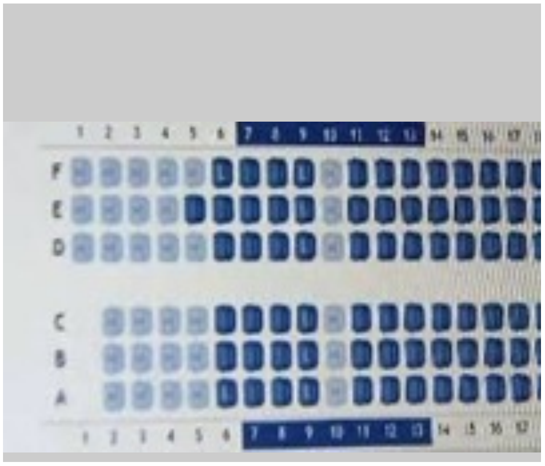
Assignment 4

These days, airlines typically allow their customers to select flight seats if they choose to. Online selection of seating typically requires customer information so that information of the seats selected is stored with the customer's itinerary. When seats are taken, the corresponding seat icons would be grayed out if a graphical interface is used, or a different symbol would be displayed if a console interface is used (see following demos for seating representation). Nonetheless, seating assignment is a relatively independent software task.

```
Flight Seating Info:
1:          [A] [A] [A] [A]
2:          [O] [O] [A] [A]
3:          [A] [A] [A] [A]
4:          [A] [A] [A] [A]
5:          [A] [A] [A] [A]
6:      [A] [A] [A] [A] [A] [A]
7:      [A] [A] [A] [A] [A] [A]
8:      [A] [A] [A] [A] [A] [A]
9:      [A] [A] [A] [A] [A] [A]
10:     [A] [A] [A] [A] [A] [A]
11:     [A] [A] [A] [A] [A] [A]
12:     [A] [A] [A] [A] [A] [A]
13:     [A] [A] [A] [A] [A] [A]
```



This design assignment is about a software application of airline seating assignment. Here is the software specification:

- There are only two seating types: business and economy. Each row has variable number of seats, and there are a variable number of rows divided into two sections: business section at the front of the plane followed by the economy section. There is only one aisle in the middle of the plane.
- A customer can request for a seat based on these preferences: business or economy class, an aisle seat or a window seat. A customer can also request for a seat with no preference. Multiple consecutive seats can also be requested within the limit of the row capacity (for example, if each side of the aisle has only three seats, then the requested number of consecutive seats cannot be more than three). If no seating assignments can be made based on the preference, customer is informed in some way (and no seating assignments are actually made).
- Making a seating assignment means: 1) to change the availability status of the seat (the seat can then be displayed differently), and 2) to associate seating information with the customer's flight reservation data.
- A customer can also request to cancel a seating assignment by giving the seat number.
- There can be different user interfaces possible: 1) a console-based interface, 2) a graphical user interface that supports "clicking on seat-icon button to reserve", and/or 3) a graphical user interface that collects information of seating selection only through textboxes. Thus, it is desirable that the design should support code reuse as much as possible Regardless of a possible user interface.

Here are the assignment requirements:

1. Complete a design with the following:

   a. A diagram that specifies all software elements (include data fields and methods), their hierarchies (if any), and their relations,

   b. A program with skeleton code (showing only the structure of the program) to verify whether the design works (no working program is required, that is, no detailed logic of seating assignment needs to be implemented). For each of the methods you identified, specify pre and post conditions.

Summary:

For this assignment I pulled everything together as a whole that ran in Index. The display class will run either the GUI or console in this case, while the rest of the logic is placed in Seating. I used an interface called Response to "re-create" a part of EMCAScript 6 that I love using. This allows me to create custom error messages and easily handle the responses. The Customer data is always held in the Seat interface and all seats that aren't currently assigned with a Customer will be available. Any seats placed on "hold" or for multiple people will have to be filled at the time of the request & booking.

2. Provide some rationale about the design:

   a. How well can it support any of the user interfaces?

      i. The design is able to support all of the user interfaces

   b. Was data appropriately encapsulated?

      i. the data is encapsulated correctly once grabbed from the database holding the values.

   c. Are software elements responsible-driven?

      i. yes

   d. Are all the elements significant?

      i. very

   e. Are the names (variables, methods, and classes) appropriate?

      i. always