**More Arrays – Planes part 2**

Cartesian Planes (Incorporated) was so pleased with your work, they’ve hired you to expand that plane program to track passengers and seat bookings. They need to know who is on a plane, where they’re sitting, what they’re eating, and when the plane is full.

Specifications:

*Classes…*

* One class for Passenger
  + Private variables Name and FoodPreference
    - Both strings
    - FoodPreference options are “chicken”, “pasta”, and “special”
  + Default values in the constructor set name to “empty” and food to “none”
  + Getter’s and Setter’s for both variables
* One class for Plane (edit the one you have to include the following)
  + New variables…
    - HasMeal - boolean
    - NumOnPlane – number of passengers, an int
    - MaxSeats – total number of seats, an int
    - Bookings – a 2D array that indicates which passengers are in which seats
      * As passengers “buy” their bookings, they will be placed here
      * Their position in the array will be their row and seat number
      * The constructor will set the size of the array and fill the array with Passenger’s default constructor
  + Getter’s for all variables
  + Setter’s for Seats and HasMeal
  + Method to count how many of each FoodPreference are on the plane
  + Method to create an alphabetical list of who’s on the plane and in which seat
  + Method to find a passenger on a plane and return which seat
  + Method to tell user how many are on plane
  + Method to tell user if the plane is full
  + Method to tell user if a seat has already been booked
  + Method to tell user which seats are available (empty)

*Main program…*

* Read “planes.txt” to create the planes (store them in an array)
  + Same as last time
* Read “bookings.csv” to fill the planes…
  + A csv file (comma separated values) is a text file for spreadsheets (so like txt is to Word) – like the name suggests, the items in each row are separated by a comma
  + Note: The seats and rows each start at 1; arrays don’t.
  + If a requested seat is already occupied or doesn’t exist on that plane, allow the user to enter a different seat based on available seats.
  + If the plane is full…
    - Search the planes array to find one with the same destination and seats available
    - If no planes are available, tell the user
    - If a plane is available, list the available seats and allow the user to choose
  + If the plane does not have a meal included, set FoodPreference to “snack” regardless of what the passenger requested.
* Display a menu for the user…
  + Add Passenger?
    - Have user enter plane and passenger info
  + List Planes?
    - Show all planes, day each is travelling as a String, and # of seats taken. Note if a plane is full.
  + List Passengers?
    - Alphabetically show all passengers on a plane
  + Food Count?
    - Tells the user the number of each kind of meal needed for a specific plane.
  + Find Passenger?
    - Tells the user where a passenger is sitting or “not here” if not on this plane.
  + End Program