**Unit 2D Arrays Flight Lab**

In this lab you will mimic the process of assigning seats on an airplane. Seat assignments are processed by the public member functions of the class Flight. The seating arrangement is represented internally by a matrix of seats in the class Flight. Additional classes Seat, Passenger, as well as the testing program are attached to this lab.

**You will be writing a few methods for the class Flight.** The declaration for the class Flight is as follows:

public class Flight {  
 private Seat[][] mySeats; //a 2D array of Seats on this flight

//a constructor that creates a row x column matrix of Seats   
 public Flight(int rows, int columns) {  
 mySeats = new Seat[rows][columns];  
 int aisle1 = mySeats[0].length/2;  
 int aisle2 = aisle1 + 1;  
 if (mySeats[0].length % 2 == 0) {  
 aisle2 = aisle1 - 1;  
 }  
 for (int i = 0; i < mySeats.length; i++) {  
 for (int j = 0; j < mySeats[i].length; j++) {  
 if (j == 0 || j == mySeats[i].length - 1) {  
 mySeats[i][j] = new Seat(Seat.WINDOW);  
 }  
 else if (j == aisle1 || j == aisle2) {  
 mySeats[i][j] = new Seat(Seat.AISLE);  
 }  
 else {  
 mySeats[i][j] = new Seat(Seat.MIDDLE);  
 }  
 }  
 }  
 }

//method assigns a passenger to the requested seat and returns true // if it was successful, false otherwise.

public boolean setPassenger(int row, int column, String name) {  
 //your code here   
 }

//method prints the matrix of Seats on this Flight **in row order**  
 public void printChart(){  
 //your code here

}

//method returns the number of empty seats whose type is seatType;

//if seatType is “any”, returns the total number of empty seats

public int emptySeatCount(String seatType) {

//your code here

}

// method returns column index of the first(lowest index)

// seat in a block of seatsNeeded adjacent empty seats

// in the specified row;

// If no such block exists, returns -1

public int findBlock(int row, int seatsNeeded)  
 //your code here

}

//If possible, assigns the group.length passengers from group to //adjacent empty seats in a single row and returns true;

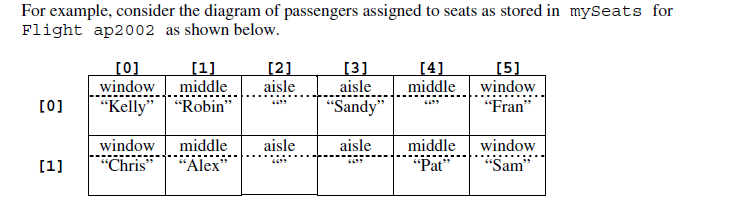
// otherwise makes no changes and returns false

public boolean assignGroup(Passenger[] group){

//your code here

}

* You will write the Flight method emptySeatCount, that returns the number of empty seats of the specified type seatType. Recall that an empty seats holds a default passenger whose name is "". If seatType is “any”, then every empty seat should be counted in determining the number of empty seats. Otherwise, only seats whose type is the same as seatType are counted in determining the number of empty seats.



The following table shows several examples of calling emptySeatCount for this flight.

Method Call Value Returned

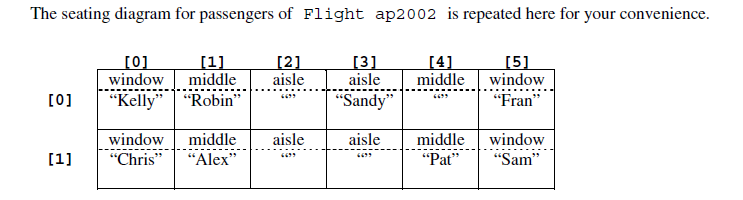
ap2002.emptySeatCount(“aisle”) 3

ap2002.emptySeatCount(“window”) 0

ap2002.emptySeatCount(“middle”) 1

ap2002.emptySeatCount(“any”) 4

* You will write the Flight method findBlock, that searches for a block of seatsNeeded adjacent empty seats in the specified row. If such a block of seats is found, findBlock returns the column index of the first (i.e., the lowest index) seat in the block; otherwise, it returns -1.



The following table shows several examples of calling findBlock for Flight ap2002 as shown.

Method Call Value Returned

ap2002.findBlock(0, 1) 2 or 4

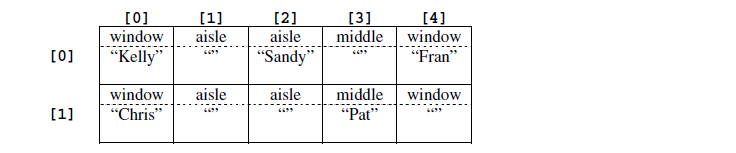
ap2002.findBlock(0, 2) -1

ap2002.findBlock(1, 2) 2

* You will write the Flight class method assignGroup, which is described as follows. The parameter to the Flight class method assignGroup is an array of passengers, group. These passengers require a block of adjacent seats in some row. The method assignGroup searches for group.length adjacent seats in a single row. If such a block of seats is found, the passengers in group will be assigned to those seats, and assignGroup returns true. Otherwise, no passengers are assigned to seats, and assignGroup returns false.

For example, the seats in Flight ap314 are as shown in the first diagram below. If the array adults contains three passengers, the call ap314.assignGroup(adults) makes no changes to ap314 and returns false, because there is no block of three adjacent empty seats in a single row. On the other hand, suppose the array kids contains passengers “Sam” and “Alex”. The call ap314.assignGroup(kids) will assign “Sam” and “Alex” to the seats shown in the second diagram below and returns true.

**Contents of mySeats for ap314 before any call to assignGroup**



**Contents of mySeats for ap314 after call to ap314.assignGroup(kids)**

