DATA

```
List<Order> orders;

class Order {
	Cook c;
	List<FoodOrder> orders;
	orderState s;
}

class Food {
	String type;
	int inventory;
}

enum OrderState { none, received, processed, readyForDelivery, delivered };

Timer timer; // For delivery times

Map<String, Food> foods;
```

MESSAGES

```
FoodOrder(CookAgent c, List<FoodOrder> orders) {
	orders.add(new Order(c, order));
}
```

SCHEDULER

```
if there is an o in orders such that o.s = received
	processOrder(o);

if there is an o in orders such that o.s = processed
	prepareOrder(o);
```

ACTIONS

```
processOrder(Order o) {
	for(FoodOrder fo : o.orders) {
		Food tempFood = foods.get(fo.foodType);
		if tempFood.inventory = 0;
```

```
                        fo.amount = 0;

                if tempFood.inventory > fo.amount
                        tempFood inventory -= fo.amount;

                if tempFood.inventory < fo.amount
                        fo.amount = tempFood.inventory;
                        tempFood.inventory = 0;

        if none of the orders can be fulfilled
                o.c.msgCannotFulfillOrder(o.orders);
                o.s = none;
        else
                o.c.msgWeWillDeliver(this, o.orders);
                o.s = processed;
}

prepareOrder(Order o) {
        o.s = readyForDelivery;
        timer.run(deliverFood(o), 10000); //wait 10 seconds to send food
}

deliverFood(Order o) {
        o.c.msgFoodDelivery(this, o.orders);
        o.s = delivered;
}
```