|  |  |
| --- | --- |
| Restaurant | |
| **Responsibilities** | **Collaborators** |
| * Maintain a list of occupied tables * Maintain a list of empty tables * Maintain a queue of waiting parties * Maintain a list of serving parties * Record and change of current restaurant state (restaurant is not full and have no waiting line, restaurant is full and have no waiting line, restaurant is not full and have waiting line, restaurant is full and have waiting line) | * Table * Table * Party * Party * State |

|  |  |
| --- | --- |
| State  (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** |
| * Record the current restaurant state * Serving new party based on the current restaurant state, and changing the table states, waiting list, serving list and the restaurant state after serving * When party is leaving, change the current restaurant state * When one party is leaving, ask the next party in the waitlist that could be served with this table size if they would like to come, and change the restaurant state accordingly | * Restaurant * Restaurant, Party, Table * Restaurant, Party, Table * Restaurant, Party, Table |

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

|  |  |
| --- | --- |
| Table | |
| **Responsibilities** | **Collaborators** |
| * Record the table’s identification number * Record the capacity of the table |  |

Explanation:

For the Din Tai Fung problem, I choose to use state pattern. Because basically there are only two actions that the restaurant needs to take care of, which are checking the new party in and checking the serving party out. The behavior of these two actions really depend on the current restaurant states. The four states of restaurant are “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”. Two actions of the restaurant should perform different actions for those four states.

Restaurant plays the role of Context.

State plays the role of State.

“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” plays the role of ConcreteState subclasses.