Problem Statement: Restaurant Seating System

“Joe’s Books and Burgers”, a tasty new uptown restaurant that’s totally not ripping off Books and Brews, needs a system that the host can keep track of available tables in the restaurant, as well as assigning reservations to tables ahead of time to make sure nobody gets double booked. The system will keep a list of open tables (Table name, table size), and a list of reservations (name of party, number of party members, phone number, time of reservation).

The host/hostess can use the system’s GUI to add reservations to the reservation list as parties call in, as well as assign those reservations to tables. While assigning reservations to tables, the host/hostess should first click the “Add New Reservation”, and then, when the interface pops up, type in values for the party name, phone number, party size, and time reserved. After this, they can drag the reservation to a table on the GUI. If the table is open at that time and large enough to accommodate the party, the reservation will be assigned to that table and the process is finished. If the table is not large enough to accommodate the party or already booked at that time, the table assignment is rejected and the user will need to drag the reservation to a different table.

Once a party arrives for their reservation, the host/hostess can remove them from only the Reservation List GUI via a button within said GUI. Additionally, at the end of the day, the host/hostess can click a button and clear the entire system, including the reservations within each table*.

The UI needs to be sleek and simple, and should be usable with minimal effort or training. As this is a local system, internet connection is not necessary. Lastly, the interface should be mostly graphical, with no typing outside of adding new reservations.

For this redo opportunity, billionaire restaurant entrepreneur Joe Rutkowski would like you to draw up three diagrams: One Use Case diagram, one UML Class diagram, and one Sequence diagram for the Use Case of adding a new reservation to the system, and then assigning them a table. ***<u>YOU DO NOT NEED TO WRITE THIS PROGRAM.</u>*** Mr. Rutkowski has unpaid college interns for that. Any and all information you need to create the diagrams will be provided below.

Actors for System:

- Host/Hostess

Use cases for Use Case Diagram:

- Add new Reservation to ReservationGui
- Add Reservation to Table
- Remove Reservation (One, just reservation list)

- Clear Reservations (All, tables and reservation list)

Classes Needed for UML Class Diagram:

- Table
    - Attributes
        - Private String tblName, a 3-character string consisting of 2 numbers and a letter
        - Private int tblSize, an integer with a minimum value of 2 and a maximum value of 10
        - Private HashMap<Reservation> resTimes, a hashmap with the reservation time as a key and the Reservation object as a value
    - Operations
        - Public boolean AddReservation(Reservation r), a method that checks if a reservation will fit this table. Returns true and adds the reservation to resTimes if it can, returns false otherwise.
        - Private Getter and Setter for resTimes
        - Private Getters for other attributes. No setters, as the table names/sizes will not change. These bad boys are bolted into the ground.
- Reservation
    - Attributes
        - Public String resNames, a string consisting of whatever name the caller gave the host
        - Public int resSize, an integer representing the number of people in the reservation.
        - Public String resNum, a string representing the phone number of whoever called.
        - Public String resTime, a string representing the time of the reservation
            - Full disclosure, I thought this whole problem statement up in my head using C#, had this as a DateTime field, but remembered we use java in this class ~5 minutes before I went to publish this. So its a string now for simplicity's sake
    - Operations
        - Constructor called by ReservationListGUI
- ReservationListGUI
    - Attributes
        - Private List <Reservation> resList, a list of reservations
        - Private Button btnResAdd, a button adding a reservation to the list

        - Private Button btnResRem, a button removing a reservation from the list.
    - Operations
        - Private void resAdd(String name, int size, String Phone, String Time), a method that calls the reservation constructor
        - Private void resRem(), a method that removes a reservation from the database when needed.
- TablesMapGUI
    - Attributes
        - Private List <Table> tblList, a list of table objects.
        - Private Button btnClearAll, a button calling the clear all method
    - Operations
        - Private void addResToTbl(Table , Reservation r) calls the corresponding table's addReservation method when a reservation is dragged to it.
        - Private void ClearAll(), clears the system

Sequence for Sequence diagram

- Wow it's almost like I gave you that in the problem statement
    - Pretty neat, huh

*The reason why you can't clear the reservations stored in the table until you clear the whole system at the end of the night is so we have a record and phone number for anyone who's dined-and-dashed. I didn't become a billionaire by letting people eat for free.