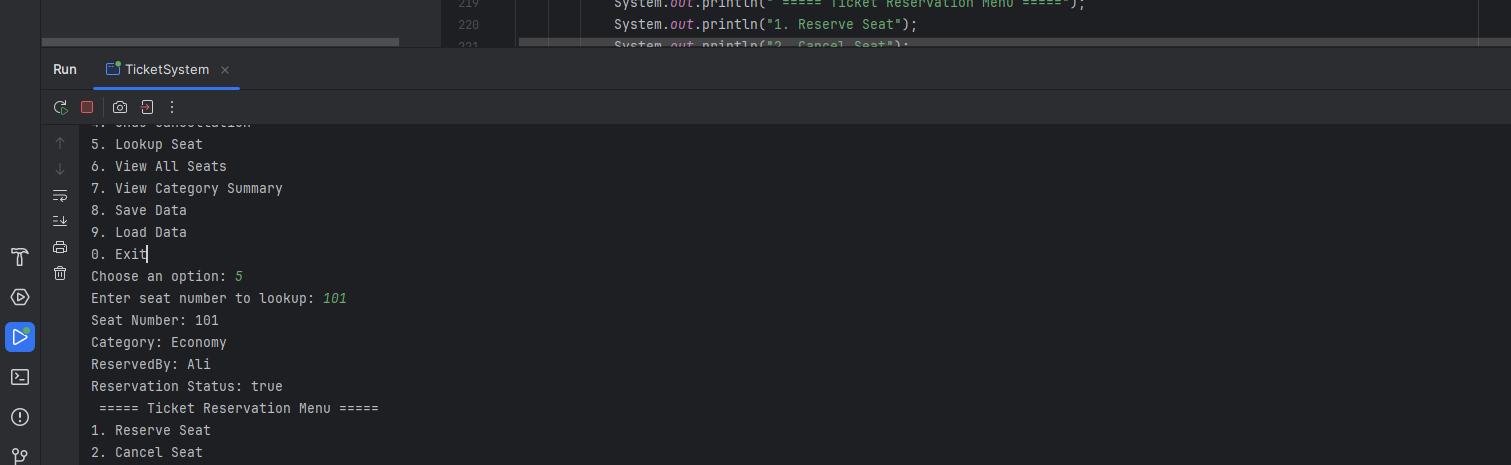
**4.1 Booking a Seat**  
The system locates and reserves the first available seat in the requested category.

4.1.1 Iterate through the TreeSet<Seat>, which maintains seats sorted by category and seat number.  
4.1.2 For each seat, check if it matches the requested category and is not reserved.  
4.1.3 Once a matching seat is found:  
 4.1.3.1 Mark the seat as reserved.  
 4.1.3.2 Set the customer name.  
 4.1.3.3 Push the seat into the bookingStack for undo functionality.  
 4.1.3.4 Decrease the available seat count in the corresponding CategoryNode.  
4.1.4 This approach ensures efficient lookup using TreeSet ordering and fast updates using HashMap for category metadata.

Booking:

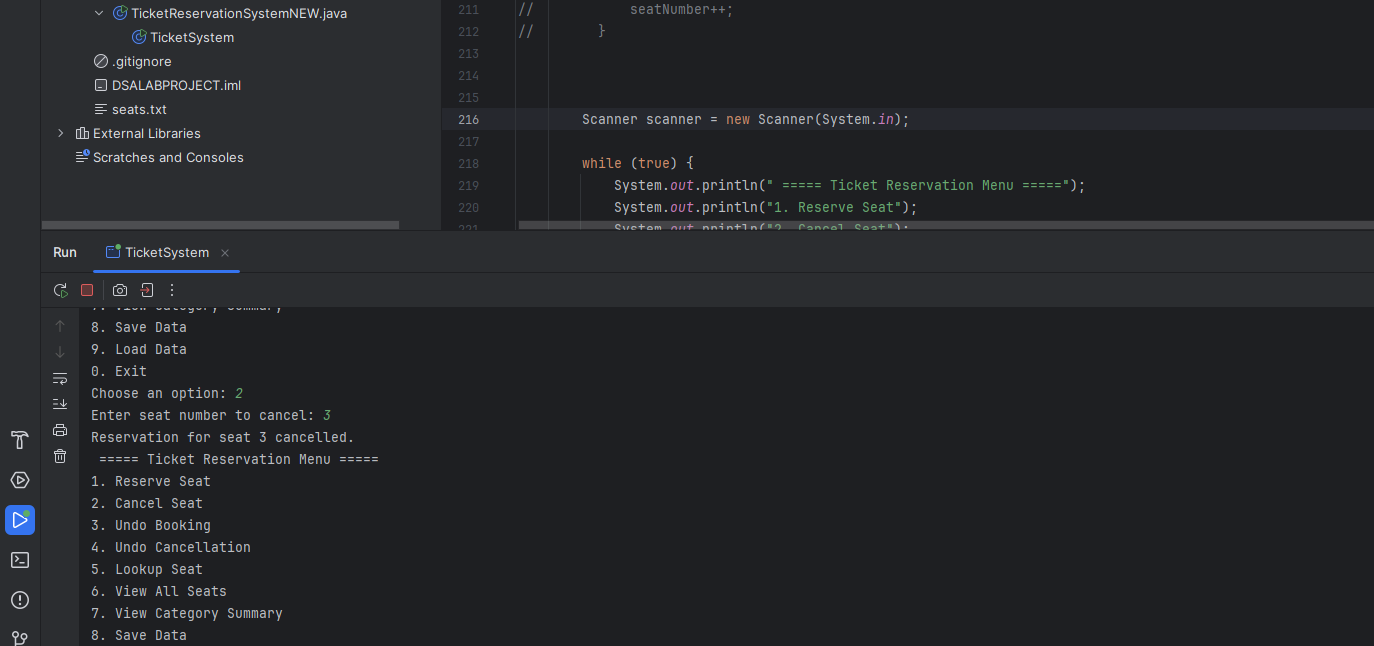


Lookup:



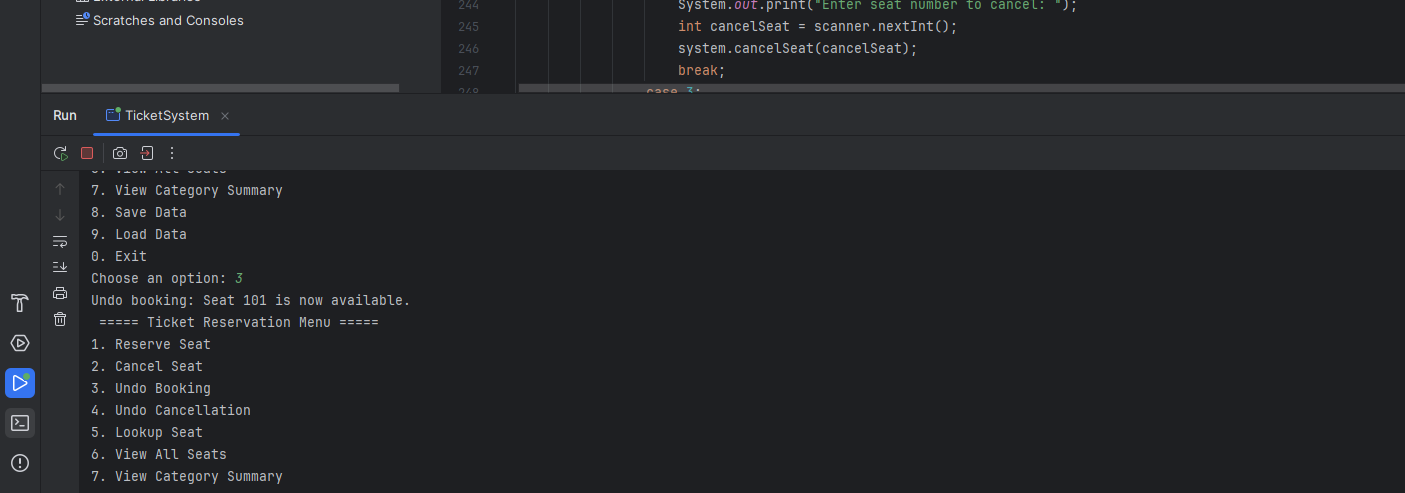
**4.2 Cancelling a Seat**  
A reserved seat can be cancelled by the user.

4.2.1 Use seatMap (HashMap) to retrieve the seat by seat number in O(1) time.  
4.2.2 If the seat is reserved:  
 4.2.2.1 Mark the seat as unreserved.  
 4.2.2.2 Clear the customer name.  
 4.2.2.3 Push the seat into the cancellationStack for undo functionality.  
 4.2.2.4 Increase the available seat count in the relevant CategoryNode.



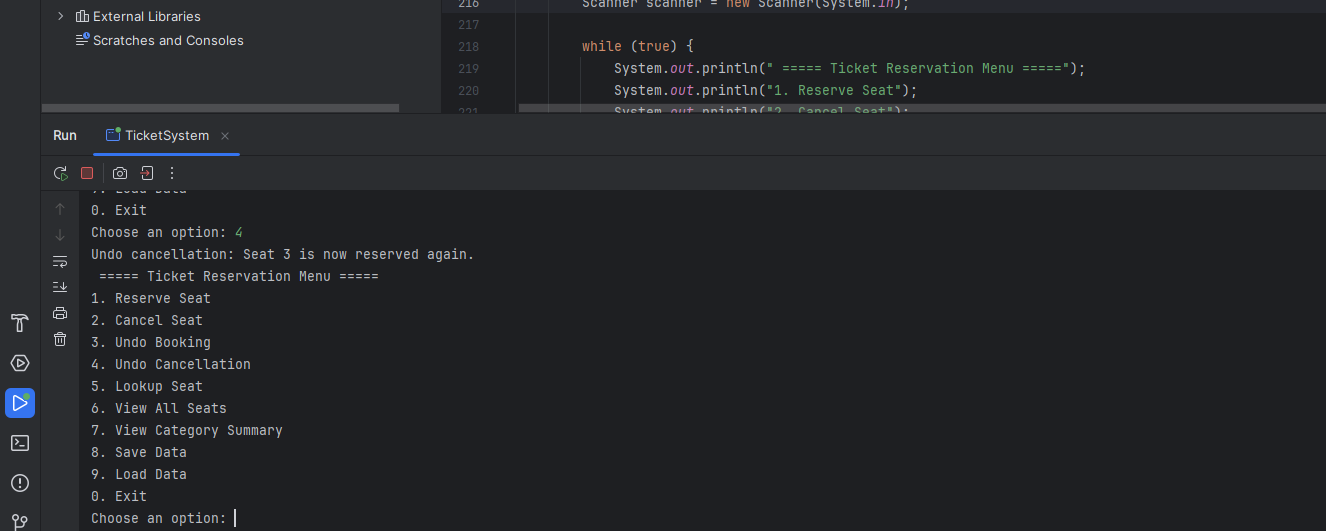
**4.3 Undo Booking**  
Reverts the most recent seat booking action.

4.3.1 Pop the last seat from the bookingStack.  
4.3.2 Restore the seat’s status:  
 4.3.2.1 Mark it as unreserved.  
 4.3.2.2 Clear the customer name.  
 4.3.2.3 Increase the category’s available seat count.



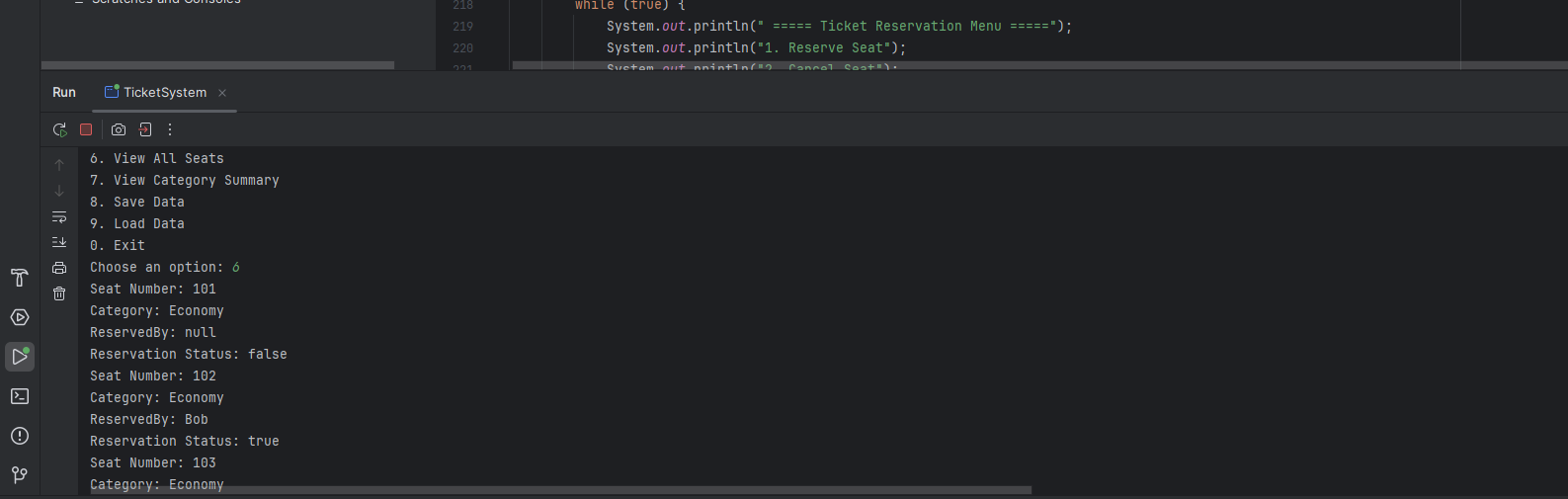
**4.4 Undo Cancellation**  
Restores the most recent seat that was cancelled.

4.4.1 Pop the last seat from the cancellationStack.  
4.4.2 Reapply the booking:  
 4.4.2.1 Mark the seat as reserved.  
 4.4.2.2 Restore the customer name.  
 4.4.2.3 Decrease the category’s available seat count.



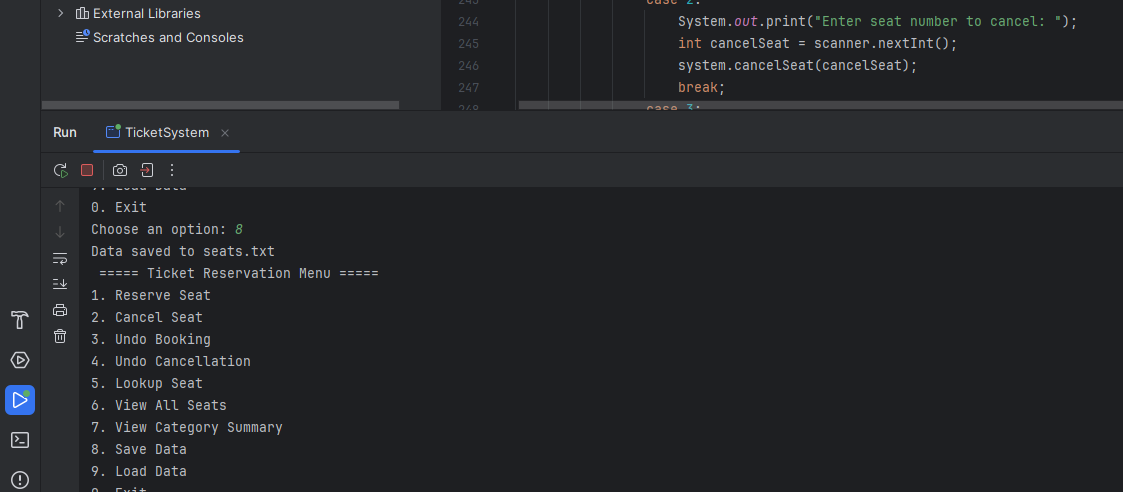
**4.5 Lookup Seat**  
Retrieves the status and details of a specific seat.

4.5.1 Use seatMap.get(seatNumber) to fetch seat details in O(1) time.  
4.5.2 Display seat number, category, reservation status, and customer name (if reserved).



**4.6 Save to File**  
Persists all seat data to a file for future use.

4.6.1 Traverse the seats collection.  
4.6.2 For each seat, write its number, category, reservation status, and customer name to a file line by line.

****

**4.7 Load from File**  
Initializes the system using previously saved data.

4.7.1 Clear all current data structures including seats, maps, and stacks.  
4.7.2 Read the saved file line by line.  
4.7.3 For each seat entry:  
 4.7.3.1 Parse the seat number, category, reservation status, and customer name.  
 4.7.3.2 Create a Seat object and add it to seats and seatMap.  
4.7.4 Recalculate available seat counts per category to reflect the loaded state.

These algorithms ensure a fast, user-friendly seat management experience with reliable undo operations and data persistence. TreeSet, HashMap, and Stack provide the necessary performance and structure for handling real-time bookings and cancellations efficiently.

