Fast Food Simulator

This app simulates customers of a take-away restaurant placing orders and waiting for them to be prepared and delivered to a pickup counter. After placing an order a customer waits on the order to be announced before picking it up and proceeding to the dining area.

The user stories that make up this app center around four distinct roles:

- User - the end user using the application
- Customer - the simulated Customer
- Order Taker - the simulated Order Taker
- Cook - the simulated Cook
- Server - the simulated Server

Take the time to sketch out the UI, but how the different actors (roles) interact.

Constraints

- Order tickets can be represented as two different types of Promises - one the Server waits on while the Cook prepares the order and another the Customer waits on while in the serving line.
- You may NOT use a simulation package or library.
- New customers arrive in the order line at a fixed interval of time. In other words, new customers arrive at a constant rate.
- Order tickets are fulfilled at a fixed interval of time as well. They are completed at a constant rate.

User Stories

Application Operation

- User can see an input area that allows the entry of the time interval for customer arrival and a time interval for the fulfilment of an *order ticket* by the cook.
- User can see a customized warning message if the customer arrival interval or the order fulfilment interval is incorrectly entered.
- User can start the simulation by clicking on a Start button.
- User can see an order line area containing a text box showing the number of customers waiting to place orders.
- User can see an order area containing text boxes showing the *order number* currently being taken.
- User can see a kitchen area containing a text box showing the *order number* that's being prepared and a text box listing the waiting orders in sequence, along with a count of the number of waiting orders.
- User can see a Pickup area containing a text box showing the *order number* that's currently available for pickup by the Customer and a text box showing the number of Customers waiting in the serving line.
- User can stop the simulation at any time by clicking a Stop button.
- The user can see how the movement of customers, orders, etc. (graphical visualization)

Order Preparation

4. *Cook* removes the next ticket from the carousel containing the orders

5. *Cook* prepares the order

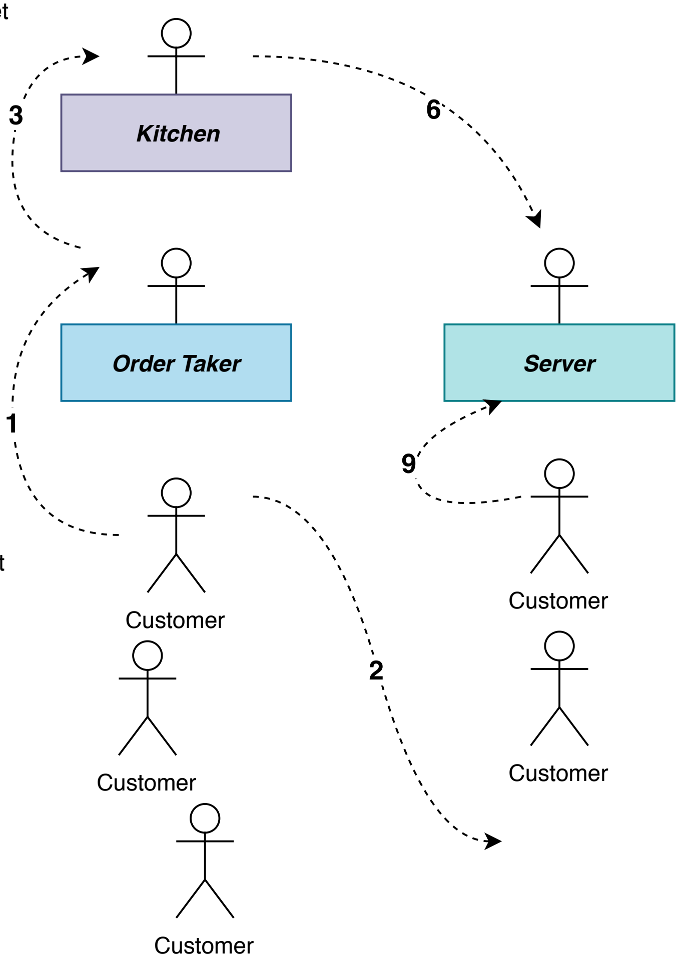
6. *Cook* places the ticket with the order and hands it to the server

Order Process

1. *Order Taker* creates a ticket for the customer order and giving receipt to the Customer.

2. *Customer* moves to the end of the serving line after paying for the order.

3. *Order Taker* places the ticket on a carousel at the kitchen window



Serving Process

7. *Server* takes the order and ticket from the Cook

8. *Server* calls out the order number from the ticket and places the completed order on the counter.

9. *Customer* picks up the order when the order number is called out and proceeds to the dining area.

<u>Technical Implementation</u>

- Order tickets can be represented as two types of Promises - one the Server waits on while the order is being prepared and one the Customer waits on until the order is ready to be served.
- A Customer process will generate an order based on a time interval. In other words, orders arrive at a constant rate.
- An Order Taker process will take the order, create a unique order number, add the order to the Kitchen queue (FIFO), and return the order number to the Customer.
- A Cook process will remove the next order from the Kitchen queue, prepare the order (also based on a fixed time interval), and then move it to the Service queue (FIFO) when it has been completed.
- A Server process will remove the next order from the Service queue and notify the Customer with the matching order number.

There's a lot going on here, but creating this app will provide you with a better understanding of Promises and also how to create applications that follow the SOLID principles, especially Separation of Concerns.