**Cashier Agent**

Data

List<PayingCustomer> payingCustomers;

List<PayingCustomer> customersWhoDidntPay;

double steakCost;

double chickenCost;

double pizzaCost;

double saladCost;

Food steak, chicken, pizza, salad;

enum State {none, Bill, Payment};

Map<String,Food> foods;

class PayingCustomer {

WaiterAgent waiter;

CustomerAgent customer;

String choice;

boolean indebt;

double debt;

State state;

}

class Food {

String type;

double cost;

}

Messages

GetBill(WaiterAgent waiter, CustomerAgent customer, String choice) {

payingCustomers.add(new PayingCustomer(waiter, customer, choice));

}

Payment(CustomerAgent customer, double cash) {

if Ǝ myCustomer in payingCustomer ϶ myCustomer.customer = customer

myCustomer.state = Payment;

myCustomer.cash = cash;

}

Scheduler

if Ǝ myCustomer in payingCustomer ϶ myCustomer.state = Bill

getBill(myCustomer)

if Ǝ myCustomer in payingCustomer ϶ myCustomer.state = Payment

calculatePayment(myCustomer)

Actions

getBill(PayingCustomer myCustomer) {

myCustomer.state = none;

double cost = foods.get(myCustomer.choice).cost;

if (myCustomer.inDebt)

cost += myCustomer.debt;

myCustomer.waiter.HereIsBill(myCustomer.customer, cost);

}

calculatePayment(PayingCustomer myCustomer) {

myCustomer.state = none;

double change = myCustomer.cash – myCustomer.choice.cost – myCustomer.debt

if (change < 0) {

myCustomer.inDebt = true;

myCustomer.debt = abs(change);

change = 0;

myCustomer.customer.InDebt(myCustomer.debt);

} else {

myCustomer.inDebt = false;

myCustomer.debt = 0;

myCustomer.customer.NotInDebt();

}

myCustomer.customer.HereIsYourChange(change);

}