**Venue Booking System**

In this assignment you are to develop a simple booking system to manage the seating for a venue, say for example at a small theatre. In this assignment you will:

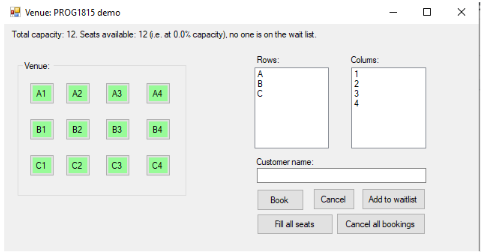
* Practice your GUI development skills by working with basic form controls and build an app that meets the specifications by handling basic events fired by such controls.
* Apply your OOP skills by implementing the main logic of the app inside a class of objects representing Venues.
* Apply your skills utilizing some basic C# collections.

For simplicity, let’s assume the venue has 3 rows with 4 seats in each row. Also, the bookings will just be maintained in memory - we will worry about storing data in files later in the course. The screenshots below will help but to implement this app proceed through the following steps…

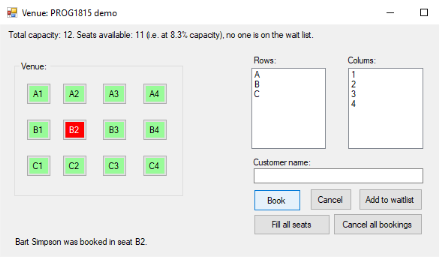
1. Use a grid of labeled buttons to simply represent the seats in the venue. The colour of the button indicates its - e.g. green means the seat is available and red means it is booked.
2. Allow the user to seat (i.e. book) people (by name) in the venue or add them to the waiting list:
   1. The user must enter the name of the person to be seated.
   2. If seats are available, let the user provide a person’s name and select a seat by seat using the row/seat list boxes and then click “book” to book that person’s seat in the venue.
   3. If no seats are available, “Book” and “Add to waiting list” buttons will add the person to the waiting list.
   4. If seats are available, “add to waiting list” button will not do anything, except showing a message “Seats are available”.
   5. Hovering the mouse over a button (representing a seat) will show the booked person’s name, if the seat is booked, otherwise it will just say available.
   6. By clicking the “Status” button, the status of the selected seat will be shown - i.e. the app will indicate if it’s empty or, if occupied, the name of the occupant.
3. Cancel a booking:
   1. Choose the seat row & number to be canceled.
   2. Click “Cancel” to delete the person’s booking.
   3. If the cancellation request is valid (i.e. valid row & number and there is a person already seated) then prompt the user for confirmation via a “confirmation-style” message box.
   4. If the waiting list is empty, display that a seat as available
   5. If the waiting list is not empty, remove the first person from the list and give him or her the newly vacated seat.
4. Show a message, ***using a label at the bottom of the form*** (i.e. do ***not*** invoke message boxes for information messages!), when:
   1. “Book” is pressed when either the person’s name is missing, or the row or seat is not selected.
   2. “Book” is pressed when the chosen seat is already taken
   3. “Add to Waiting List” is pressed when there are seats available.
   4. “Cancel” is pressed without specifying the seat.
   5. After a successful booking.
   6. After successfully canceling a seat.
   7. After successfully adding to the waiting list.
5. Add a “Fill all seats” button. Clicking this button will fill all the vacant seats, up to the maximum of all 12 seats. You must still enter a person’s name but then that name will be used for all the seats filled by this action.
6. Add a “Cancel all bookings” button that cancels all the current bookings.
7. Don’t restrict or constrain the length of the passenger name. For example, I should be able to book a person with the name “a”, as long as the name is not just whitespace.
8. And finally, maintain a summary status at the top indicating the total capacity, the number of seats available (and include with that a percentage of current capacity, and the number of people on the waitlist, if any. Again, see the screenshots below for examples of this.
9. **Remember to incorporate the following for this (and all) program:**

* Program comments before classes, describing what their objects represent.
* Comments before each method describing what it does
* Form control and program variable names should be camel notation, full words
  + UI controls should have a prefix denoting their control type: txt, btn, etc.
  + Variables must not have a datatype prefix or suffix (i.e. no Hungarian notation)
* Methods should be Pascal notation, except for generated event handlers
* Add Implementation comments where (you think) necessary or helpful.

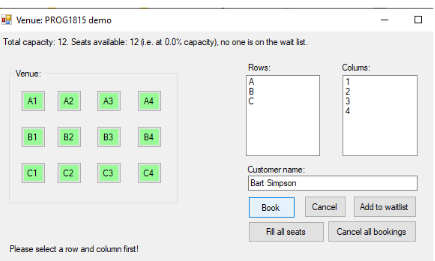
The following screenshots will help guide you. The first one shows the app when it first starts:



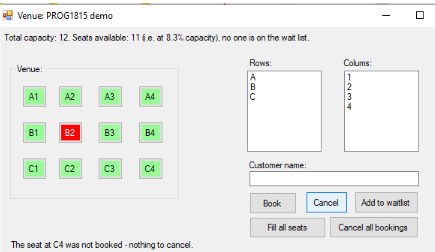
And after a successful booking:



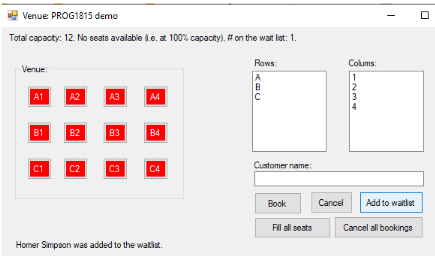
After a failed attempt at booking:



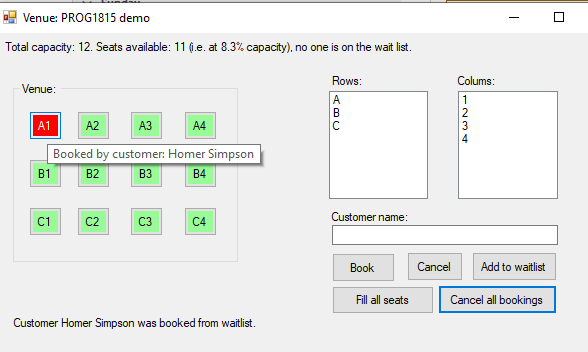
After a failed attempt at canceling a booking:



After one person was added to the waitlist when at full capacity:



And finally, showing the effect of hovering the mouse over a booked seat:



Implementation Notes:

* You *may* use your own imagination for designing the user interface. The screenshots of the Form are given only to clarify the requirements.
* You can also choose whether or not you want to use Exceptions to handle error conditions.
* However, you must define and make use of a class that implements the logic of booking seats in the venue, canceling seats, access seat status, adding to the waitlist, etc.
* You have complete freedom in terms of how to structure the data that represents the seating plan at any given moment in time, but that data structure must be encapsulated within your class.
* To minimize the complexity of the solution, it is ok to keep all the buttons enabled all the time**.**

**What to hand in?**

Bundle and upload your solution folder to the assignment drop-box on eConestoga as a ***single zip*** file. You may submit multiple versions but only the last (i.e.most recent) submission will be looked at and graded.

**How will it be graded?**

Please see the marking sheet that accompanies the assignment. It will be used to calculate your grade so be sure that you are meeting all the criteria included there. Please ensure that you ***submit only your own work***! Remember - all your code submissions will be run through [MOSS](https://theory.stanford.edu/~aiken/moss/), the software plagiarism detection tool.