# Java Track

# Assessment 1

(Topics included: Core Java: OOPS, collections, multithreading, IO)

Contents

[Java Track 1](#_Toc486319760)

[Assessment 1 1](#_Toc486319761)

[Write a java application called ‘Railway Reservation System’. 2](#_Toc486319762)

[Specification 2](#_Toc486319763)

[How application works: 3](#_Toc486319764)

[Menu 3](#_Toc486319765)

[Book the ticket 3](#_Toc486319766)

[Cancel ticket: 6](#_Toc486319767)

[Get Ticket Details: 7](#_Toc486319768)

[Reservation Chart: 8](#_Toc486319769)

[Display unbooked tickets: 8](#_Toc486319770)

[Exit: 9](#_Toc486319771)

[Multithreading: 9](#_Toc486319772)

[Marking Scheme: 9](#_Toc486319773)

**Note:**

Please read the whole document carefully to understand the specification before starting coding. In case of any doubts, please ask.

Minimum 15 minutes should be given to just understand the requirement.

# Write a java application called ‘Railway Reservation System’.

## Specification

Application should provide the following features to the user. Detail description of each feature along with the screenshot is mentioned below.

**Overview of features**

1) [Booking of tickets](#_Book_the_ticket). New transaction id would be generated for each transaction. A user can book multiple tickets with 1 transaction id.

2) Ability to [Cancel booked tickets](#_Cancel_ticket:) using the transaction id.

3) [Display booked ticket,](#_Get_Ticket_Details:) passing in the transaction id.

3) [Display the reservation chart](#_Reservation_Chart:)

4) Display tickets available, i.e. [tickets left/](#_Display_unbooked_tickets:) un-booked.

5) [Exit](#_Exit:)

Note: The text in yellow background displays the screenshots for how your app should look like.

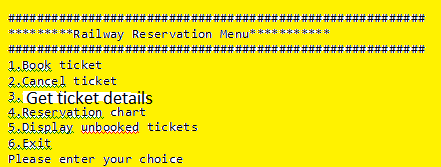
[Marking scheme](#_Marking_Scheme:) is mentioned at the end of this document. Kindly go through it to know the marks distribution

# How application works:

Once the user runs this application, he should see the menu as below and prompt for the user input to select proper option as per his request.

## Menu

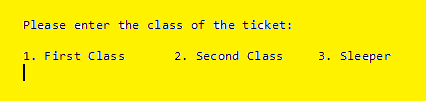
should be like this:



**Option1:**

## Book the ticket

If the user chooses option 1 i.e. to **Book the ticket,** it should again ask the user for making a choice in the various compartments available i.e. whether he wants to book the seat in the sleeper class or the second class or the first Class.



If the user chooses the class, it should prompt for the number of tickets the user wants to buy.



After this, it should take in the details of the user:

1. Name of the passenger
2. Age
3. Phone Number

Since the user entered 2 as the number of tickets, the menu should be displayed twice for taking in the details of the user. After successfully taking in these details for number of tickets booked, it should show the ticket to be booked successfully. Ticket would **only be booked if details of both the users was filled in properly**. Code should apply **validations** on name, age and phone number.

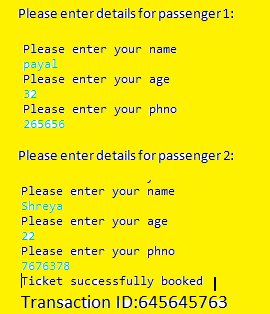
Name: should be alphabetical characters only

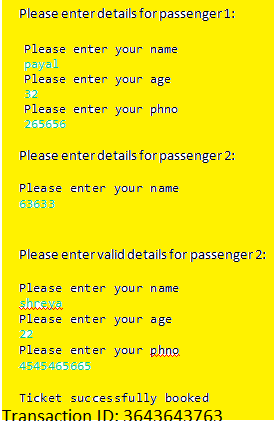
Age: should only be numeric and greater than 0 and less than 150

Phone number: should not be more than 13 digits.

In case any of the user entered wrong details, whole transaction should get cancelled. No ticket should be booked for any of the user.

**Unique transaction id should get generated for each transaction.** It could be string or long. This would be used for cancelling the ticket or display of booked tickets





After taking inputs from the user, should display the amount the user needs to pay. Must have the amount saved for all three classes in the data-structure and total amount should be equal to the number of passengers \* cost per ticket.

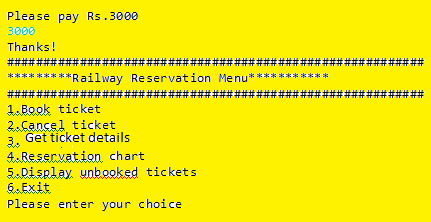
Cost for First Class: 2000

Cost for Second Class: 1500

Cost for sleeper class: 1000

After the user types in the amount on the console, it should display Thanks!

And the main menu should be displayed again.



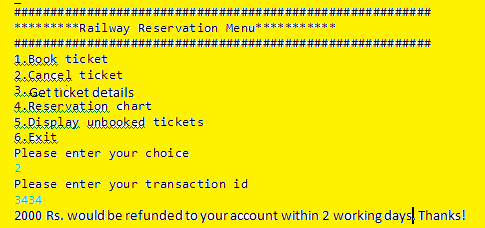
Similarly user should be able to book a ticket for sleeper and second class

Display the main menu on enter

**Option2:**

## Cancel ticket:

User should be asked for the transaction ID which got generated during the booking of the ticket to cancel the ticket.



Corresponding to the transaction id, it should display the amount which was paid to book the ticket. This amount should be refunded back on cancellation. When the user presses enter, again the Main menu should get displayed. If invalid transaction id, should display message: “Invalid ID” and should go back to main menu. Even this record should be saved in the db for the user may want to see the cancelled ticket later corresponding to this transaction id.

Display the main menu on enter

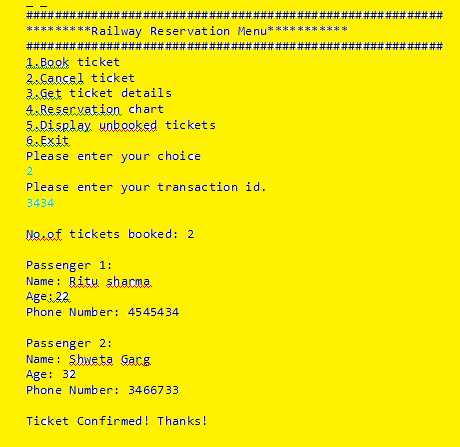
**Option3:**

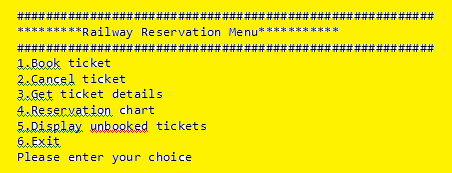
## Get Ticket Details:

Should ask for transaction id and should display all the details regarding that particular id.

1. Number of tickets booked
2. Name, age and phone number for all the tickets booked in that transaction
3. Amount paid.
4. Confirmed/ Cancelled

If ticket was cancelled, should display the details along with Ticket cancelled. Else should display ticket confirmed with all details. By default all tickets booked would be automatically confirmed unless cancelled.

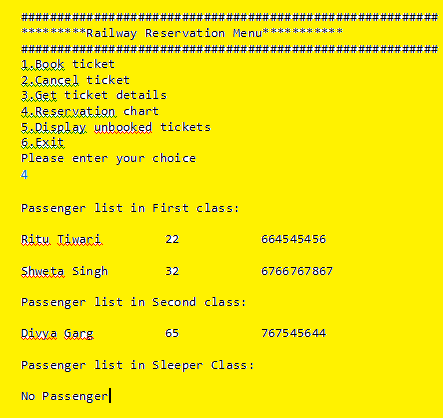




**Option 4:**

## Reservation Chart:

Should display all tickets booked user name, age, phone number corresponding to ticket classes.



If no passenger: should display “No Passenger” under that particular class.

Display the main menu on enter

**Option 5:**

## Display unbooked tickets:

Considering total seats available in First class be 5, Second class be 10 and sleeper be 50, should display count of tickets available for booking

Show back the main menu on enter

**Option 6:**

## Exit:

Choosing this option should terminate the application.

## Multithreading:

Either attempt 1 or both 2 and 3:

1. If only 1 ticket left and 2 users try to book tickets in parallel, 1 must be able to book whose transaction is completed earlier.

OR

1. Cancel of tickets should be allowed to happen in parallel and
2. Correct tickets left should be displayed to both the users.

## Marking Scheme:

5 marks for choosing correct collection classes

5 marks for naming conventions

15 marks for correct functioning of all options and able to navigate from one option to other without breaking the application

10 marks for correct exception handling

5 marks for correct [multithreading](#_Multithreading:) application

END OF DOCUMENT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*