|  |  |
| --- | --- |
| Restaurant | |
| **Responsibilities** | **Collaborators** |
| * Maintain a queue of waiting parties * Maintain a list of table observers * Check new party in * Check old party out * When a party leave, notify and serve the next capable party in the waitlist if any | * Party * ConcreteObserver * ConcreteObserver, Party * ConcreteObserver, Party * ConcreteObserver, Party, Table |

|  |  |
| --- | --- |
| ConcreteObserver | |
| **Responsibilities** | **Collaborators** |
| * Maintains the reference to a table * Keep states consistent with table state | * Table * Table, FullTableState, EmptyTableState |

|  |  |
| --- | --- |
| Table | |
| **Responsibilities** | **Collaborators** |
| * Record the current table state * Check new party in based on the current state and change its state accordingly * Check old party out based on the current state and change its state accordingly * When its state changes, notify the observer * Record the table size | FullTableState, EmptyTableState  FullTableState, EmptyTableState  FullTableState, EmptyTableState  Observer |

|  |  |
| --- | --- |
| FullTableState | |
| **Responsibilities** | **Collaborators** |
| * Maintain a reference to table * No action is token when checking new party in * Check party out and change the table state to empty | Table |

|  |  |
| --- | --- |
| EmptyTableState  (restaurant is full and have no waiting line, restaurant is full and have waiting line, restaurant is not full and have no waiting line, restaurant is not full and have waiting line) | |
| **Responsibilities** | **Collaborators** |
| * Maintain a reference to table * Check new party in if the party size can be accommodated and changes the table state to full * No action is token when checking party out | * Table |

|  |  |
| --- | --- |
| Party | |
| **Responsibilities** | **Collaborators** |
| * Record a String[] of their names * Record Party’s cell phone number * Size of the party * When a table is ready, be able to “confirm” or “leave” the wait list * Record a reference to current occupied table | * Restaurant |

Explanation:

For the Din Tai Fung problem, I choose to use observer and state pattern. Because basically there are only two states for a table, which are full and empty. Those two states should behave differently when checking new party in and checking old party out. A state pattern should work well for the Table object.

Also, a restaurant object should be notified if a table’s state changed from full to empty, so it can serve the next capable party in its waiting list. It is suitable to use an observer pattern to help the restaurant to check all the current state of its tables.

State Pattern Role:

Table plays the role of Context.

FullTableState, EmptyTableState plays the role of ConcreteState subclasses.

Observer Pattern Role:

Table plays the role of ConcreteSubject.

ConcreteObserver plays the role of ConcreteObserver.