RAILWAY TICKET BOOKING

OUTLINE

Introduction

Objective

Description Of Modules

Future Enhancements

Conclusion

INTRODUCTION

- Railway reservation systems are widely used real-time applications that require accuracy, efficiency, and proper handling of shared resources such as seat availability. Manual ticket booking systems are error-prone and inefficient, making automation essential.
- This project presents a Java-based Railway Ticket Booking System with a Graphical User Interface (GUI) developed using Swing and AWT components. The system supports multiple coaches, dynamic seat allocation, passenger-based ticket generation, and cancellation functionality.
- The application is designed to be user-friendly, modular, and scalable, making it suitable for understanding how real-world booking systems function while applying core Java programming concepts such as inheritance, polymorphism, encapsulation, synchronization, collections, and event handling.

OBJECTIVE

The objective of this project is to design and implement a Railway Ticket Booking System using Java Swing that simulates real-world railway reservation operations. The system allows users to:

- Select source and destination stations
- Choose different classes of coaches
- View seat availability
- Book tickets with passenger details
- Calculate ticket fare based on age and coach type
- Cancel booked tickets
- View booked tickets in a printable ticket format

The project aims to demonstrate the effective application of Object-Oriented Programming (OOP) concepts, GUI development using Java Swing, and thread-safe booking mechanisms

DESCRIPTION OF MODULES

MODULES

- **Main Menu Module**
  - ➢ Entry point of the application
  - ➢ Provides navigation to booking, cancellation, and ticket viewing
- **Booking Module**
  - ➢ Coach selection
  - ➢ Station selection
  - ➢ Seat availability display
  - ➢ Seat selection
- **Passenger Module**
  - ➢ Passenger name and age input
  - ➢ Validation of passenger data
- **Ticket Module**
  - ➢ Ticket generation
  - ➢ Ticket storage
  - ➢ Fare calculation
- **Cancellation Module**
  - ➢ Ticket cancellation using seat ID
  - ➢ Seat availability restoration
- **View Tickets Module**
  - ➢ Display of booked tickets
  - ➢ Printable ticket-style layout

BLOCK DIAGRAM

```
User
│
▼
Main Menu
│
├── Book Ticket
│    ├── Select Coach & Stations
│    ├── Select Seats
│    ├── Enter Passenger Details
│    └── Generate Ticket
│
├── Cancel Ticket
│    └── Free Seat & Remove Ticket
│
└── View Tickets
     └── Display Printable
```

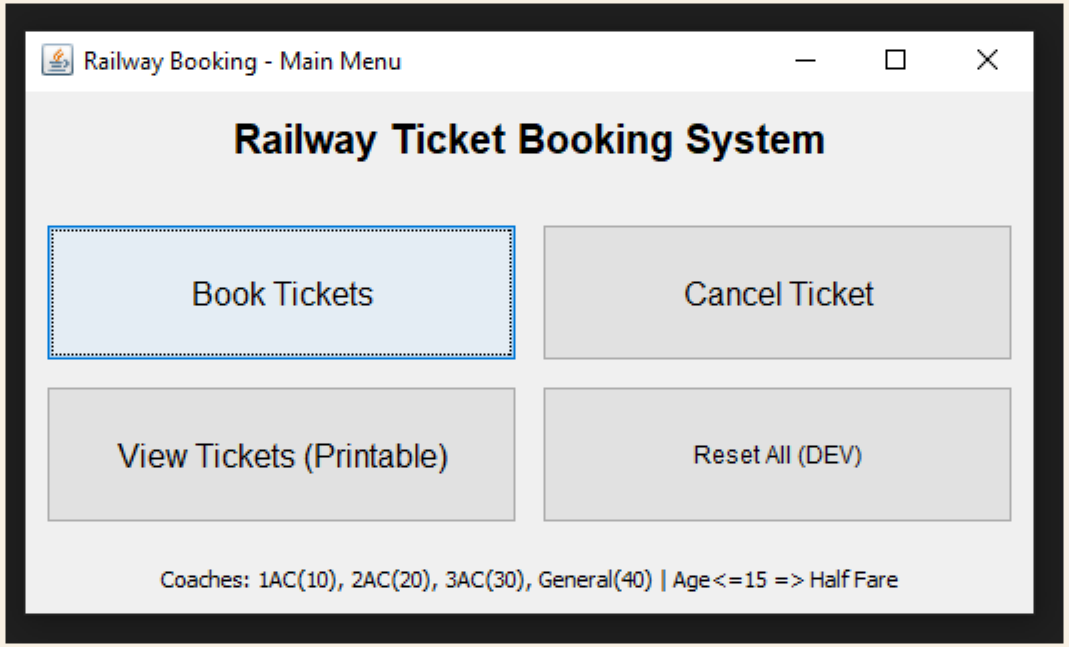
Main Menu Module

Requirement & Functionality

- Acts as the central control of the application
- Displays options for booking, cancellation, viewing tickets, and resetting data

Java Techniques Used

- Swing components (JFrame, JButton, JLabel)
- Event handling using ActionListener
- Layout managers (BorderLayout, GridLayout)



Coaches: 1AC(10), 2AC(20), 3AC(30), General(40) | Age<=15 => Half Fare

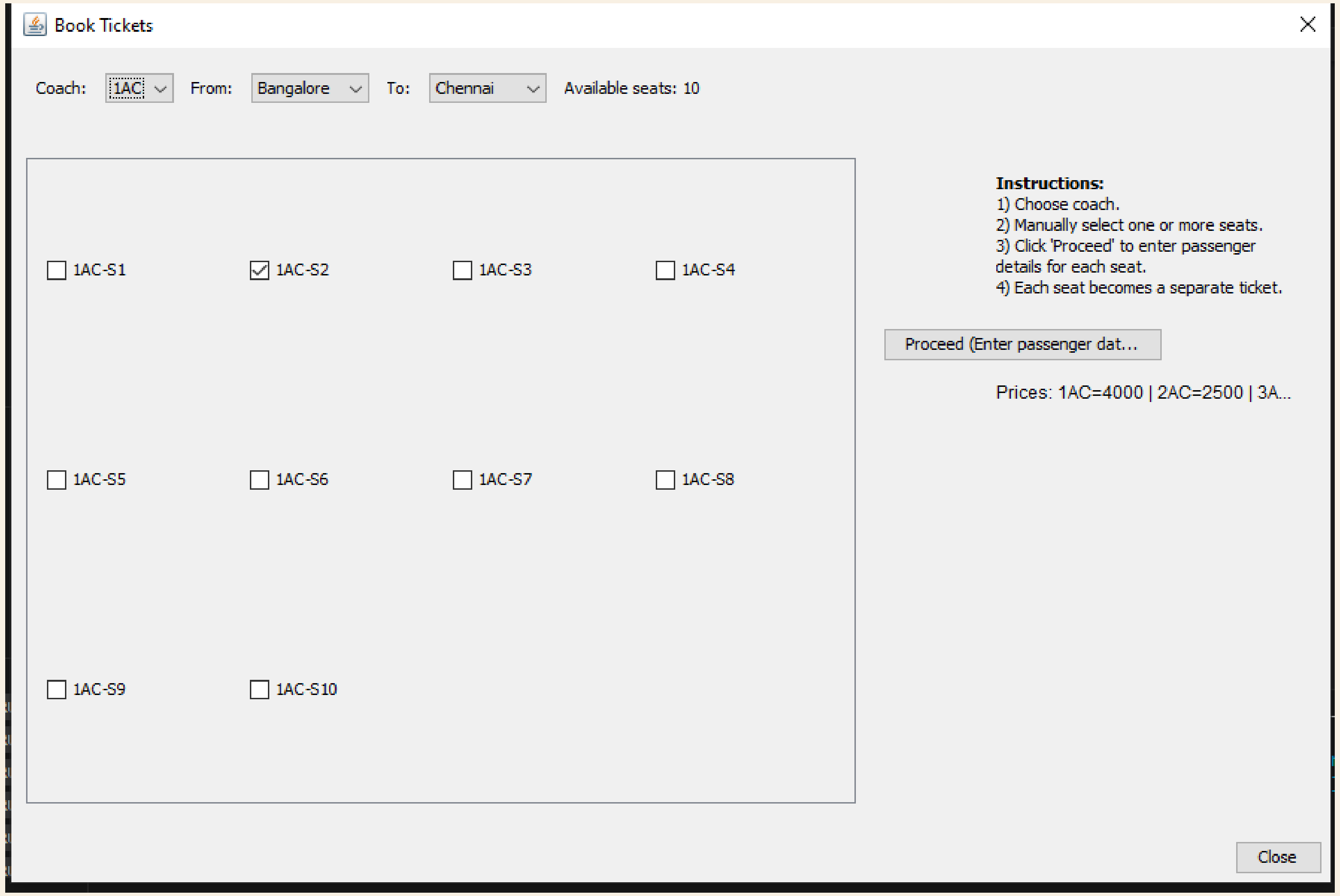
Ticket Booking Module

Requirement & Functionality

- Allows user to select:
    - Coach type (1AC, 2AC, 3AC, GEN)
    - Source and destination stations
- Displays seat availability dynamically
- Supports booking of multiple seats at once

Java Techniques Used

- Swing components (JDialog, JComboBox, JCheckBox)
- Java Collections (Map, List)
- Java Streams (stream(), filter(), collect())
- Validation using conditional checks


Book Tickets

Coach: 1AC From: Bangalore To: Chennai Available seats: 10

☐ 1AC-S1 ☑ 1AC-S2 ☐ 1AC-S3 ☐ 1AC-S4
☐ 1AC-S5 ☐ 1AC-S6 ☐ 1AC-S7 ☐ 1AC-S8
☐ 1AC-S9 ☐ 1AC-S10

Instructions:
1) Choose coach.
2) Manually select one or more seats.
3) Click 'Proceed' to enter passenger details for each seat.
4) Each seat becomes a separate ticket.

Prices: 1AC=4000 | 2AC=2500 | 3A...

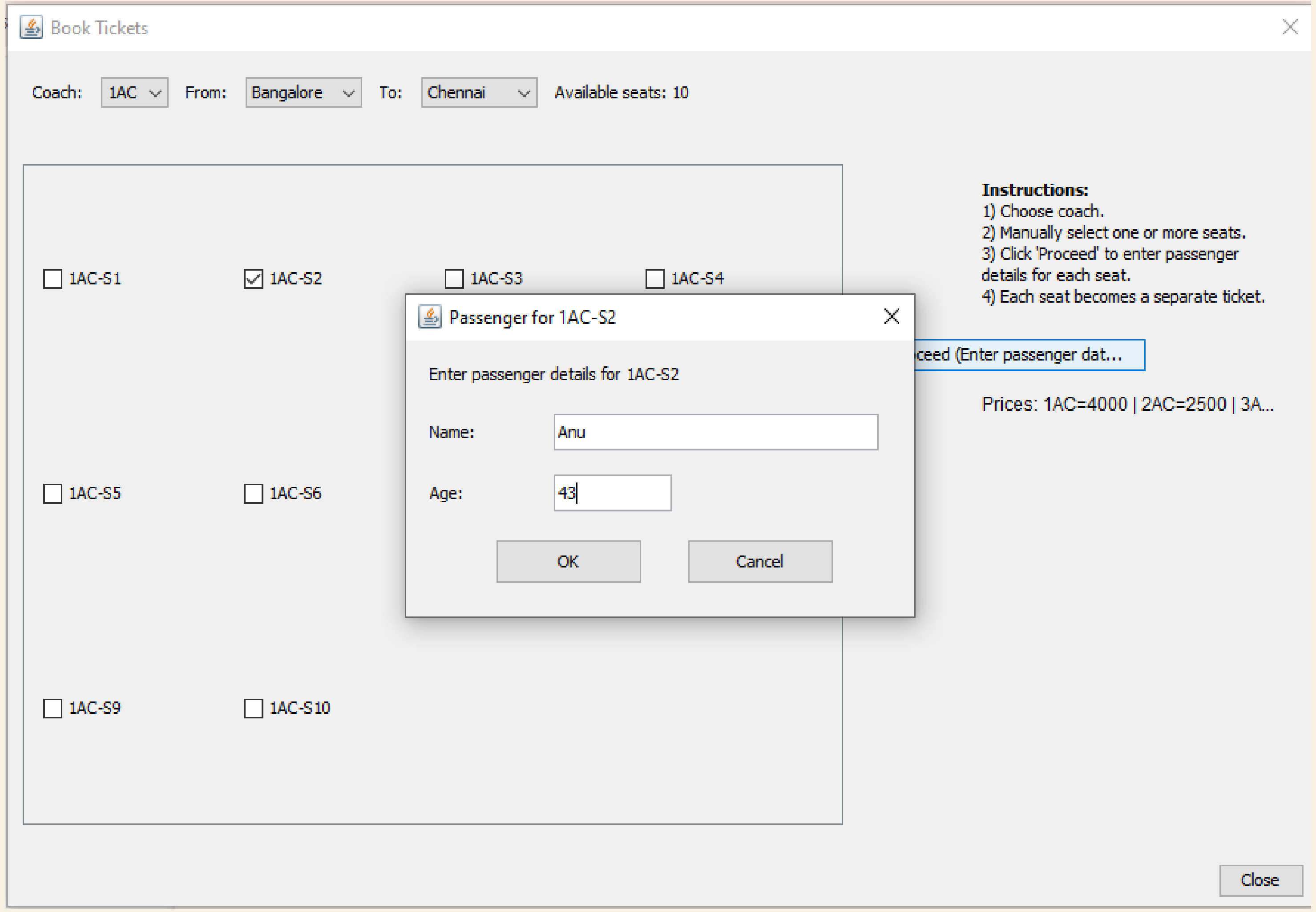
Passenger Details Module

Requirement & Functionality

- Collects passenger name and age
- Performs validation for empty input and invalid age
- Confirms booking per passenger

Java Techniques Used

- Modal dialogs (JDialog)
- Exception handling (NumberFormatException)
- Encapsulation of passenger data
- Input validation logic


Book Tickets

Coach: 1AC From: Bangalore To: Chennai Available seats: 10

Instructions:
1) Choose coach.
2) Manually select one or more seats.
3) Click 'Proceed' to enter passenger details for each seat.
4) Each seat becomes a separate ticket.

Prices: 1AC=4000 | 2AC=2500 | 3A...

Passenger for 1AC-S2

Enter passenger details for 1AC-S2

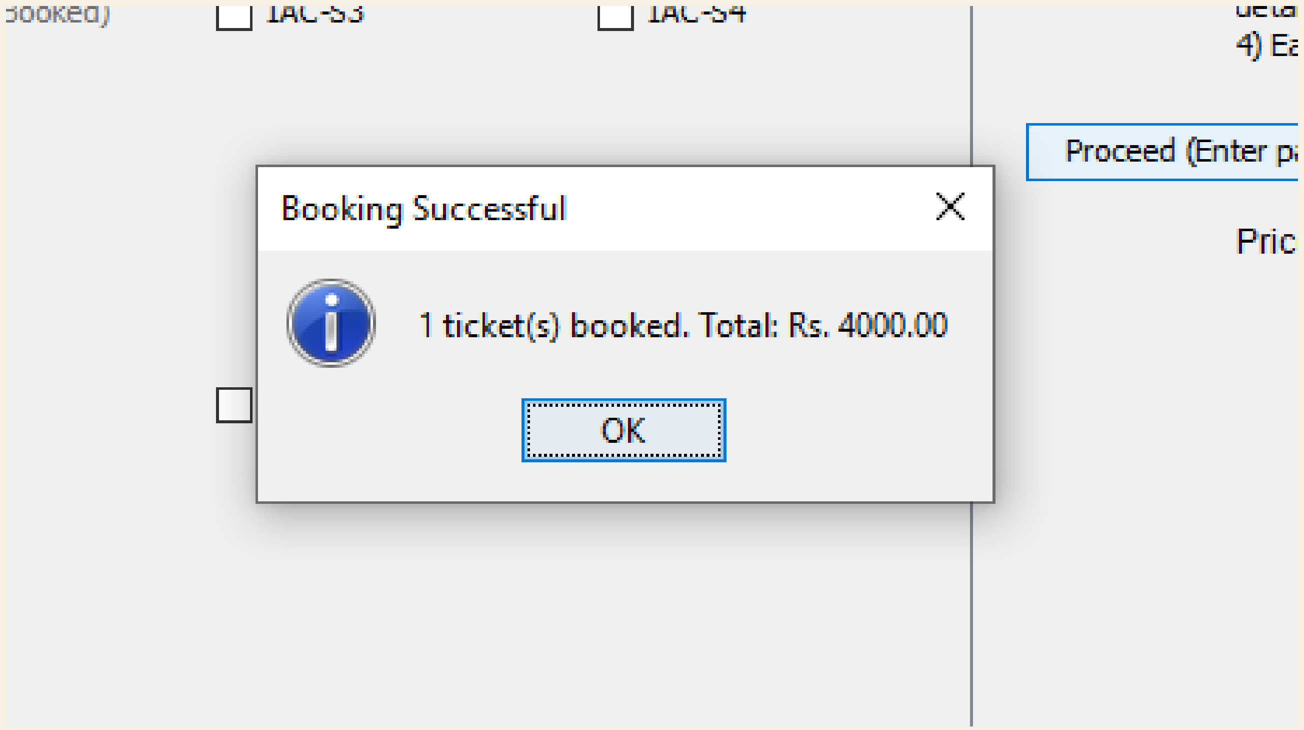
Name: Anu

Age: 43

Ticket Management Module

Requirement & Functionality

- Generates unique ticket IDs
- Stores ticket details
- Calculates fare based on:
    - Coach type
    - Passenger age (half fare for age ≤ 15)



1 ticket(s) booked. Total: Rs. 4000.00

Java Techniques Used

- **Encapsulation** using Ticket and BaseTicket classes
- Java Date & Time (Date, SimpleDateFormat)
- Use of HashMap and ArrayList

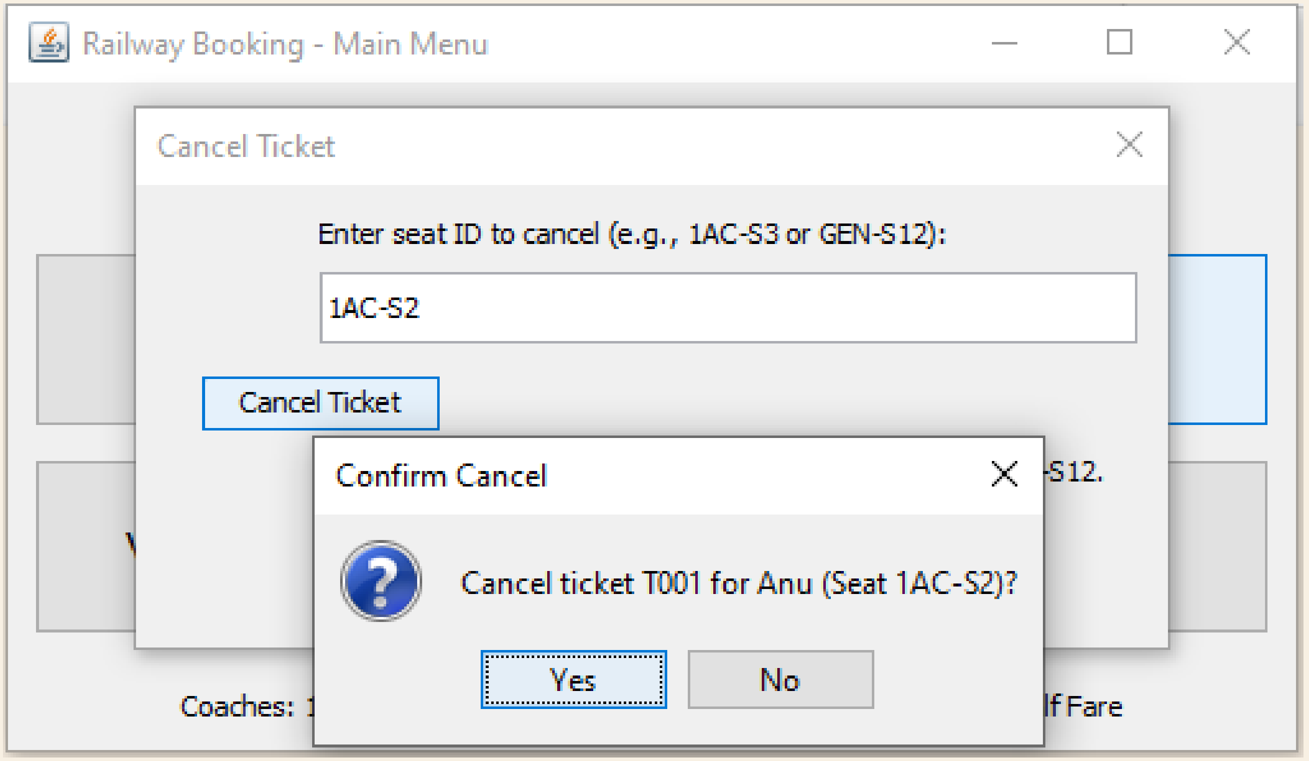
Ticket Cancellation Module

Requirement & Functionality

- Cancels ticket using seat ID
- Frees the seat for future booking
- Removes ticket from system

Java Techniques Used

- Swing dialogs
- Map lookups
- Confirmation dialogs
- Collection modification



Enter seat ID to cancel (e.g., 1AC-S3 or GEN-S12):

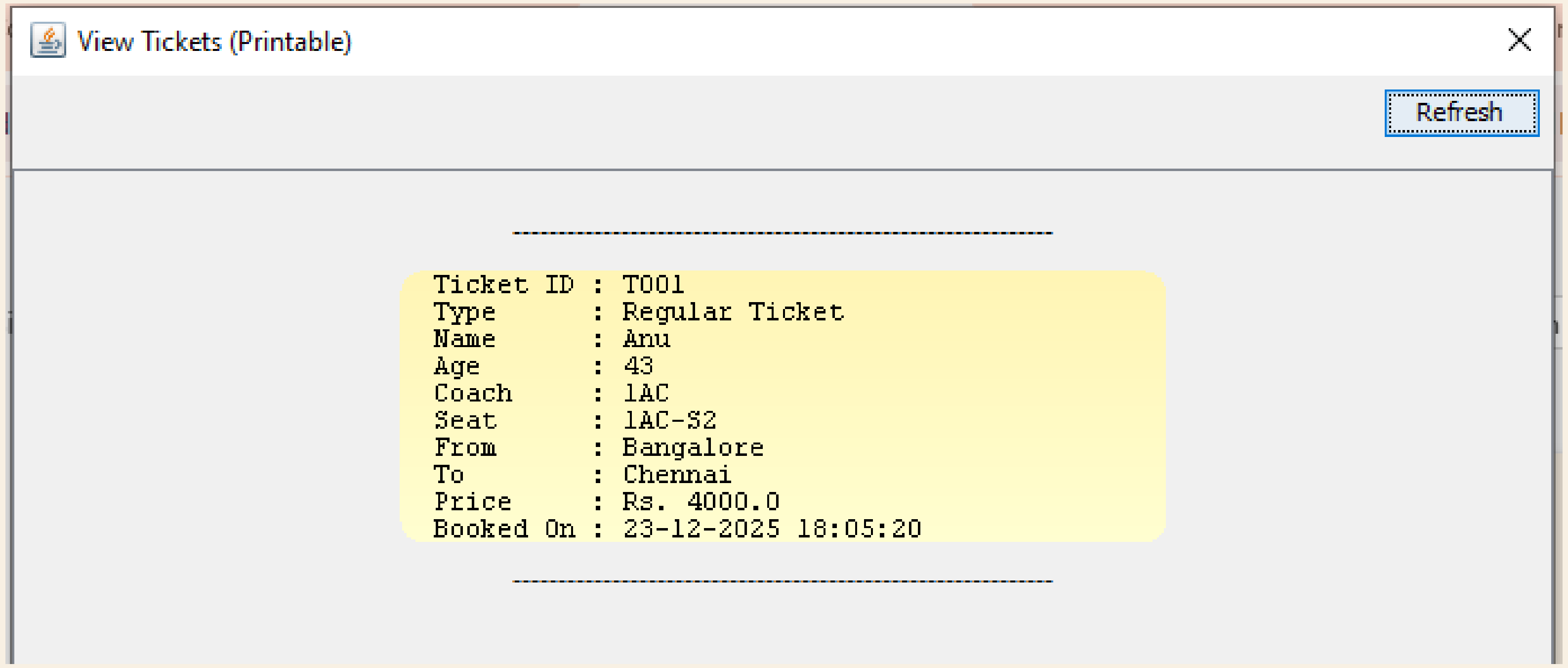
1AC-S2

Cancel ticket T001 for Anu (Seat 1AC-S2)?

Ticket Viewing Module

Requirement & Functionality

- Displays all booked tickets
- Printable ticket-style layout
- Shows all ticket details clearly



```
----------------------------------------------------------------
Ticket ID : T001
Type      : Regular Ticket
Name      : Anu
Age       : 43
Coach     : 1AC
Seat      : 1AC-S2
From      : Bangalore
To        : Chennai
Price     : Rs. 4000.0
Booked On : 23-12-2025 18:05:20
----------------------------------------------------------------
```

Java Techniques Used

- Custom painting using paintComponent()
- Graphics2D, GradientPaint
- Swing layouts (BoxLayout, JScrollPane)

OOP Design

Inheritance

- Ticket class extends abstract class BaseTicket
- abstract class BaseTicket
  class Ticket extends BaseTicket

Polymorphism

- Abstract method getTicketType() implemented in subclass
- Method called dynamically while displaying ticket details

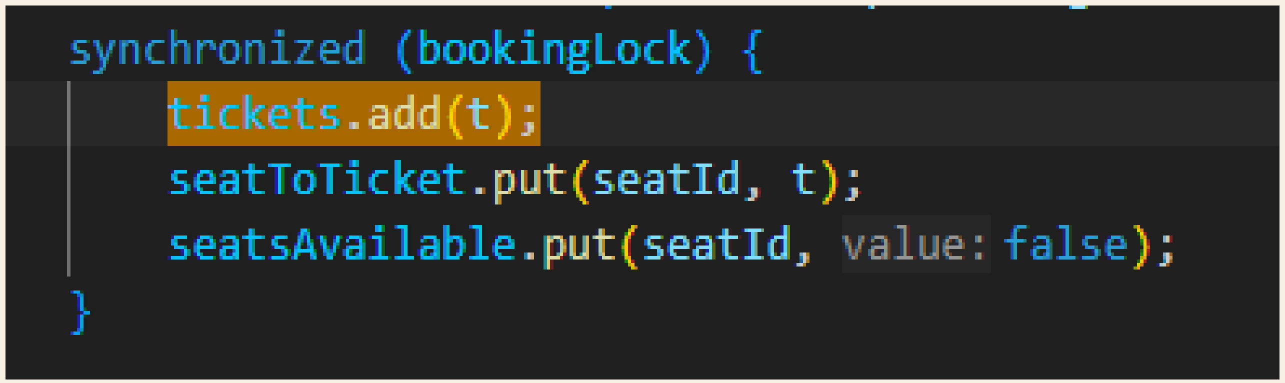
- **Encapsulation**
- Ticket-related data bundled inside classes
- Accessed through controlled methods

```
BaseTicket (abstract)
----------------------
ticketId
name
age
seat
coach
from
to
price
bookedOn
----------------------
getTicketType() (abstract)
          ▲
          |
Ticket
----------------------
getTicketType()
```

Synchronization and Thread Safety

Requirement & Functionality

- Prevents inconsistent seat booking
- Ensures data integrity when multiple operations occur



```
synchronized (bookingLock) {
    tickets.add(t);
    seatToTicket.put(seatId, t);
    seatsAvailable.put(seatId, value: false);
}
```

Java Techniques Used

- Synchronization using:
- synchronized (bookingLock)
- Thread-safe access to shared resources

BOOKING FLOW

User selects Book Ticket

|

▼

Coach Selected

|

▼

Seats Loaded from Map

|

▼

Seat Availability Checked

|

▼

Passenger Dialog Opened

|

▼

Age Validation

|

▼

Price Calculation

|

▼

Ticket Object Created

|

▼

Seat Locked using synchronized block

|

▼

Ticket Stored Successfully

SYNCHRONIZATION FLOW

Multiple Booking Requests
│
▼
Access Shared Data
│
▼
synchronized(bookingLock)
│
▼
Update seatsAvailable
│
▼
Update seatToTicket
│
▼
Update tickets list
│
▼
Release Lock

FUTURE ENHANCEMENTS

Although the current system meets the core requirements, several enhancements can be implemented to improve functionality and scalability.

- Database Integration
  The system can be enhanced by integrating a database such as MySQL or PostgreSQL to store ticket and passenger data permanently instead of using in-memory collections.
- User Authentication
  Login and registration modules can be added to allow multiple users to book and manage their tickets securely.
- Multiple Ticket Types
  Additional ticket types such as Tatkal, Senior Citizen, or Student tickets can be introduced to demonstrate stronger polymorphism.
- Online Payment Gateway
  A payment module can be added to support digital transactions such as UPI, debit, or credit card payments.

CONCLUSION

- Successfully developed a **GUI-based Railway Ticket Booking System** using Java
- Implemented **Object-Oriented Programming concepts** such as:
  - Encapsulation
  - Inheritance
  - Polymorphism
  - Abstraction
- Designed an interactive interface using **Java Swing & AWT**
- Ensured **thread safety** using synchronization to avoid seat conflicts
- Improved understanding of **real-world application design**, event handling, and collections

This project strengthened practical knowledge of Java and OOP concepts through a real-time application.

THANK YOU

Pooja Basavaraj Kuruvinakoppa(1BM24CS203)

Poorvi T(1BM24CS204)

Pradnya Bhosale (1BM24CS207)

Pragathi M (1BM24CS208)