DATA

```
Class MyCustomer {
        Customer c;
        int table;
        CustomerState s;
        string choice;

        Check check;
}

HostAgent host;

CookAgent cook;

CashierAgent cashier;

List<MyCustomer> customers;

enum CustomerState = { waiting, seated, readyToOrder, askedForOrder, ordered,
orderSentToCook, orderOut, foodReady, served, checkReady, checkGiven, finished,
leftRestaurant };

enum breakState { none, wantABreak, askedForBreak, onBreak, doneWithBreak };
breakState breakStatus = none;

Semaphore atDestination;
Semaphore customerAtTable;

enum waiterState { working, onBreak };
waiterState state = working;

enum waiterEvent { none, backToWork, takeABreak };
waiterEvent = none;
```

MESSAGES

```
PleaseSeatCustomer(Customer c, int table) {
        if customer is not in list,
                customers.add(new MyCustomer(c, table, waiting));
        else
                c.s = waiting;
                c.table = table;
}
```

```
ImReadyToOrder(Customer c) {
	MyCustomer mc = customers.find(c);
	mc.s = readyToOrder;
}

HereIsMyChoice(Customer c, string choice) {
	MyCustomer mc = customers.find(c);
	mc.s = ordered;
	mc.choice = choice;
}

OrderDone(string choice, int t) {
	MyCustomer mc = customer at table t;
	mc.s = foodReady;
}

HereIsCheck(Check c) {
	MyCustomer mc = customers.find(c.cust);
	mc.check = c;
	mc.s = checkReady;
}

OutOf(string choice, int table) {
	MyCustomer mc = customer at table t;
	mc.s = orderOut;
}

ImDoneEating(Customer c) {
	MyCustomer mc = customers.find(c);
	mc.s = finished;
}

IWantABreak() { //Called by button on the gui
	breakStatus = wantABreak;
}

SorryNoBreakNow() {
	breakStatus = none;
}

FinishUpAndTakeABreak() {
	event = takeABreak;
}
```

```
BreakIsFinished() { //Called by button on the gui
        breakStatus = doneWithBreak;
}

msgAtDestination() { // Message from animation
        atDestination.release();
}

msgCustomerSatDown() { // Message from animation
        customerAtTable.release();
}
```

SCHEDULER

```
if breakStatus = doneWithBreak
        finishBreak();

if breakStatus = wantABreak
        askForBreak();

If there is a c in customers such that c.s = finished
        tellHostCustomerIsDone(c);

If there is a c in customers such that c.s = foodReady
        bringFoodToCustomer(c);

If there is a c in customers such that c.s = checkReady
        getCheckFromCashier(c);

If there is a c in customers such that c.s = waiting
        seatCustomer(c);

If there is a c in customers such that c.s = readyToOrder
        takeOrder(c);

If there is a c in customers such that c.s = orderOut
        removeChoice(mc);
        askToReorder(mc);

If there is a c in customers such that c.s = ordered
        sendOrderToCook(c);

if event = takeABreak and state = working and allCustomersDone()
```

```
        state = onBreak;
        takeABreak();

if event = backToWork and state = onBreak;
        state = working;
        event = none;
```

ACTIONS

```
seatCustomer(MyCustomer c) {
        c.c.followMe(this, new Menu());
        DoSeatCustomer(c);
        c.s = seated;
        DoLeaveCustomer();
}

takeOrder(MyCustomer c) {
        DoGoToTable(c.table);
        c.c.WhatDoYouWant();
        c.s = askedForOrder;
        DoLeaveCustomer()
}

removeChoice(MyCustomer c) {
        c.c.RemoveFromMenu(c.choice);
}

askToReorder(MyCustomer c) {
        DoGoToTable(c.table);
        c.s = askedForOrder;
        c.c.PleaseReorder();
        DoLeaveCustomer();
}

sendOrderToCook(MyCustomer c) {
        cook.HereIsOrder(this, c.choice, c.table);
        c.s = foodOrdered;
}

bringFoodToCustomer(MyCustomer c) {
        DoGoToCook();

        DoGoToTable(c.table);
        c.c.HereIsYourFood(c.choice);
```

```
        c.s = served;

        DoLeaveCustomer();
}

tellHostCustomerIsDone(MyCustomer c) {
        host.tableIsFree(c.table, this);
        c.s = leftRestaurant;
        c.table = 0; // Customer is not at a table
}

askForBreak() {
        host.IWantABreak(this);
        breakStatus = askedForBreak;
}

takeABreak() {
        DoGoToBreakArea();
        breakStatus = onBreak;
}

finishBreak() {
        event = backToWork;
        breakStatus = none;
        host.ImDoneWithMyBreak(this);
}

getCheckFromCashier(MyCustomer c) [
        DoGoToCashier();
        giveCustomerCheck(c);
}

giveCustomerCheck(MyCustomer c) {
        DoGoToTable(c.table);
        c.c.HereIsYourBill(c.check);
}

allCustomersDone() {
        if all customers are finished or have left restaurant
                return true;
        else
                return false;
}
```