

## 1. 목표

order, food\_truck, food\_zone, start\_simulation의 코드를 완성하여 푸드 트럭 시뮬레이터가 실행되게 함

## 2. 과제를 해결하는 방법

메인 클래스에서 사용될 order, food\_truck, food\_zone를 완성하여 주문 개수, 가격, 요리 시간, 푸드 트럭, 손님 큐 등 정보들을 저장할 수 있게 한다.

## 3. 과제를 해결한 방법

각각의 클래스를 완성할 때 해당 클래스의 테스트 클래스를 보면서 작성하였다.

### 1) order.py

(1) \_\_init\_\_(self, order\_time)

test 클래스의 init에서 timestamp와 ordertime이 들어간 것을 보고 초기 값으로 order\_time과 order\_time을 설정하였다. order\_time은 지역 변수를 통해 입력하도록 하였다. 또한 1 이상 5 이하인 qty를 보고 randomint를 통해 1부터 5사이의 값을 저장하는 qty를 생성하였다.

(2) get\_order\_time(self)

\_\_init\_\_에서 저장된 order\_time을 반환하도록 하였다.

(3) get\_qty(self)

\_\_init\_\_에서 저장된 qty를 반환하도록 하였다.

(4) wait\_time(self, current\_time)

현재 시간을 입력받아 현재 시간에서 order\_time을 뺀 시간을 반환하도록 하였다.

### 2) food\_truck.py

(1) \_\_init\_\_(self, price, making\_time)

test 클래스의 init에서 FoodTruck에 price와 making\_time을 입력한 것을 보고 지역 변수로 price와 making\_time을 입력받도록 하였다. 또한, current\_order와 time\_remaining을 None과 0으로 비교하는 것을 보고 각각의 초기값을 None과 0으로 설정하였다.

(2) tick(self)

9부터 -1까지 내려가면서 남아있는 주문이 있다면 남은 시간을 비교하는 것을 보고 주문이 있다면 남은 시간을 -1씩 하면서 0이 될 때까지 비교하고 0 밑으로 가면 남은 주문이 없다는 것을 나타내기 위해 None이라 표시하였다.

(3) is\_busy(self)

주문이 없고, 남은 시간도 0이라면 작업 중이지 않은 상태라 판단하여 False라 반환하고, 나

머지 상황에 대해서는 작업 상태라 판단하여 True를 반환하였다.

(4) `start_next_food(self, next_order)`

`next_order`를 통해 다음 주문을 받아서 `current_order`에 저장하고, `time_remaining`에 `makint_time`과 다음 주문의 수량을 곱하여 저장한다.

### 3) `food_zone.py`

(1) `__init__(self, order_probability)`

test코드에서 주문 가능성인 `order_probability`를 입력받는 것을 보고 지역 변수로 넣었다. 또한 `assertEqual`에서 `food_trucks`, `order_queue`, `waiting_times`, `income`을 각각 `set`, `list`와 비교하는 것을 보고 각각을 `set`과 `list`로 초기화했다. 마지막으로 `self`의 `order_probability`에 지역 변수 `order_probability`를 저장했다.

(2) `add_truck(self, var_truck)`

`set`인 `food_trucks`에 `set`함수인 `add`를 이용해서 지역 변수로 받은 `var_truck` `set`에 추가한다.

(3) `is_new_order(self, picked_num = 0)`

0으로 초기화 된 지역 변수 `picked_num`을 입력하고 1부터 지역 변수인 `order_probability`까지의 정수 중 무작위로 한 숫자를 뽑아서 `picked_num`에 저장한다. 저장한 숫자가 `order_probability`와 같다면 True를 반환하고 그렇지 않다면 False를 반환한다.

### 4) `start_simulation(zone, test_second)`

`printer_simulator`와 유사하다 생각하여 비교하면서 하였다. 우선 `for` 반복문을 통하여 입력된 시간 내에서 무작위로 주문을 추가하여 푸드 존의 주문 리스트인 `order_queue`에 추가한다. 그 후 다시 `for` 반복문을 통하여 푸드 존에 속한 푸드 트럭 내에서 일하는 상태가 아니고, 주문이 남아있을 때, 현재 주문 중 첫 번째 주문을 `pop`을 통해 꺼내어 `new_order`에 저장한다. 또한 새 주문의 준비 시간만큼 푸드 존의 대기 시간에 추가한다. 다음으로 `start_next_food`를 통하여 새 주문을 시작한다. 그리고 새 주문의 가격과 양을 각각 `price`와 `qty`에 저장하고 두 개를 곱한 값을 푸드 존의 수입에 추가한다. 마지막으로 `tick`을 이용하여 푸드 트럭에서 작업을 시작한다.

## 4.결과화면

1) `order_test`

```
order_test x
C:\Users\HyunTaek\PycharmProjects\week03\venv\Scripts\python.exe C:/Users/HyunTaek/PycharmProjects/week03/order_test.py
test__init__ (__main__.OrderTest) ... ok
test_get_order_time (__main__.OrderTest) ... ok
test_get_qty (__main__.OrderTest) ... ok
test_wait_time (__main__.OrderTest) ... ok

-----
Ran 4 tests in 0.000s

OK

Process finished with exit code 0
```

## 2) food\_truck\_test

```
food_truck_test x
C:\Users\HyunTaek\PycharmProjects\week03\venv\Scripts\python.exe C:/Users/HyunTaek/PycharmProjects/week03/food_truck_test.py
test__init__ (__main__.FoodTruckTest) ... ok
test_is_busy (__main__.FoodTruckTest) ... ok
test_start_next_food (__main__.FoodTruckTest) ... ok
test_tick (__main__.FoodTruckTest) ... ok

-----
Ran 4 tests in 0.000s

OK

Process finished with exit code 0
```

## 3) food\_zone\_test

```
food_zone_test x
C:\Users\HyunTaek\PycharmProjects\week03\venv\Scripts\python.exe C:/Users/HyunTaek/PycharmProjects/week03/food_zone_test.py
test__init__ (__main__.FoodZoneTest) ... ok
test_add_truck (__main__.FoodZoneTest) ... ok
test_is_new_order (__main__.FoodZoneTest) ... ok

-----
Ran 3 tests in 0.001s

OK

Process finished with exit code 0
```

## 4) food\_main

```
food_main x
C:\Users\HyunTaek\PycharmProjects\week03\venv\Scripts\python.exe C:/Users/HyunTaek/PycharmProjects/week03/food_main.py
----# Food zone 결과(영업 시간: 1440.0, 손님 확률: 대략 300초당 1명) #----
- Food price: 2,000, making time: 60
Average wait:    160.42 secs
Order remain:    2
Total income: 103,902,000 Won
Setting cost: 40,000,000 Won
Recipe cost: 31,170,600 Won
Staff cost: 12,024,000 Won
Profit: 20,707,400 Won

Process finished with exit code 0
```