CS603 Programming Assignment 5

Due by **end of day** on 11/21

1. Getting Started

In this assignment, you will be adding code to predefined classes for maintaining information on passengers seated on a plane.

The following files have been posted to Blackboard and should be downloaded to the same folder on your computer:

(1) **Passenger.java**: Represents Passenger objects by first, middle (optional), and last names, and whether or not they require a special meal.

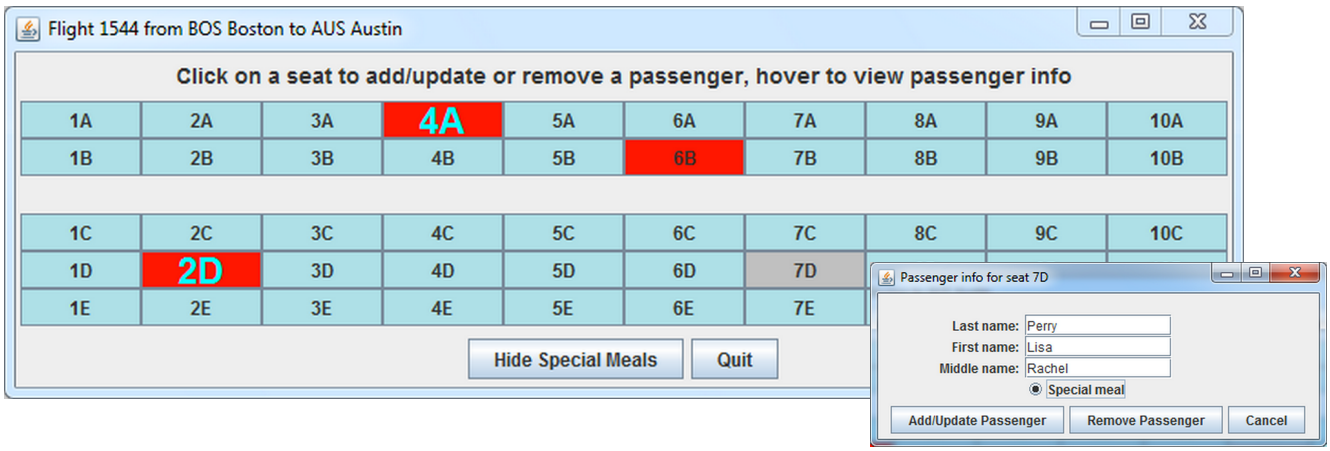
(2) **Seat.java**: Represents Seat objects on a flight by row number and seat ID. Each seat is associated with at most one Passenger object.

(3) **Flight.java**: Represents Flight objects by the flight number, departure and arrival cities, maximum number of seats that can be added to the flight, the number of seats that have actually been added, and an array of those seats.

(4) **FlightGUI.java**: Interface for displaying seats on a flight, adding/updating/removing passengers to those seats, and highlighting those seats in which passengers require special meals.

You will be adding code to the **Passenger, Seat,** and **Flight** classes. **DO NOT make any other changes** to the instance variables or methods that have already been defined. Be sure to **add your name** to the start of these files and make any other appropriate changes to the comments that have been provided. You **will not** be making any changes to the **FlightGUI** class. Once you have completed the other classes correctly, the GUI interface provided by this class should function properly.

Be sure to read over the code provided in the first three classes prior to beginning the assignment. When your code has been completed, the working interface should look like the screenshots shown below:



2. Programming Assignment (27 points total)

The Passenger class (15 points)

Add the following methods after the numbered comments describing them in the code. Be sure your method names exactly match those specified. Only the first method is static; all others are instance methods.

1. **formatName(String)**: **static** method that is passed a string and returns a string containing the same characters but with the first letter in uppercase and all subsequent letters in lowercase. For Example, if passed "toNY" this method should return "Tony." If an empty string is passed to the method, it should be returned unchanged.
2. **setFirstName(String)**, **setMiddleName(String)**, **setLastName(String)**: mutator (set) methods that use the static formatName() method to format the string passed as an argument and then sets the instance variable for which the method is named to that value. No value is returned by any of these methods.
3. **getFirstName()**, **getMiddleName()**, **getLastName()**, **getSpecialMeal()**: accessor (get) methods that return the value of the corresponding instance variable.
4. **four argument (4-arg) constructor**: The header for this constructor has been provided for you. Fill in the code within it to assign each of the arguments to the corresponding instance variables as follows:
   - The name-related parameter values should be assigned using the set methods described above.
   - Assign the value of the fourth parameter directly to the corresponding instance variable.
5. **three argument (3-arg) constructor**: The header for the constructor has been provided for you. This constructor is used for creating Passenger objects without middle names. Invoke the **4-arg constructor for full credit**, setting the middle name to the **empty** string (NOT null).
6. **toString()**: Add this instance method after the comment describing it. The string returned by this method should provide a description of the invoking Passenger object as the last name followed by a comma and a space, the first name followed by a space, the middle name followed by a space if there is a middle name, and ** special meal ** if the passenger gets a special meal. In the examples below, <sp> refers to the **one and only one** space required between each element:

   Xu,<sp> Han<sp>**<sp>special meal<sp>**
   Parker,<sp>Harriet<sp>Rose<sp>**<sp>special meal<sp>**
   Johnson,<sp>Francis<sp>Thomas

Fill in the **main method** provided in this class to use your constructors for creating instances of the Passenger class. Print the objects you have created (your print statements will automatically invoke the toString() method). Since your constructors use your set methods, and your set methods use the formatName() method, this will test all of the code you have added so far except for your get methods. Print the values returned by those methods as well.

Testing with **FlightGUI**: The errors in this class should disappear after you have provided the required get methods. After verifying your code is working correctly, run the FlightGUI class. Click on a seat, add passenger information, and click the Add/Update Passenger button. If you click on the seat again, you will be able to make changes/remove that passenger from the seat.

The Seat class (6 points)

1. **hasSpecialMeal()**: Add this instance method after the comment describing it. This method has no arguments and returns a boolean value of true if a seat has a passenger and that passenger has a special meal. Otherwise, a value of false should be returned.
2. **toString()**: Add this instance method after the comment describing it. The string returned by this method should provide a description of the Seat object as the row number concatenated with the seat ID ("2A" for example). If there is a passenger associated with the seat, then the description of that passenger should be added to the seat description.

   Examples:

   1F
   2A Parker, Harriet Rose ** special meal **

The **main method** includes testing code for creating and printing a Seat object, adding a passenger to that seat, and printing again. Add additional code to cover all test cases – i.e., use your hasSpecialMeal() method on a seat with/without a passenger and with a passenger with/without a special meal.

Testing with **FlightGUI**: If you hover the cursor over a seat containing a passenger, you should now see a description of that passenger.

The Flight class (5 points)

This class has six instance variables, including an **array of type Seat** for storing all the seats on the flight, a **numSeats** variable for storing the maximum number of seats that can be added to the flight (used in the constructor for initializing the size of the Seat array), and a **numAddedSeats** variable that stores the number of seats that **have actually been added to the flight** (used in the addSeat() method).

1. **getSpecialSeats()**: The header for this instance method has been provided for you. Fill in the code within this method so that it returns an array of type **int** containing the **index values** of all Seat objects in the Seat array that have a passenger who is getting a special meal. For full credit, the size of the array that is returned must be **exactly** equal to the number of seats on the plane in which passengers are getting special meals. For example, if three seats have passengers with special meals, then the array must be of size three.

Testing code has been added to the **main method** for adding seats to a flight and adding passengers to those seats. Add code that uses your getSpecialSeats() method for returning an array of seat indices. The contents of that array should be {0,2} given the code that has already been included for you. Be sure to write additional test cases.

Testing with **FlightGUI**: after the above code is working correctly, run the FlightGUI class. Clicking on the "Show Special Meals" button should highlight all seats with special meals and switch the button label to "Hide Special Meals" if there are any such seats.

3. Grading

Your programs must compile and run to receive any credit. Points will be allocated as follows:

15 points for correct Passenger class

- 3 points for formatName() method
- 3 points for set methods
- 2 points for get methods

- 4 points for constructors
- 3 points for toString() method

6 points for correct Seat class
- 3 points for hasSpecialMeal() method
- 3 points for toString() method

5 points for getSpecialSeats() method in the Flight class

1 point for following style guidelines

4. Submitting Your Assignment

When you are satisfied that your code meets all requirements, submit **three** files: **Passenger.java**, **Seat.java**, and **Flight.java**, via the submit link from the Blackboard Assignments page for Assignment 5.