Doreen Hakimi
CS201 Restaurant V2.1
Fall 2013

DATA

```
int mktnum;
Map<String, Food> marketFood
List<InventoryOrder> myOrders
Cook cook;

class InventoryOrder {
        List<FoodOrder> order;
        enum state = {pending, processed, delivered};
        int ORDERID;
}
```

MESSAGES

```
HereIsAnInventoryorder(List<FoodOrder> orderToMarket, int id) {
        myOrders.add(new InventoryOrder(orderToMarket, id));
}
```

SCHEDULER

```
if there exists an InventoryOrder i in myOrders such that i.state=pending
        then ProcessOrder(i);
```

ACTIONS

```
ProcessOrder(InventoryOrder tobeProcessed) {
        List<FoodOrder> orderList=tobeprocessed.myOrder;
        List<FoodOrder> deliveryList;
        List<FoodOrder> notFulfilledList;

        for each FoodOrder foo in orderList
                Food temp. marketFood.get(foo.choice);
                if(foo.val<=temp.amount)
                        temp.amount-=foo.val;
                        marketFood.put(foo.choice, temp);
                        deliveryList.add(foo);
                else
                        if(temp.amount>0)
                                deliveryList.add(new FoodOrder(foo.choice, temp.amount);
                                notFulfilledList.add(new FoodOrder(foo.choice,
foo.val-temp.amount));
                        else
                                notFulfilledList.add(foo);
```

```
        tobeprocessed.state=processed;
        if(!notFulfilled.empty())
                cook.CouldNotFulfillThese(noFulfilledList, tobeprocessed.ORDERID);
        if(!deliveryList.empty())
                timer.start() {
                        cook.HereIsYourFoodOrder(deliveryList);
                        myOrders.remove(tobeprocessed);
                }
}
```