**Waiter Agent**

Data

class MyCustomer {

CustomerAgent customer;

int table;

String choice;

double dues;

CustomerState state = Waiting;

CookEvent event = none;

CashierEvent cashierEvent = none;

}

List<MyCustomer> myCustomers;

Semaphore waitingForOrder;

Semaphore walkingToCook;

Semaphore atTable;

Semaphore atCashier;

Semaphore getBill;

Semaphore customerSeated;

Semaphore readyToTakeOrder;

Semaphore seatingCustomer;

enum CustomerState {

Waiting, Seated, ReadyToOder, AskedToOrder, WaitingForOrder, Paying

};

enum CookEvent {

None, DeliveringOrder, OrderReady, OutOfStock

};

enum CashierEvent {

None, GetBill, BillReady

};

enum WaiterState {

None, RequestingBreak, BreakDenied, BreakOver

};

WaiterState state;

WaiterEvent event;

HostAgent host;

CookAgent cook;

Messages

PleaseSeatCustomer(HostAgent host, CustomerAgent customer, int table) {

this.host = host;

cook = host.getCook();

myCustomers.add(new MyCustomer(customer, table));

}

ReadyToOrder(CustomerAgent customer) {

if Ǝ mC in myCustomers ϶ mc.customer = customer

mC.state = ReadyToOder;

}

HereIsMyOrder(String choice, CustomerAgent customer) {

if Ǝ mC in myCustomers ϶ mc.customer = customer

waitingForOrder.release();

mC.choice = choice;

mC.state = WaitingForOrder;

mC.event = DeliveringOrder;

}

OrderReady(String choice, int table) {

if Ǝ mC in myCustomer ϶ mC.choice = choice and mC.table = table

mC.event = OrderReady;

}

OutOf(String choice, int table) {

if Ǝ mC in myCustomers ϶ mC.choice = choice and mC.table = table

mC.event = OutOfStock;

}

HereIsBill(CustomerAgent customer, double cost) {

if Ǝ mC in myCustomers ϶ mC.customer = customer

getBill.release();

mC.cashierEvent = BillReady;

mC.dues = cost;

}

DoneAndPaying(CustomerAgent customer) {

if Ǝ mC in myCustomers ϶ mc.customer = customer

host.TableFree(mC.table);

mC.remove(myCustomer);

}

CantAffordAnythingElse(CustomerAgent customer) {

waitingForOrder.release();

if Ǝ mC in myCustomers ϶ mC.customer = customer

host.TableFree(mC.table);

myCustomers.remove(mC);

}

WantToGoOnBreak() {

event = RequestingBreak;

}

BreakOkay() {

state = WaitingForBreak;

}

BreakNotOkay() {

state = Available;

event = BreakDenied;

}

WantToGoOffBreak() {

event = BreakOver;

}

BreakOver() {

state = Available;

}

Scheduler

if state = Available and event = RequestingBreak

askHostForBreak();

if state = Available and event = BreakDenied

DoBreakDenied();

if state = WaitingForBreak and myCustomers.isempty

goOnBreak();

if state = OnBreak and event = BreakOver

goOffBreak();

if Ǝ mC in myCustomers ϶ mC.state = Waiting

seatCustomer(mC);

if Ǝ mC in myCustomers ϶ mC.state = ReadyToOrder

takeOrder(mC);

if Ǝ mC in myCustomers ϶ mC.state = WaitingForOrder and mC.event = DeliveringOrder

processOrder(mC);

if Ǝ mC in myCustomers ϶ mC.state = WaitingForOrder and mC.event = OrderReady

deliverFood(mC);

if Ǝ mC in myCustomers ϶ mC.state = WaitingForOrder and mC.event = OutOfStock

deliverNewMenu(mC);

if Ǝ mC in myCustomers ϶ mC.cashierEvent = GetBill

getBill(mC);

if Ǝ mC in myCustomers ϶ mC.cashierEvent = BillReady

giveCustomerBill(mC

Actions

seatCustomer(MyCustomer myCustomer) {

DoSeatCustomer(myCustomer);

myCustomer.customer. FollowMe(this, new Menu(), myCustomer.table);

myCustomer.state = Seated;

customerSeated.acquire();

}

takeOrder(MyCustomer myCustomer) {

DoGoToTable(myCustomer);

readyToTakeOrder.acquire();

myCustomer.customer.WhatDoYouWant();

myCustomer.state = AskedToOrder;

waitingForOrder.acquire();

}

deliverNewMenu(MyCustomer myCustomer) {

DoGoToTable(myCustomer);

readyToTakeOrder.acquire();

myCustomer.customer.OutOfChoice(new Menu(myCustomer.choice));

myCustomer.state = AskedToOrder;

waitingForOrder.acquire();

}

processOrder(MyCustomer myCustomer) {

DoTellCookOrder();

walkingToCook.acquire();

myCustomer.event = None;

cook.startThread();

cook.HereIsAnOrder(this, myCustomer.choice, myCustomer.table);

}

deliverFood(MyCustomer myCustomer) {

myCustomer.event = CookEvent.None;

DoDeliverFood(myCustomer);

atTable.acquire();

myCustomer.customer.HereIsYourFood(myCustomer.choice);

}

getBill(MyCustomer myCustomer) {

DoGoToTable();

atTable.acquire();

myCustomer.cashierEvent = none;

myCustomer.customer.HereIsYourBill(myCustomer.dues);

}

askHostForBreak() {

host.WantToGoOnBreak(this);

event = none;

}