**Railway Ticket Booking System**

**BY**

**Sathish Kumar G**

**Introduction**

The Railway Ticket Booking System is a simple console-based application developed in Java. It simulates a ticket booking experience for passengers traveling on the Chennai Beach Train. This program provides an interactive menu for users to select their destination, preferred time of travel, and mode of payment. Additionally, it verifies payment status and generates tickets accordingly.

**Features**

1. **Destination Selection:**
   * Options for three destinations:
     + Chennai Park
     + Chennai Fort
     + Chennai Beach
2. **Time Slot Selection:**
   * Multiple time slots are provided for each destination.
3. **Payment Options:**
   * Cash payment
   * UPI payment
4. **Payment Verification:**
   * Confirms whether the payment is received before issuing the ticket.
5. **User Feedback:**
   * Displays appropriate messages for successful and failed payments.
6. **Graceful Exit:**
   * Handles invalid input scenarios gracefully.

**Code Breakdown**

**Package Declaration**

package RandomProjects;

Defines the package name as RandomProjects.

**Imports**

import java.util.Scanner;

The Scanner class is imported to facilitate user input.

**Main Class**

public class RailwayTicket {

The main class is named Railway Ticket and contains the program logic.

**Entry Point**

public static void main (String [] args) {

The main method serves as the entry point of the program.

**Welcome Message**

System.out.println("WELCOME TO CHENNAI BEACH TRAIN");

Displays a welcome message to the user.

**Destination Selection**

System.out.println("Select the Destination:");

System.out.println("1) Chennai Park 2) Chennai Fort 3) Chennai Beach");

Prompts the user to choose their travel destination.

**Nested Selection and Payment Flow**

* **Destination Options:** Based on the user's input (i), the program branches to specific destinations:
  + Chennai Park
  + Chennai Fort
  + Chennai Beach
* **Time Slot Options:** For each destination, different time slots are provided. The user selects their desired slot (n).
* **Payment Options:**
  + Users can choose between Cash or UPI payment methods (m).
  + Payment statuses are verified:
    - 1) Received
    - 2) Not Received
* **Feedback:** Depending on the payment status, appropriate messages are displayed:
  + "Amount Received"
  + "Amount Not Received, TRY AGAIN!!!"
  + "Payment Cancel"

**Exit Message**

System.out.println("x---------------Have a Nice Day----------------x");

Displays a goodbye message.

**Program Logic**

The system employs nested if-else statements to handle the multi-level menu structure. Each branch manages specific user interactions:

1. **Destination and Time Slot Handling:**
2. if (i == 1) {
3. // Logic for Chennai Park
4. } else if (i == 2) {
5. // Logic for Chennai Fort
6. } else if (i == 3) {
7. // Logic for Chennai Beach
8. } else {
9. System.out.println("No Trains are Available");
10. }
11. **Payment Verification:**
12. if (m == 1) {
13. System.out.println("Cash");
14. // Further nested logic
15. } else if (m == 2) {
16. System.out.println("UPI");
17. // Further nested logic
18. } else {
19. System.out.println("Access Denied");
20. }

**Sample Output**

**Example 1: Successful Booking**

WELCOME TO CHENNAI BEACH TRAIN

Select the Destination:

1)Chennai Park 2) Chennai Fort 3) Chennai Beach

1

Chennai Park

1) 10.00AM 2) 6.00PM

1

10.00AM

1) Cash or 2) UPI

2

UPI

Pay your Cash using Mobile Banking of Rs.20 using This UPI id: sathishupi@513

1)Received 2) Not Received

1

Amount Received

Have Your Ticket

x---------------Have a Nice Day----------------x

**Example 2: Failed Payment**

WELCOME TO CHENNAI BEACH TRAIN

Select the Destination:

1)Chennai Park 2) Chennai Fort 3) Chennai Beach

2

Chennai Fort

1) 4.00PM 2) 7.00PM

2

7.00PM

1) Cash or 2) UPI

1

Cash

Pay your cash Payment of Rs.200

1)Received 2) Not Received

2

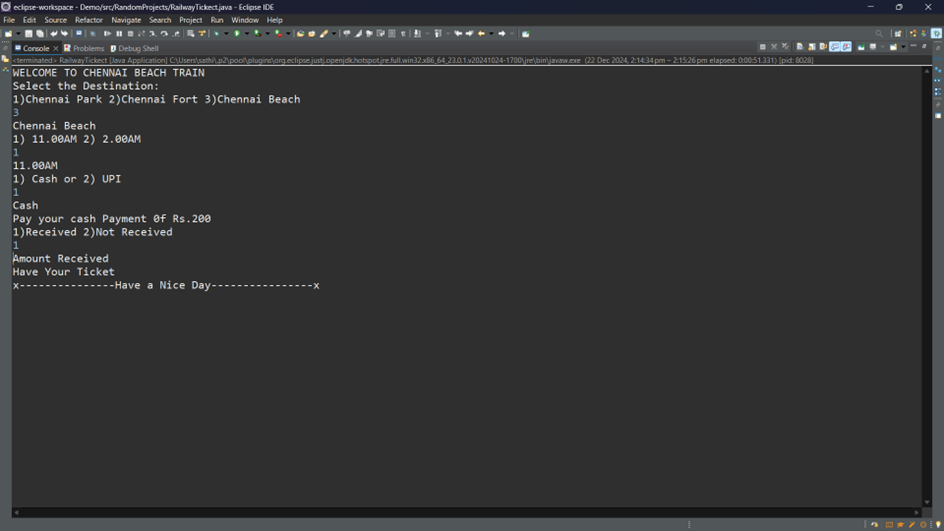
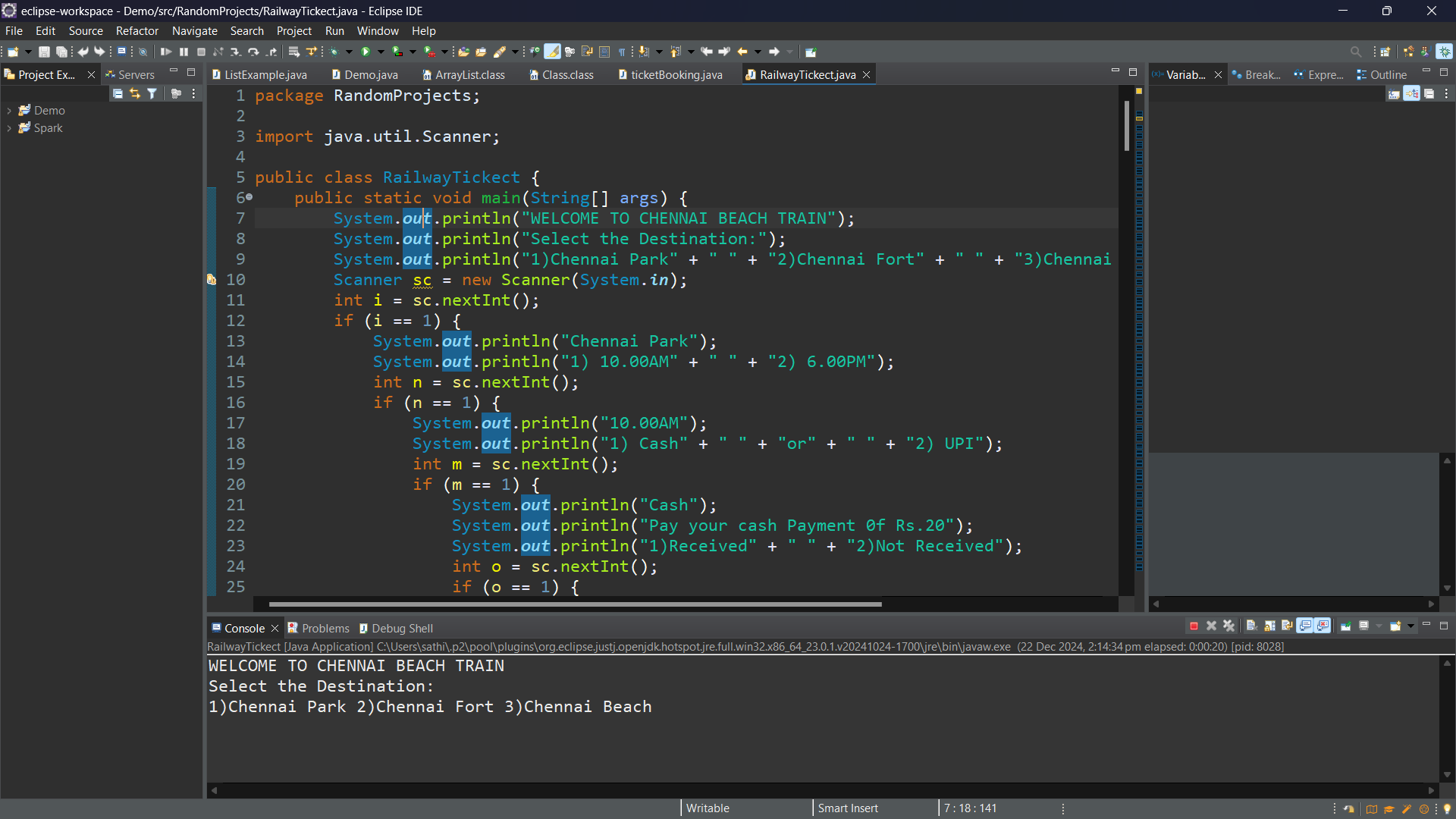
Amount Not Received, TRY AGAIN!!!

x---------------Have a Nice Day----------------x

**Possible Enhancements**

1. **Error Handling:**
   * Use exception handling to manage invalid inputs.
2. **Scalability:**
   * Replace nested if-else with a more scalable approach like switch statements or dynamic data structures.
3. **GUI Integration:**
   * Develop a graphical user interface for better user experience.
4. **Dynamic Pricing:**
   * Implement a pricing algorithm based on factors like peak hours.
5. **Database Integration:**
   * Store user and transaction details for future reference.

**Screenshots**



**Conclusion**The Railway Ticket Booking System demonstrates a practical implementation of conditional statements and user input handling in Java. While it is a basic project, it serves as a foundation for developing more complex systems with enhanced functionality and user experience.