# OOP Java Project: Railway Ticket Booking System

| | |
|---|---|
| **Problem Statement:** | **You are designing and implementing a system to enable users to book train tickets through a command prompt. In other words, you are working for IRCTC to implement the train reservation system.** |
| **Book Tickets** | This train has 5 coaches, S1, S2, B1, A1 and H1. In S1, S2 and B1, there are 72 seats/berths in each coach. In A1, there are 48 seats and in H1, there are 24 seats.<br>S - Sleeper (SL)<br>B - 3 Tier AC (3A)<br>A - 2 Tier AC (2A)<br>H - 1st class AC (1A) |
| **Input:** | Enter Starting Point,<br>Enter Destination,<br>Enter class: S - Sleeper (SL)/B - 3 Tier AC (3A)/A - 2 Tier AC (2A)/H - 1st class AC (1A)<br>No. of passengers:_____ |
| **Output:** | You will be given a list of trains as follows:<br>Total Trains=2<br>1. 17726 Rajkot Mumbai S1-72 S2-72 B1-72 A1-48 H1-24<br>2. 17728 Rajkot Mumbai S1-15 S2-20 S3-50 B1-36 B2-48<br>here,<br>1st word: train no. (E.g. 17726,17728-predefined)<br>2nd word : Source Station<br>3rd word : Destination Station<br>4th word: indicates the list of coaches and available seats within each coach. For example, 17226 S1-72 S2-72 B1-72 A1-48 H1-24 |
| **Input:** | Enter Train number to book Ticket:______<br>*now, system will calculate ticket price based on train selected, coach selected, distance in KM between two stations and no. of passengers) SL - 1 INR per KM per passenger3A - 2 INR per KM per passenger2A - 3 INR per KM per passenger1A - 4 INR per KM per passenger*<br>*E.g. 750 km x 1 INR x 6 passengers = 4500/- Rs.*<br>*Here, distance between Rajkot and Mumbai is 750 KMCoach select is SL & Total passenger=06* |
| **Output:** | Total Fare= 4500 |
| **Input:** | Booking Confirm: Yes/No |
| **Output:** | Ticket Booked<br>PNR:100000001<br>Seat Allcated: 17226 Rajkot Mumbai S1:1,2,3,4,5,6<br>Note: 100000001 represents PNR, which is a unique ticket number. It must be a 9 digit integer number. First ticket issued by the system will have PNR as 100000001, second ticket will have PNR as 100000002 and so on. |

| **Assumptions:** | |
|---|---|
| **1** | Number of trains will be between 1 and 10 |
| **2** | Number of coaches in sleeper category/class will be between 1 and 18 |
| **3** | Number of coaches in 3A (3 tier AC) category/class will be between 0 and 3 |
| **4** | Number of coaches in 2A (2 tier AC) category/class will be between 0 and 3 |
| **5** | Number of coaches in 1A (First Class AC) category/class will be between 0 and 1 |
| **6** | Number of seats in each coach will be between 1 and 72 |
| **7** | Each train runs 7 days a week |
| **8** | Each train completes entire journey on the same day, from first station to last station |
| **9** | Book ticket only if the ticket/seat is confirmed for all passengers, e.g. if there are only 2 seats available in sleeper class (across all coaches), and if user submits a new request with total passengers as 4, then display message “No Seats Available” |
| **10** | It is okay, if all tickets/seats are not available in the same coach, e.g. if there are 2 seats available in S1 and 3 seats are available in S2, and if user submits a new request with total passengers as 4, the ticket will be booked successfully, i.e. 2 seats in coach S1 and another 2 tickets in coach S2. |
| **11** | Based on availability, always book tickets from lowest coach number to highest. Similarly, within each coach, book tickets from lowest seat number to highest seat number |

# Weekly Evaluation Schedule

| Sr. | Week | Description | Deadline | Reference Unit(OOPJ) |
|---|---|---|---|---|
| 1 | **Week1** | **Thorough study of existing system**: IRCTC website Prepare a document (pdf): Aim of study, about the system, enlist features of existing system, system requirements, system drawback can be included. | 20/01/2025 | IRCTC website |
| 2 | **Week2** | Identify classes and draw UML Class Diagram.<br>**Encapsulation**: Basic Class Implementation with attributes, Identification of methods.<br>Array Implementation | 27/01/2025 | Unit-1<br>Unit-2 |
| 3 | **Week3** | **Constructor**: To implement parameterized constructor. **Polymorphism**: To identify and implement Method Overloading. **Inheritance**: To identify Inheritance relationship among classes.<br>Study and implement concept of Date and Time class in java | 03/02/2025 | Unit-2 |
| 4 | **Week4** | Implement **Inheritance** to create parent child relationship between different coach types (General Coach, AC Coach, Sleeper Coach) | 10/02/2025 | Unit-3 |
| 5 | **Week5** | Implement booking logic with Dynamic Method Dispatch | 17/02/2025 | Unit-3 |
| 6 | **Week6** | Implement Tariff logic using **abstraction** (Implement Abstract class, abstract method, Interface among class) | 24/02/2025 | Unit-3 |
| 7 | **Week7** | Write a code to display Current Train seat status report | 03/03/2025 | Unit-4 |
| 8 | **Week8** | Apply Validations on user input and if invalid handle with an **Exception**. | 10/03/2025 | Unit-4 |
| 9 | **Week9** | Create custom Exception to handle if ticket is not available. | 17/03/2025 | Unit-4 |
| 10 | **Week10** | Use **Multithreading** to maintain internal clock. | 24/03/2025 | Unit-4 |
| 11 | **Week11** | Convert project to handle dynamic data using file IO, e.g. get trains data from file, save booked ticket to file | 31/03/2025 | Unit-5 |
| 12 | **Week12** | Generate Report as Text File containing current status of all trains (booked/remaining tickets) | 07/04/2025 | Unit-5 |
| 13 | **Week13** | Testing by Other Student: your project will be given to other student to test thoroughly, you need to test other student's project | 14/04/2025 | TEST |

**Note:**

1. Code should include all the **Object-Oriented concepts**:
   a. **Class and Object**: Use array of Objects, constructor
   b. **Inheritance**: IS-A relationship between class.[Dynamic Method Dispatch-if possible]
   c. **Encapsulation**: Class bound with access specifier(public/private/protected/default)
   d. **Polymorphism**: Method overloading and Method Overriding
   e. **Abstraction**: Implement Abstract class with abstract method and also implement interface in java.
   f. **Keyword**: use static, this, final
   g. **Exception Handling**: Handle runtime error through custom exception.

**This is the bare minimum solution which is expected for the problem.**

*You can add some more features once you are done with these:*

*Example:*

- Convert project to use collection framework instead of array.

***Best of luck***