Engineering Challenge

Challenge Prompt

Create a real-time system that simulates the fulfillment of delivery orders for a kitchen. The system should receive 2 delivery orders per second (see *Order Data)*. Each order takes some time (defined as `prepTime` in order JSON) to be prepared. Once an order is prepared, it is waiting and ready for courier pickup.

Upon receiving an order, the system should immediately dispatch a courier to pick it up. Couriers arrive randomly following a uniform distribution, 3-15 seconds after they've been dispatched. Couriers have to wait at the kitchen if the order they are picking up is not ready. Once an order is picked up by a courier it is instantaneously delivered.

You are tasked with building two courier dispatch strategies and evaluating their performance.

- **Matched**: a courier is dispatched for a specific order and may only pick up that order
- **First-in-first-out**: a courier picks up the next available order upon arrival. If there are multiple orders available, pick up an arbitrary order. If there are no available orders, couriers wait for the next available one. When there are multiple couriers waiting, the next available order is assigned to the earliest arrived courier.

You must print the statistics below each time an order is picked up. After your system has finished processing all orders, an average of each of the statistics below must be printed.

- Average food wait time (milliseconds) between order ready and pickup
- Average courier wait time (milliseconds) between arrival and order pickup

You can use any programming language, framework, and IDE to demonstrate your best work. Consider this a real-time simulation as opposed to a discrete-event simulation. Please design your solution as a standalone executable. You're free to utilize any third party frameworks you deem appropriate.

We discourage the use of microservices, kafka, REST APIs, RPCs, DBs, and other machinery. We're looking for projects that demonstrate good object oriented programming fundamentals and software engineering judgement within the context of the Grading Rubric below.

Grading Rubric

- ❏ Meets all requirements from the *Challenge Prompt*
- ❏ Is valid, runnable code *(via CLI or IDE)*

- ❏ Is production-quality code; both interfaces and implementations are clean. Public interfaces are documented *(as if someone had to maintain your system)*
- ❏ Has appropriate usage of design patterns, concurrency, and data structures
- ❏ Has comprehensive unit testing
- ❏ Has a README that explains how to compile and run the system, how design decisions are made, and how to adjust the configurations under which it can be run.
- ❏ Has console output that allows **interviewers to *understand your system's operation* as it runs in real-time**. Output events (order received, order prepared, courier dispatched, courier arrived, order picked up) as they occur so the operation of your system can be understood as it runs

Submission

Please compress/zip your code and submit it to us via a GreenHouse link. You can find the link in your email with the subject "Coding Challenge from CSS".

Please do not place your submission on a public facing repository, such as GitHub/Bitbucket/etc.

Order Data

Download and run your system with this order JSON file https://bit.ly/css_dispatch_orders.

Schema

dispatch_orders.json

```
[
    {
        "id": "0ff534a7-a7c4-48ad-b6ec-7632e36af950",
        "name": "Cheese Pizza",
        "prepTime": 13          // Order preparation time in seconds.
    },
    ...
]
```