Kartik Chillakanti
CS 201

COOK AGENT

Data

```
enum state {pending, cooking, done, sent};
class CookOrder {
        WaiterAgent waiter;
        String choice;
        int table;
        State state;
}
class Food {
        string choice;
        int cookingTime;
        boolean orderPending = false;
}
List<CookOrder> orders;
List<MarketAgent>markets;
Timer timer;
Map (String choice, Food f) foodMap;
Int marketNum = 0;
```

Kartik Chillakanti
CS 201

COOK AGENT

Scheduler

if there exists o in orders such that o.state == done then PlateIt(o);

if there exists o in orders such that o.state == pending then CookIt(o);

```
if there exists a Food in foodmap such that ((f.amount <= 0) && (!f.orderPending)) {
        orderFromMarket(f.choice);
        f.orderPending = true;
}
```

Kartik Chillakanti
CS 201

COOK AGENT

Messages

```
msgHereIsAnOrder(WaiterAgent w, string choice, int table) {
        orders.add(new Order(w, choice, table, state.pending);
}

msgOrderFulfilled(string type, int amount) {
        foodMap.get(type).amount += amount;
        foodMap.get(type).orderPending = false;
}

msgOrderUnfulfilled() {
        marketNum++;
        if (marketNum == 3)
                marketNum = 0;
        }
}
```

Kartik Chillakanti
CS 201

COOK AGENT

Actions

```
CookIt(Order o){
        timer.start() // create a TimerTask and override run function
        o.state = done;
}


PlateIt(Order o){
        o.w.OrderIsReady(choice, table);
        orders.remove(o);
}


private void orderFromMarket(String type) {
        markets.get(marketNum).msgHereIsMarketOrder(type, 5);
}


public void addMarkets(List<MarketAgent> markets) {
        this.markets = markets;
}

public void drainInventory() {
        for (each food in foodMap) {
                food.amount = 0;
        }
}
```