

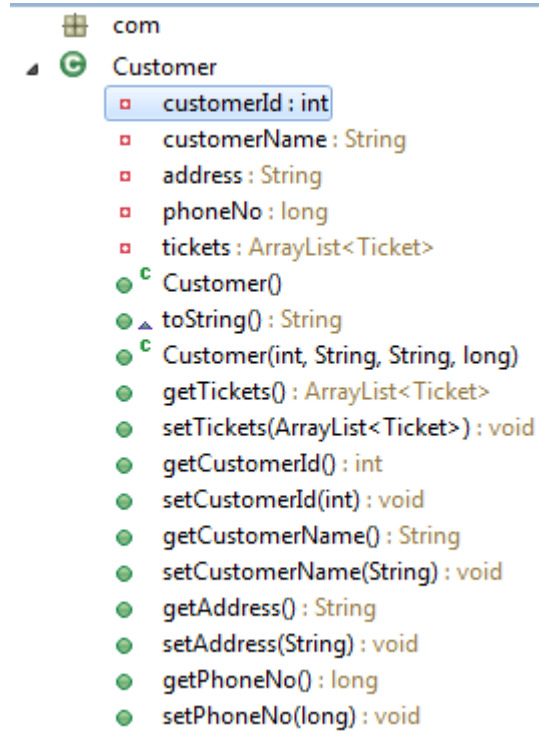
PS2_Diagnostics

- Go through the problem statement clearly.
- Time limit is 120 minutes.
- Make sure that project is created in eclipse only. The project name in eclipse must contain your employee ID and it should be of the form "PS2_yourEmployeeID_FirstName".
- Create all your java files in package "com" within src folder of eclipse project.
- Make sure that exact class outline is followed as you did in previous assignments.
- You need to zip the eclipse project folder and upload the same once completed. The project folder will be available in your workspace folder.
- It is mandatory to upload eclipse project and not only java files for your code to be assessed.
- Make sure that there is no compilation error in your code before submission. Even if there is minor error, entire solution could be rejected. Make sure to check the spelling of the given class outline.
- You may refer previous assignments for any reference.

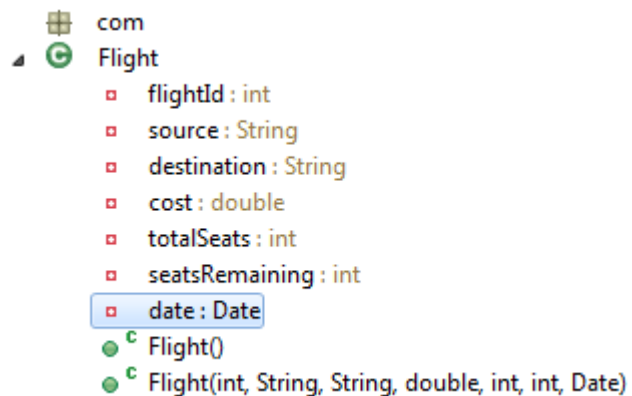
Problem Statement:

A new aviation company, **ABN Airlines**, wants to automate their Air Ticket Reservation system. You are required to make the program for them. Develop a Java project which serves this system following below guidelines. Class outline is shared after the guidelines. Ensure the same is met 100%, else your solution would be not considered.

1. Create project AirReservationDemo in eclipse. Create package **com** within **src** folder. All classes should be created inside this package.
2. Create a class Customer as following

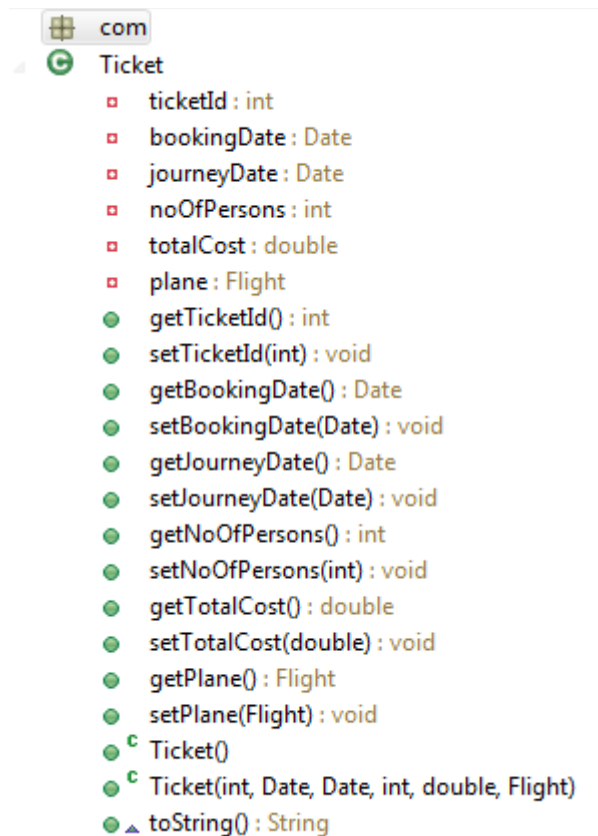


3. Create a class Flight as following :



- getDate() : Date
- setDate(Date) : void
- getFlightId() : int
- setFlightId(int) : void
- getSource() : String
- setSource(String) : void
- getDestination() : String
- setDestination(String) : void
- getCost() : double
- setCost(double) : void
- getTotalSeats() : int
- setTotalSeats(int) : void
- getSeatsRemaining() : int
- setSeatsRemaining(int) : void
- toString() : String

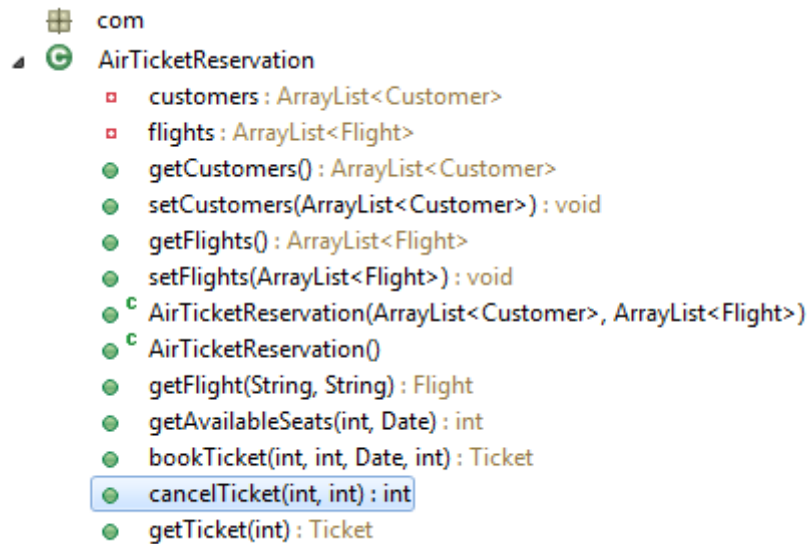
4. Create a class Ticket as following :



The screenshot shows a package named 'com' containing a class named 'Ticket'. The class has the following attributes and methods:

- Attributes (indicated by red squares):
 - ticketId : int
 - bookingDate : Date
 - journeyDate : Date
 - noOfPersons : int
 - totalCost : double
 - plane : Flight
- Methods (indicated by green circles):
 - getTicketId() : int
 - setTicketId(int) : void
 - getBookingDate() : Date
 - setBookingDate(Date) : void
 - getJourneyDate() : Date
 - setJourneyDate(Date) : void
 - getNoOfPersons() : int
 - setNoOfPersons(int) : void
 - getTotalCost() : double
 - setTotalCost(double) : void
 - getPlane() : Flight
 - setPlane(Flight) : void
 - Ticket() (constructor)
 - Ticket(int, Date, Date, int, double, Flight) (constructor)
 - toString() : String

5. Create a class AirTicketReservation as follows:



- **Methods to be implemented in AirTicketReservation:**

- 1) `getFlight(String source, String destination):`

This method will take two Strings – 'Source' and 'Destination' as parameters and then search for a flight between originating from the source and going to the destination given. Assume that for each pair of source and destination there is at most only one flight. This method will return the object of Flight.

If there is no flight matching the required criteria then the method should throw a **FlightDoesNotExist** exception and a message "Flight with destination : destination and Source : source does not exist" should be displayed.

- 2) `getAvailableSeats(int flightId, Date date)`

This method will return the number of seats available on a particular date in a flight and take FlightId and Date as the arguments. It will return an integer.

If no seats are available then it will throw a **FlightAlreadyFull** Exception and a message "The flight with ID : flightId is already full." should be displayed.

- 3) `bookTicket(int flightId, int custId, Date dateOfJourney, int noOfPersons):`

This method will book a ticket in a flight. It will take flightId of the flight, customerid of the customer booking the ticket, date of journey and number of persons travelling as arguments and return an object of ticket which has been booked. Whenever a ticket is booked, the number of available seats of the corresponding flight will be decreased by the number of the persons travelling on the booked ticket.

If the flight with the given Flight Id does not exist then it will throw a **FlightNotFound** Exception and a message "Flight with id : flightId could not be found" should be displayed.

If there is no seats remaining then it will throw a **FlightAlreadyFull** Exception with

the above mentioned message.

If the available number of seats is less than the number of persons travelling then show a proper message.

4) `cancelTicket(int ticketId, int custId) :`

This method will cancel a ticket. It will take ticket id and customer id(who has booked the ticket) as arguments. It will then search the tickets of the customer with the given customer id and if the ticket with the given ticket id is found then it will be cancelled. Whenever a ticket is cancelled, the number of available seats of the corresponding flight will be increased by the number of the persons travelling on the cancelled ticket.

If the ticket with the given ticket id is not found then this method will return a `TicketNotFoundException` and display the message `"The ticket with id: ticketId could not be found"`.

5) `getTicket(int ticketId) :`

This method will return a ticket. It will take ticket id as an argument and return the corresponding ticket object with the given ticket id.

If the ticket is not found then the method will throw a `TicketNotFoundException`.