DATA

```
string choice = random choice from menu;

double money = random number under 50;

Check check;

Timer timer; // Timer for eating food

Semaphore atDestination;

Menu menu;

enum agentState { doingNothing, waitingInRestaurant, beingSeated, seated, ordering,
reordering, eating, doneEating, payingBill, leaving };

enum agentEvent { none, gotHungry, followWaiter, seated, askedToOrder, ordered, reordering,
startedEating, doneEating, payBilll, leaving };

agentState state = doingNothing;
agentEvent event = none;

HostAgent host;
WaiterAgent waiter;
```

MESSAGES

```
RestaurantFull() {
    int randChoice = ranGenerator.nextInt(2);

    if(randChoice == 0) {
        event = leaving;
    }
    else {
        print("I'm in no hurry. I can wait.");
    }
}

GotHungry() {
    event = gotHungry;
}

FollowMe(Waiter w, Menu m) {
        waiter = w;
        menu = m;
        event = followWaiter;
}

msgAnimationFinishedGoToSeat() {
        event  = seated;
}

WhatDoYouWant() {
        event = askedToOrder;
}

PleaseReorder() {
        event = reordering;
}

RemoveFromMenu(String choice) {
        menu.remove(choice); //For when the cook runs out of a food
}

HereIsYourBill(Check c) {
        check = c;
}

HereIsYourFood(string choice) {
        event = startedEating;
}

msgAnimationDoneEatingFood() {
        event = doneEating;
}

atDestination() {
        atDestination.release();
}
```

<u>SCHEDULER</u>

```
if state = doingNothing and event = gotHungry
        state = waitingInRestaurant;
        goToRestaurant();
```

```
if state = waitingInRestaurant and event = leaving
        state = leaving;
        leaveRestaurant();

if state = waitingInRestaurant and event = followWaiter
        state = beingSeated;
        sitDown();

if state = beingSeated and event = seated
        state = seated;
        readyToOrder();

if state = seated and event = askedToOrder
        state = ordering;
        orderFood();

if state = ordering and event = reordering
        state = reordering
        readyToOrder();

if state = reordering and event = askedToOrder
        state = ordering;
        reorderFood();

if state = ordering and event = leaving
        state = leaving;
        leaveRestaurant();

if state = ordering and event = startedEating
        state = eating;
        eatFood();

if state = eating and event = doneEating
        state = doneEating;
        tellWaiterImDone();

if state = doneEating and event = payBill and check != null
        state = payingBill;
        payBill();

if state = payingBill and event = leaving
        state = leaving;
        leaveRestaurant();
```

```
if state = leaving and event = leaving
    state = doingNothing;
    event = none;
```

ACTIONS

```
goToRestaurant() {
    DoGoToRestaurant();
    host.ImHungry(this);
}

sitDown() {
    DoGoToSeat();
}

readyToOrder() {
    run timer for a few seconds
    waiter.ImReadyToOrder(this);
}

orderFood() {
    choice = random food; // Steak, chicken, or fish
    if choice is too expensive
        if another food is cheap enough
            choice = another food
        else //Leave because all food is too expensive
            waiter.ImDoneEating(this);
            event = leaving
            return;

    waiter.HereIsMyChoice(this, choice);
}

reorderFood() {
    choice = random food that is cheap enough
        waiter.HereIsMyChoice(this, choice);
    if no foods are cheap enough or everything has run out
        waiter.ImDoneEating(this);
        event = leaving;
}

eatFood() {
        DoEatingFood();
        run timer for a few seconds
        event = doneEating;
}

tellWaiterImDone() {
        event = payBill;
        waiter.ImDoneEating(this);
}

leaveRestaurant() {
        DoLeaveRestaurant();
}

payBill() {
        DoGoToCashier();

        if(money > check.amount) {
                check.cashier.payBill(check, check.amount);
                money -= check.amount;
        }
        else {
                check.cashier.payBill(check, money);
                money = 0;
        }

        event = leaving;
}
```