Program #3 - Design Notes

Here are some connect-the-dot notes on Program #3 for you.

1. The Reservation File

The reservation file is a CSV file that supports comments. I'll supply you with some test files on the k: drive. The file looks like this:

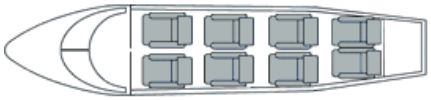
```
# A line that starts with the pound sign is a comment
# The first un-commented line defines the CSV fields
Seat,Passenger
1A,Prof Bill
1B,Ally A
2A,Matt M
2B,open
2C,WilliamW
2D, open
...
```

Details:

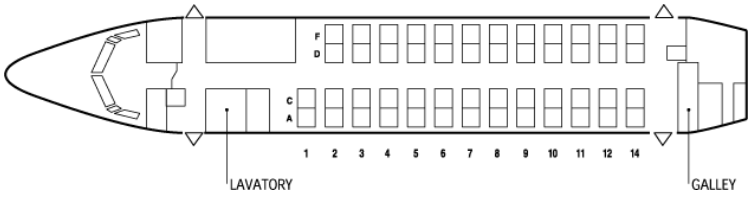
- Our usual CSV rules apply: Lines that start with '#' are comments, and values are separated by commas
- The first non-comment line defines the fields: Seat and Passenger
- If Passenger is "open" then the seat is currently not reserved
- All seats in the plane will be listed in the file. Not all rows have the same number of seats.

I have some test files for you on the k: drive. I used a couple of real aircraft for my seat arrangements.

Tiny example - Cessna 402 has 8 seats in 4 rows. EZ.



Biggie example - Fokker 50 has 50 seats in 13 rows… I won't use their seat letters though. It's A, B, C, D for us. I do, however, like the missing (unlucky) row 13.



2. Simple Sample console session

Here's a (bogus) console session (user typing is in **bold**)

```
***>
      Welcome to p3 Flight Resv System (TM)
      by Prof Bill, CSC 161
***>

Please enter your flight:
Air PB 007
Enter your resv file name:
pb007.txt
Thank you.
There are 50 seats on flight Air PB 007.
21 of these seats are open.

p3>  reserve A1 George W
Error: Seat A1 is already reserved to Thomas J

p3> reserve A2 George W
Your reservation is confirmed: A2,George W

p3> report
Flight Air PB 007 seat assignments are:
      1A,Thomas J
      1B,George W
      2A,open
      2B,open
      2C,Benjamin F
      2D,Samuel A
      3A,open
      … and so on
p3> open
20 seats are open on Flight Air PB007. They are:
      2A,open
      2B,open
      3A,open
      4C,open
      … and so on
p3> save
Num of resv changes made: 1
Resvs saved to file pb007.txt

p3> exit
Thank you. Please drive through.
```

3. Organization

There were some questions about organizing classes on Friday.
Here are some comments:

- ★ Your `Resv` class should be small, holding one reservation: the seat and passenger data.
- ★ The `ResvSystem` class should do a lot of the work in making and breaking reservations. It holds a list of `Resv` objects.
- ★ The `ResvConsole` is the guy who interacts with the user via the console: `System.out.println` and keyboard scanner. He has-a `ResvSystem` and calls system methods to get the work done on reservation changes and reports.
- ★ And `Program3` holds your `main()`. A guideline, not a rule - a BIG main() is usually a problem (because it's code that can't be shared) and a small, simple main() is *usually* on the right track.

I am **not** dictating what you call your classes and exactly what goes where. Your mileage may vary. For example, where do you actually read the file? ResvSystem? ResvConsole? It's up to you.

I **am** dictating, however, that your classes work **if** we were building a gui. For example, in my setup, I envision a `ResvPanel` that is-a `JPanel` and has-a `ResvSystem`. I could call system methods to add and remove reservations. And then call a system method to save my work to a file.

A couple more notes:

- Don't forget to mind the design requirements: param from main(), exception, etc.
- Don't hard-code any file names into your code. Ask the user!

thanks… yow, bill