Waiters

2013년 9월 17일 화요일

오후 11:16

Messages

```
// 2:
msgSitAtTable(cust, table) {
    customer.add(cust, table);
}

// 4:
msgReadyToOrder(cust) {
    If exists in customers such that customer.c = cust
        Then customer.state = readyToOrder;
}

// 6:
msgHereIsMyChoice(cust, choice) {
    If exists in customer such that cutomer c= cust
        Then
            MyCustomer.state = waitingFood1;
            Customer.choice = choice;
            state = waiting;
}

// 8:
msgOrderIsReady(cust, choice) {
    If exists in customer such that cutomer c= cust
        Then customer.state = FoodIsReady;
}

// 10:
msgIAmDone() {
    If exists in customer such that cutomer c= cust
        Then customer.state = doneEating;
}
```

Data

```
List<MyCustomer> customers;

Cook cook;
Host host;

Set menu;

enum AgentState
    {Waiting, Serving}
AgentState state = Waiting;

Class MyCustomer {
    Customer c;
    Table t;
    String choice ;

    public enum CustState
            {Waiting, seated, readyToOrder, waitingFood1,
            waitingFood2, foodIsReady, eating, doneEating};
    CustState state = CustState.Waiting;
}
```

Scheduler

```
if (state == AgentState.Waiting) {
    for (MyCustomer customer : MyCustomers) {
        if (customer.state == MyCustomer.CustState.Waiting) {
            state = AgentState.Serving;
            SitAtTable(customer);
        }
        else if (customer.state == MyCustomer.CustState.readyToOrder) {
            state = AgentState.Serving;
            WhatWouldYouLike(customer);
        }
        else if (customer.state == MyCustomer.CustState.waitingFood1) {
            state = AgentState.Serving;
            HereIsAnOrder(this, customer);
        }
        else if (customer.state == MyCustomer.CustState.foodIsReady) {
            state = AgentState.Serving;
            HereIsYourOrder(customer);
        }
        else if (customer.state == MyCustomer.CustState.doneEating) {
            TableIsCleared(customer);
        }
    }
}
```

Actions

```
SitAtTable(cust) {
    State = Serving;
    customer.msgFollowMe(menu);
    DoSeatCutomer(cust);

    state = Waiting;
}

DoSeatCustomer(cust) {
    Customer.state = seated;
    Customer.t.setOccupant(cust);
    State = Waiting;
}

WhatWouldYouLike() {
    Customer.msgWhatWouldYouLike();
}

HereIsAnOrder(this.waiter, customer) {
    Cook.msgHereIsAnOrder(new order(waiter, customer));
    State = Waiting;
}

HereIsYourOrder(customer) {
    Customer.msgHereIsYourOrder();
    Customer.state = eating;
    State = Waiting;
}

TableIsCleared(customer) {
    Customers.remove(customer);
    Host.msgTableIsCleared(customer.t);
}
```

Customers

2013년 9월 17일 화요일

오후 11:16

Messages

```
// 0:
IAmHungry() {
        Event = gotHungry
}

// 3:
msgFollowMe(menu) {
        Choices = menu;
        Event = followHost;
}

// 5:
msgWhatWouldYouLike() {
        Event = makeOrder;
}

 // 9:
msgHereIsYourOrder() {
        Event = getFood;
}
```

Scheduler

```
if (state == AgentState.DoingNothing && event == AgentEvent.gotHungry ){
                state = AgentState.WaitingInRestaurant;
                goToRestaurant();
        }
else if (state == AgentState.WaitingInRestaurant && event == AgentEvent.followHost ){
                state = AgentState.BeingSeated;
                SitDown();
        }
else if (state == AgentState.BeingSeated && event == AgentEvent.seated){
                state = AgentState.Seated;
                ChooseMenu();
        }
else if (state == AgentState.Seated && event == AgentEvent.callWaiterToOrder){
                state = AgentState.ReadyToOrder;
                ReadyToOrder();
        }
else if (state == AgentState.ReadyToOrder && event == AgentEvent.makeOrder){
                state = AgentState.WaitingFood;
                HereIsMyChoice(choice);
        }
else if (state == AgentState.WaitingFood && event == AgentEvent.getFood){
                state = AgentState.Eating;
                EatFood();
        }
else if (state == AgentState.Eating && event == AgentEvent.doneEating){
                state = AgentState.Leaving;
                leaveTable();
        }
else if (state == AgentState.Leaving && event == AgentEvent.doneLeaving){
                state = AgentState.DoingNothing;
        }
```

Data

```
String name;
HostAgent host;
Waiter wait;
Set<String> menu;
String choice;

public enum AgentState
        {DoingNothing, WaitingInRestaurant, BeingSeated, Seated,
        ReadyToOrder, WaitingFood, Eating, DoneEating, Leaving};
private AgentState state = AgentState.DoingNothing;//The start state

public enum AgentEvent
        {none, gotHungry, followHost, seated, callWaiterToOrder,
        makeOrder, getFood, doneEating, doneLeaving};
AgentEvent event = AgentEvent.none;
```

Actions

```
goToRestaurant() {
        Host.msgIWantFood(this.cust);
}

SitDown() {
        State = Seated;
        Event = seated;
}

ChoseMenu() {
        Choice = choices.get(random);
        event = AgentEvent.callWaiterToOrder;
}

ReadyToOrder(cust) {
        Wait.msgReadyToOrder(this.cust);
}

HereIsMyChoice(cust, choice) {
        Wait.msgHereIsMyChoice(this.cust, choice);
}

EatFood() {
    State = Eating;
    Timer.start( doneEating() );
}

leaveTable() {
    wait.msgLeavingTable(this);
}
```

Host

2013년 9월 17일 화요일

오후 11:16

Messages

```
// 1:
msgIWantFood(cust){
        WaitingCustomer.add(cust);
}

// 11:
msgTableIsCleared (table) {
        Table.unoccupied();
}
```

Scheduler

```
If exits a table in tables such that table is empty
        If exits customer in waitingCustomers
                Then tellWaiter(customer, table);
```

Data

```
List<Customer> WaitingCustomers;
List<Table> Tables;
List<Waiter> Waiters;

Waiter wait;

Class Table() {
        Int tableNumber;
        Customer occupiedby;
}
```

Actions

```
tellWaiter(customer, table) {
        wait.msgSitAtTable(cust, table);
        WaitingCustomer.remove(0);
}
```

Cook

2013년 9월 17일 화요일

오후 11:16

Messages

```
// 7
msgHereIsAnOrder(order) {
    Orders.add(order);
}
```

Scheduler

```
If there exits in Order
    If state == pending
        CookOrder(order);
    Else if state == cooked
        msgOrderIsReady(order);
```

Data

```
List<Order> orders;

Class Order {
    Waiter w;
    Customer cust;
    Food choice;

    Enum OrderStatus:Pending, Cooking, Cooked;
}

Class Food {
    String name;
    Int time;
}
```

Actions

```
CookOrder(order) {
    DoCooking(order);
}

DoCooking(order) {
    state = cooking;
    Time.start { Done(Order) };
    state = cooked;
}
```