|  |  |
| --- | --- |
| **Customer** | |
| **Responsibilities** | **Collaborators** |
| Customer knows its name, no, and party size |  |
| Customer signs up on the waitlist via their phone | WaitlistManager |
| Customer types in STOP to receive messages when customer receives a confirmation message and no longer want to wait | WaitlistManager |

­­­

|  |  |
| --- | --- |
| **WaitlistManager** | |
| **Responsibilities** | **Collaborators** |
| WaitListManager knows about the wait list queue |  |
| Adds a particular customer to the back of the waiting list when a customer signs up on the waitlist via their phone | Customer |
| Deletes a particular customer from the waiting list when customer types in STOP to no longer be on the waitlist | Customer |
| Deletes a particular customer from the waiting list when customer confirms with “CONFIRM” or types in “LEAVE” | TableHandler1 |

|  |  |
| --- | --- |
| **TableManager** | |
| **Responsibilities** | **Collaborators** |
| TableManager knows the size and status of the next available table |  |
| TableManager notifies the TableHandler1 when the next table becomes available | TableHandler1 |

|  |  |
| --- | --- |
| **TableHandler1** | |
| **Responsibilities** | **Collaborators** |
| Handler knows the next handler to handle the request if it cannot process the request |  |
| TableHandler1 receives a message with table available and its size from TableManager when a table becomes available | TableManager |
| Handler notifies the next customer in line that satisfy the party size requirements | Customer |
| If the handler receives a CONFIRM message from the customer when being notified, TableHandler1 notifies WaitListManager to delete | WaitListManager |
| If the handler receives a LEAVE message from the customer when being notified, the handler notifies TableHandler2 to handle the request | TableHandler2 |
| If the handler cannot accommodate the size it sends the request to TableHandler2 to assign | TableHandler2 |

|  |  |
| --- | --- |
| **TableHandler2** | |
| **Responsibilities** | **Collaborators** |
| Handler searches the next customer in line that satisfy the party size requirement | TableHandler1 |
| Handler notifies the next customer in line that satisfy the party size requirements | Customer |
| If the handler receives a CONFIRM message from the customer when being notified, TableHandler1 notifies WaitListManager to delete | WaitListManager |
| If the handler receives a LEAVE message from the customer when being notified, the handler finds the next customer in line that the table request satisfies and sends a message to the waitlist manageger to delete customer from queue | WaitListManager |

I used Observer Pattern and Chain of Command pattern to solve this problem.

The two participating classes in observer pattern are TableManager and Tablehandler1. TableHandler1 (the observer) is subscribing to updates from TableManager (the subscriber) to receive messages when a table is *soon to be ready*. The TableManager passes the table size information to the TableHandler1 so that it can determine whether the first customer in line can accommodate the request.

I am using Chain of Command Pattern because TableHandler1 might not be able to handle the request from the TableManager. TableHandler1 will look up the party size information for the first customer in line. If the table cannot accommodate the party size, the TableHandler1 passes on the request to TableHandler2. TableHandler2 will search and find the next customer in line that best fit the table size.

I chose to draw a communication diagram in Asta. I chose not to include the maintenance of the queue in WaitListManager to avoid clutter (message to delete customer from queue if STOP message sent, etc.).