Design Document

## Host:

Data:

**Int NTABLES = 3;**

**List<CustomerAgent> Customers;**

**Collection<Table> tables;**

**List<MyWaiter> Waiters;**

**enum WaiterState{Pending,OnBreak, Working};**

**String name;**

**Semaphore atTable ;**

**Class Table{**

**CustomerAgent occupiedBy;**

**Int tableNumber;**

**}**

**Class MyWaiter{**

**WaiterAgent w;**

**Int CustNum;**

**WaiterState s;**

**}**

Message:

**msgIWantFood(CustomerAgent cust){**

**Customers.add(cust);**

**}**

**msgLeavingTable(int seatNum){**

**for(Table table : tables){**

**if (table.tableNumber == seatNum){**

**table.setUnoccupied();**

**}**

**}**

**}**

**msgWaiterOnBreak(WaiterAgent w){**

**for(MyWaiter mw :Waiters){**

**if(mw.w == w){**

**mw.s = WaiterState.Pending;**

**}**

**}**

**msgWaiterOffBreak(WaiterAgent w){**

**for(MyWaiter mw :Waiters){**

**if(mw.w == w){**

**mw.s = WaiterState.Working;**

**}**

**}**

Scheduler:

**If there exists a table in Table such that !table.isOccupied(),**

**If there exists a customer in Customers,**

**If there exists a waiter in Waiters such that waiter.CustNum is min,**

**Then waiter.CustNum++; seatCustomer(waiter, customer);**

**If there exists a waiter in MyWaiter such than waiter.s = Pending, then WaiterIsOnBreak(waiter)**

Action:

**seatCustomer(WaiterAgent w, CustomerAgent Customer, Table table)**

**{**

**w.msgSeatCustomer(customer, table.tableNumber);**

**table.setOccupant(customer);**

**Customers.remove(customer);**

**}**

**WaiterIsOnBreak(MyWaiter w){**

**Do("WaiterIsOnBreak");**

**w.w.msgOnBreak();**

**w.s = WaiterState.OnBreak;**

}

## Waiter:

Data:

**List<MyCustomer> Customers;**

**Boolean origin = false;**

**Boolean atcook = false;**

**String name;**

**int seatnum;**

**Semphore atTable = new Semaphore(0,true);**

**CookAgent Cook;**

**WaiterGui waiterGui = null;**

**HostAgent host;**

**Enum CustomerState{DoingNothing, Waiting, Seated, ReadyToOrder, Asked, Ordered, BeingServed, Served, Leaving, Reorder}**

**Class MyCustomer{**

**CustomerAgent c;**

**Int table;**

**String Choice;**

**CustomerState ;**

**}**

Message:

**msgSeatCustomer(CustomerAgent c , int table){**

**Customers.add(new MyCustomer(c,table,CustomerState.Waiting));**

**}**

**msgIWantFood(CustomerAgent c) {**

**for(MyCustomer mycust : Customers){**

**if(mycust.getCust() == c)**

**{**

**MyCustomer mc = mycust;**

**mc.s = CustomerState.ReadyToOrder;**

**}**

**}**

**}**

**msgHereIsMyOrder(CustomerAgent c, String Choice){**

**for(MyCustomer mycust : Customers){**

**if(mycust.getCust() == c)**

**{**

**MyCustomer mc = mycust;**

**mc.Choice = Choice;**

**mc.s = CustomerState.Asked;**

**seatnum = mc.table;**

**}**

**}**

**}**

**msgFoodReady(String Choice, int table){**

**for(MyCustomer mycust : Customers){**

**if(mycust.table == table && mycust.Choice == Choice){**

**mycust.s = CustomerState.BeingServed;**

**}**

**}**

**}**

**msgLeavingTable(CustomerAgent c) {**

**for(MyCustomer mycust : Customers){**

**if(mycust.getCust() == c)**

**{**

**mycust.s = CustomerState.Leaving;**

**}**

**}**

**msgAtTable() {**

**atTable.release();**

**}**

**msgIsorigin(){**

**origin = true;}**

**msgAtCook(){**

**atcook = true;}**

**msgIWantToPay(CustomerAgent c){**

**Do("msgIWantToPay");**

**for(MyCustomer mycust : Customers){**

**if(mycust.getCust() == c){**

**mycust.s = CustomerState.Paying;**

**}**

**}**

Scheduler:

**If there exists a customer in MyCustomer such that customer.s = Leaving, then NotifyHost(customer);**

**If there exists a customer in MyCustomer such that customer.s = waiting and origin = true, then origin = false, seatCustomer(customer);**

**If there exists a customer in MyCustomer such that customer.s = ReadyToOrder, then TakeOrder(customer);**

**If there exists a customer in MyCustomer such that customer.s = Asked, then GiveOrder(customer);**

**If there exists a customer in MyCustomer such that customer.s = WaitingForfood, then WaitCook();**

**If there exists a customer in MyCustomer such that customer.s = BeingServed and atcook = true, then atcook = false, ServeFood(customer);**

**If there exists a customer in MyCustomer such that customer.s = Paying, then TakeCheck(customer);**

**If there exists a customer in MyCustomer such that customer.s = reorder, then Reorder(customer);**

Action:

**seatCustomer(MyCustomer c) {**

**waiterGui.DoBringToTable(c.c,c.table);**

**c.c.msgFollowMe(new Menu(),c.table,this);**

**atTable.acquire();**

**waiterGui.DoLeaveCustomer();**

**c.s = CustomerState.Seated;**

**}**

**TakeOrder(MyCustomer c){**

**waiterGui.DoBringToTable(c.c,c.table);**

**c.c.msgWhatDoYouWant();**

**c.s = CustomerState.Ordered;**

**}**

**GiveOrder(MyCustomer c){**

**waiterGui.DoBringToTable(c.c,c.table);**

**atTable.acquire();**

**waiterGui.DoGoToCook();**

**Cook.msgCookOrder(this, c.Choice, c.table);**

**c.s = CustomerState.WaitingForfood;**

**}**

**WaitCook(){**

**waiterGui.DoGoToCook();**

**}**

**ServeFood(MyCustomer c){**

**waiterGui.msgshowOrder(c.Choice);**

**waiterGui.DoBringToTable(c.c, c.table);**

**atTable.acquire();**

**c.c.msgHereIsTheFood();**

**c.s = CustomerState.Eating;**

**waiterGui.DoLeaveCustomer();**

**waiterGui.msghideOrder();**

**}**

**NotifyHost(MyCustomer c){**

**host.msgLeavingTable(c.table);**

**c.s = CustomerState.DoingNothing;**

**c.table = 0;**

**c.Choice = null;**

**}**

**TakeCheck(MyCustomer c){**

**Do("Taking The Check");**

**waiterGui.DoBringToTable(c.c, c.table);**

**try {**

**atTable.acquire();**

**} catch (InterruptedException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**waiterGui.DoGoToCashier();**

**cashier.msgComputeCheck(this, c.c, c.Choice);**

**c.s = CustomerState.Leaving;**

**}**

**ReOrder(MyCustomer c){**

**Do("Let Customer Reorder");**

**waiterGui.DoBringToTable(c.c, c.table);**

**c.c.msgReorder(menu);**

**c.s = CustomerState.Ordered;**

**}**

## Customer:

Data:

**Int tablenum;**

**Int ThinkingTime = 8000;**

**Int EatingTime = 5000;**

**String Choice;**

**Menu menu;**

**String name;**

**Int hungerLevel = 5;**

**Timer timer;**

**CustomerGui customerGui;**

**WaiterAgent waiters;**

**HostAgent host;**

**Enum AgentState {DoingNothing, WaitingInRestaurant, BeingSeated,Ordered,Eating, DoneEating, Leaving, TakingOrder};**

**AgentState state = AgentState.DoingNothing;**

**enum AgentEvent**

**{none, gotHungry, followHost, seated,ordering,beingServed, doneEating, doneLeaving, TakingOrder};**

**AgentEvent event = AgentEvent.none;**

Message:

**gotHungry() {**

**event = AgentEvent.gotHungry;**

**}**

**msgFollowMe(Menu m,int table,WaiterAgent w) {**

**this.waiters = w;**

**seatnum = table;**

**event = AgentEvent.followHost;**

**}**

**msgWhatDoYouWant() {**

**event = AgentEvent.TakingOrder;**

**}**

**msgHereIsTheFood() {**

**event = AgentEvent.beingServed;**

**}**

**msgAnimationFinishedGoToSeat() {**

**event = AgentEvent.seated;**

**}**

**msgAnimationFinishedLeaveRestaurant() {**

**event = AgentEvent.doneLeaving;**

**}**

**msgReorder(Menu m){**

**Do("msgReorder");**

**this.menu = m;**

**Random r = new Random();**

**int setChoice = r.nextInt(menu.menu.size());**

**Choice = m.getName(setChoice);**

**event = AgentEvent.TakingOrder;**

**state = AgentState.TakingOrder;**

**stateChanged();**

**}**

Scheduler:

**If state = DoingNothing and event = gotHungry, then state = WaitingInRestaurant; goToRestaurant();**

**If state = WaitingInRestaurant and event = followHost, then state = BeingSeated; SitDown();**

**If state = BeingSeated and event = ordering, then state = TakingOrder;**

**TakeOrder();**

**If state = TakingOrder and event = TakingOrder, then state = Ordered;**

**HereIsMyOrder();**

**If state = Ordered and event = AgentEvent.beingServed, then state = Eating; EatFood();**

**If state = Eating and event doneEating, then state = Leaving; leaveTable();**

**If state = Leaving and event = doneLeaving, then state = DoingNothing;**

Action:

**goToRestaurant() {**

**host.msgIWantFood(this);**

**}**

**SitDown() {**

**customerGui.DoGoToSeat(seatnum);**

**timer.schedule(new TimerTask(){**

**public void run(){**

**event = AgentEvent.ordering;**

**stateChanged();**

**}**

**}, ThinkingTime);**

**}**

**TakeOrder(){**

**customerGui.msgSetOrder();**

**waiters.msgIWantFood(this);**

**}**

**HereIsMyOrder(){**

**waiters.msgHereIsMyOrder(this, Choice);**

**customerGui.msgCancelSignal();**

**}**

**void EatFood() {**

**timer.schedule(new TimerTask() {**

**public void run() {**

**event = AgentEvent.doneEating;**

**stateChanged();**

**}**

**},**

**EatingTime);**

**customerGui.msgShowOrder();**

**}**

**leaveTable() {**

**waiters.msgLeavingTable(this);**

**customerGui.DoExitRestaurant();**

**customerGui.msgHideOrder();**

**}**

## Cook:

Data:

**enum CookState{Pending, Cooking, Plating, Done};**

**enum FoodState{Out, Plenty, Asked, Replenished}**

**CookState state = CookState.Pending;**

**Timer timer;**

**String name;**

**List<Order> order;**

**List<Food>food;**

**WaiterAgent waiter;**

**Boolean done = false;**

**Map<String, Double> map;**

**Market market;**

**Market market2;**

**Market market3;**

**Class Order{**

**WaiterAgent w;**

**String Choice;**

**Int table;**

**CookState s;**

**}**

**Class Food{**

**String Choice;**

**Int CookingTime;**

**Int inventory;**

**FoodState s;**

**}**

Message:

**msgCookOrder(WaiterAgent w, String Choice, int table){**

**order.add(new Order(w,Choice,table, CookState.Pending));**

**}**

**msgFoodDone(Order o ){**

**o.s = CookState.Done;**

**}**

**msgHereIsDelivery(String choice, int number){**

**Do("Food Replenished");**

**for(Food f : food){**

**if(f.Choice.equals(choice)){**

**f.number += number;**

**f.s = FoodState.Replenished;**

**Do(choice+f.number);**

**}**

**}**

**msgMarketNoFood(){**

**int count = 0;**

**count++;**

**if(count == 1)**

**this.market = market1;**

**if(count == 2)**

**this.market = market2;**

**if(count == 3)**

**noFood = true;**

**stateChanged();**

**}**

Scheduler:

**If there exists an o in order such that o.s = Pending, then CookIt(o);**

**If there exists an o in order such that o.s = Cooking, then msgFoodDone(o);**

**If there exists an o in order such that o.s = Done, then PlateIt(o);**

**If there exists a f in food such that f.s = out, then AskMarket(f)**

**If there exists a o in Order such that o.s = out, then TellWaiter(o);**

Action:

**CookIt(Order o){**

**o.s = CookState.Cooking;**

**long time = (map.get(o.Choice)).longValue();**

**timer.schedule(new TimerTask(){**

**public void run(){**

**done = true;**

**stateChanged();**

**}**

**}, time);**

**}**

**msgFoodDone(Order o );**

**o.s = CookState.Done;**

**stateChanged();**

**}**

**PlateIt(Order o) {**

**o.w.msgFoodReady(o.Choice,o.table);**

**order.remove(o);**

**}**

**TellWaiter(Order o){**

**o.w.msgOutOfFood(o.Choice, o.table);**

**order.remove(o);**

**}**

**AskMarket(Food f){**

**market.msgOutOfFood(f.Choice);**

**f.s = FoodState.Asked;**

**}**

## Cashier:

Data:

**String name;**

**Menu m;**

**List<MyCustomer> customer;**

**Enum CustomerState{Unpay, Paying, Paied};**

**Class MyCustomer{**

**WaiterAgent waiter;**

**CustomerAgent c;**

**Double check;**

**String Choice;**

**CustomerState s;**

**Double Cash = 0;**

**Double Change = 0;**

**}**

Message:

**msgComputeCheck(WaiterAgent w,CustomerAgent c, String Choice){**

**double price = 0;**

**for(int i=0; i< m.menu.size(); i++){**

**if(m.getName(i) == Choice){**

**price = m.getPrice(i);**

**}**

**}**

**customer.add(new MyCustomer(w,c,Choice,CustomerState.Unpay, price));}**

**msgPay(CustomerAgent c, double cash){**

**for(MyCustomer mycust : customer){**

**if(mycust.getCust() == c)**

**{**

**MyCustomer mc = mycust;**

**mc.s = CustomerState.Paying;**

**mc.setCash(cash);**

**}**

**}**

Scheduler:

**If there exists a mycust in Customer such that mycust.s = Paying, then ComputeCheck(mycust);**

**If there exists a mycust in Customer such that mycust.s = Paied, then GiveChange(mycust);**

Action:

**ComputeCheck(MyCustomer c) {**

**double change = 0;**

**change = c.Cash - c.check;**

**c.setChange(change);**

**}**

**GiveChange(MyCustomer c){**

**c.c.msgGiveChange(c.Change);**

**}**

## Market:

Data:

**String name;**

**List<MyFood>Food;**

**CookAgent cook;**

**Timer timer;**

**Enum FoodState{Pending, Sending, ReadyToSend};**

**Boolean send = false;**

**Class MyFood{**

**String Choice;**

**Int inventory;**

**FoodState s;**

**}**

Message:

**msgOutOfFood(String Choice){**

**Do("msgCookOutOfFood");**

**for(MyFood f : food){**

**if(f.Choice.equals(Choice)){**

**if(f.inventory == 0){**

**Do("Run out of Food go to next market");**

**cook.msgMarketNoFood();**

**}**

**else**

**{**

**Do("Prepare to ship the food");**

**f.s = FoodState.ReadyToSend;**

}

}

}

Scheduler:

If there exists a food in MyFood such that food.s = ReadyToSend, then PrepareFood(food);

If there exists a food in MyFood such that food.s = Sending && send = true, then send = false, DeliverFood(food)

Action:

**PrepareFood(MyFood f){**

**Do("Preparing Food");**

**timer.schedule(new TimerTask() {**

**public void run() {**

**send = true;**

**stateChanged();**

**}**

**},**

**5000);**

**f.s = FoodState.Sending;}**

**DeliverFood(MyFood f){**

**Do("Delivering Food");**

**cook.msgHereIsDelivery(f.Choice,20);**

**f.s = FoodState.Pending;**

**}**