Kartik Chillakanti
CS 201

WAITER AGENT

Data

```
enum CustomerState {waiting, seated, readyToOrder, asked, ordered, orderGiven,
done};
class MyCustomer {
        CustomerAgent c;
        int t;
        string c;
        CustomerState s;}
List<MyCustomer> customers;
List<WaiterOrder> readyOrders;
List<Check> checks;
WaiterGui waiterGui;
HostAgent host;

Class WaiterOrder {
        String choice;
        int table;
}
```

Kartik Chillakanti
CS 201

WAITER AGENT

<u>Scheduler</u>

```
if !onBreak {

if there exists c in customers such that c.s ==  gone then customers.remove(c)

if there exists c in customers such that c.s ==  ordered then GiveOrderToCook(c);

if there exists c in customers such that c.s ==  unpaid then DoDeliverCheck(c);

if there exists c in customers such that c.s ==  doneEating then prepareCheck(c);

if there exists c in customers such that c.s == waiting then seatCustomer(c);

if there exists c in customers such that c.s == readyToOrder then TakeOrder(c);

if there exists c in customers such that c.s ==  notAvailable then
TellCustomerFoodUnavailble(c);

if there exists c in customers such that c.s == done then DoLeaveCustomer(c);


if there exists o in readyOrders then TakeFoodToCustomer();

if (WantBreak) {
        if !pendingActions
                host.msgIWantABreak(this);
}
```

Kartik Chillakanti
CS 201

WAITER AGENT

<u>Messages</u>

```
msgSitAtTable(CustomerAgent c, int table) {
        customers.add(new MyCustomer(c,table, waiting));
}

msgImReadyToOrder(CustomerAgent c) {
        MyCustomer mc = customers.find(c);
        mc.s = readyToOrder;
}

msgHereIsMyChoice(String choice, CustomerAgent c) {
        MyCustomer mc = customers.find(c);
        mc.choice = choice;
        mc.state = ordered;
}

msgOrderIsReady(String choice, int table) {
        readyOrders.add(new WaiterOrder(choice, table);
}

msgDoneEating(CustomerAgent c){
        for(MyCustomer mc:customers){
                if(mc.c.equals(c)){
                        mc.s = CustomerState.doneEating;
                }
        }
}

msgLeaving(CustomerAgent c) {
        for (MyCustomer mc : customers)
        {
                if (c.equals(mc.c)) {
                host.msgTableIsFree(mc.t);
                host.msgLeaving(mc.c);
                mc.s = CustomerState.gone;
                }
        }
}
```

Kartik Chillakanti
CS 201

WAITER AGENT

```
msgImOutOfFood(int table) {
        for (MyCustomer mc : customers)
        {
                if (mc.t == table) {
                        mc.s = CustomerState.notAvailable;
                }
        }
}

msgBreakApproved() {
                onBreak = true;
}

msgBreakDenied() {
        onBreak = false;
}

msgHereIsComputedCheck(Check c) {
        checks.add(c);
}
```

Kartik Chillakanti
CS 201

WAITER AGENT

Actions

** Included animations in design since they are closely tied with the semaphores that dictate waiter's processes **

```
seatCustomer(MyCustomer c) {
        c.c.FollowMe(new Menu());
        DoSeatCustomer(c); // animation
        c.s = seated;
        DoLeaveCustomer(); //animation
}

TakeOrder(MyCustomer c) {
        DoGoToTable(c.t); // animation
        c.c.WhatWouldYouLike();
        c.s = asked;
}

GiveOrderToCook(MyCustomer c) {
        c.s = orderGiven;
        DoGoToCook() // animation
        cook.msgHereIsAnOrder (this, c.choice, c.t);
        c.s = orderGiven;
}

TakeFoodToCustomer(MyCustomer c)
{
        WaiterGui.GiveFoodToCustomer(c); // animation
        if there exists c in customers such that c.s != done and
                readyOrders.get(0).table = c.table then
                        c.HereIsYourFood();
                        ProcureFood(); // animation
                        DoGoToTable(); // animation
                        serveFood() // animation
                        c.msgHereIsYourFood
                        readyOrders.remove(0);
                        DoLeaveCustomer // animation

}

DoSeatCustomer(CustomerAgent customer, int tableNumber)
{
        WaiterGui.DoBringToTable(c.getGui(), tableNumber);
}
```

Kartik Chillakanti
CS 201

WAITER AGENT

```
prepareCheck(MyCustomer customer) {
            customer.s = CustomerState.done;
            waiterGui.DoClearTable(customer.t);
            cashier.msgGiveOrderToCashier(customer.choice,
            customer.t, customer.c, this);
}

DoDeliverCheck(Check c) {
            waiterGui.DoGoToTable(c.tableNum);
            c.c.msgHereIsCheck(c);
            c.state = CheckState.delivered;
}
```