**\*Railway Reservation System using core Java\***

**Code:**

import java.util.\*;

class Train {

private String tn;

private int avseats = 50;

private int count = 1;

public String[] pn;

public int[] pa;

ArrayList<String> h = new ArrayList<String>();

public void setTn(String tn)

{

this.tn = tn;

}

public String getTn()

{

return tn;

}

Train(String tn)

{

this.tn = tn;

}

public void bookTicket(Map<String, Train> trainMap)

{

Scanner sc = new Scanner(System.in);

System.out.println("Enter the Train Name:");

String a = sc.nextLine();

if (!trainMap.containsKey(a))

{

trainMap.put(a, new Train(a));

}

Train t = trainMap.get(a);

System.out.println("Enter the number of tickets you want to book:");

int n = sc.nextInt();

String m[] = new String[n];

int k[] = new int[n];

sc.nextLine();

System.out.println("Passenger Details:");

for (int i = 0; i < n; i++)

{

System.out.println("Passenger " + (i + 1) + " Details");

m[i] = sc.nextLine();

if (t.h.contains(m[i]))

{

System.out.println("Name already exists! Please check again.");

i = i - 1;

continue;

} else

{

k[i] = sc.nextInt();

t.h.add(m[i]);

sc.nextLine();

}

}

for (int i = 0; i < 100; i++)

{

System.out.print("\*");

}

System.out.println("\n");

System.out.println("\t\tYour Ticket Details\n");

for (int i = 0; i < 100; i++)

{

System.out.print("\*");

}

System.out.println("\n");

System.out.println("\t\tTrain Name:" + a + "\n");

for (int i = 0; i < 100; i++) {

System.out.print("-");

}

System.out.println("\n");

System.out.println("\t\tPassenger Details");

for (int i = 0; i < 100; i++) {

System.out.print("-");

}

System.out.println("\n");

System.out.println("Passenger Name\t Passenger age \t Seat Number");

for (int i = 0; i < n; i++) {

System.out.println(m[i] + "\t\t " + k[i] + "\t\t " + t.count);

t.count++;

}

for (int i = 0; i < 100; i++) {

System.out.print("-");

}

System.out.println("\n");

System.out.println("Ticket(s) booked Successfully");

t.avseats = t.avseats - n;

System.out.println("Seats remaining:" + t.avseats);

for (int i = 0; i < 100; i++) {

System.out.print("\*");

}

System.out.println("\n");

System.out.println("Thank you for using the Ticket Reservation System");

}

}

public class Main

{

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

String tn = " ";

String a = " ";

Map<String, Train> trainMap = new HashMap<String, Train>();

for (int i = 0; i < 75; i++)

{

System.out.print("\*");

}

System.out.print("\n");

System.out.println(" Welcome! to Railway Ticket Reservation System ");

for (int i = 0; i < 75; i++)

{

System.out.print("\*");

}

System.out.println("\n");

while (true)

{

System.out.println("Please Choose an option");

System.out.println("1. Book a Ticket\n2. Exit");

int choice = sc.nextInt();

switch (choice) {

case 1:

Train t = new Train(tn);

t.bookTicket(trainMap);

break;

case 2:

System.out.println("Thank you for using the Ticket Reservation System. Goodbye!");

break;

}

if (choice == 2)

break;

}

}

}

**Explaination:**

# Code Report for Railway Ticket Reservation System

This Java program represents a simple Railway Ticket Reservation System. Users can book tickets for different trains, and the system keeps track of available seats and passenger details.

## Code Structure

The code is divided into two classes: `Train` and `Main`.

### Train Class

The `Train` class represents a train and contains the following attributes and methods:

- Attributes:

- `tn` (String): Represents the train name.

- `avseats` (int): Represents the number of available seats on the train (initially set to 50).

- `count` (int): Represents the seat number for booking (initially set to 1).

- `pn` (String array): Stores passenger names.

- `pa` (int array): Stores passenger ages.

- `h` (ArrayList of Strings): Stores passenger names to check for duplicates.

- Methods:

- `setTn(String tn)`: Sets the train name.

- `getTn()`: Returns the train name.

- `Train(String tn)`: Constructor to initialize a train with a given name.

- `bookTicket(Map<String, Train> trainMap)`: Allows users to book tickets for a train.

### Main Class

The `Main` class contains the `main` method and serves as the entry point of the program. It interacts with users and manages the booking process.

## Program Execution Flow

1. The program starts with a welcome message and a menu with two options: "Book a Ticket" and "Exit."

2. Users can choose to book a ticket or exit the program.

3. If the "Book a Ticket" option is selected:

- The program prompts the user to enter the train name.

- If the train does not exist in the `trainMap`, it creates a new train object.

- Users are prompted to enter the number of tickets they want to book and passenger details for each ticket.

- Duplicate passenger names are checked to ensure uniqueness.

- The program prints a ticket summary with passenger details and updates the available seat count.

4. If the "Exit" option is selected, the program displays a goodbye message and terminates.

## Code Quality and Potential Improvements

1. Code Organization: The code is relatively well-organized with clear separation between the `Train` class and the `Main` class.

2. User Input Handling: The code handles user input using a `Scanner` object, but it could benefit from input validation to handle invalid input gracefully.

3. Magic Numbers: The use of constants for values like the initial seat count (50) and the number of asterisks in the output could improve code readability.

4. Exception Handling: It's a good practice to handle exceptions that may arise from user input or file operations.

5. Code Comments: Adding comments to explain the purpose of methods and significant sections of code would make the code more understandable.

6. Code Refactoring: The code in the `bookTicket` method could be refactored to improve readability and maintainability, particularly by breaking it down into smaller methods.

7. Data Storage: Currently, passenger data is stored in parallel arrays (`pn` and `pa`) and an `ArrayList` (`h`). Consider using a `Passenger` class to represent passengers and store them in a collection for better data organization.

8. Reservation System Features: Enhance the reservation system by adding features like ticket cancellation, seat selection, and storing bookings persistently.

9. Documentation: Provide detailed documentation or user instructions to make it easier for users to understand and use the system.

Overall, the code provides a basic implementation of a railway ticket reservation system but can be improved and expanded to handle more advanced scenarios and user interactions.