Data

List<MyCheck> computedChecks;

List<Check> uncomputedChecks;

Double money;

class MyCheck {

Check check;

Double amountPaid;

}

Map<String, Double> priceMap;

Scheduler

if there exists c in uncomputedChecks such that c.state == CheckState.uncomputed then {

c.state = CheckState.unpaid;

computeCheck(c);

}

if there exists c in computedChecks such that c.state == CheckState.paid then {

processCheck(c);

}

Messages

msgGiveOrderToCashier (String c, int tNum, CustomerAgent cust, WaiterAgent waiter) {

uncomputedChecks.add(new Check(c, tNum, cust, waiter));

}

msgPayingcheck(Check check, Double amountPaid) {

if there exists c in computedChecks such that c.check = check then

c.amountPaid = amountPaid;

c.check.state = CheckState.paid;

}

Actions

void computeCheck(Check c){

computedchecks.add(new MyCheck(c));

c.price = priceMap.get(c.choice);

if (c.c.oweMoney) {

if there exists c in computedChecks such that c.check.state == incomplete and c.check .c == c.c

c.price += c.c.check.price;

c.w.msgHereIsComputedCheck(c);

}

void processCheck (Mycheck c) {

if (c.check.price <= c.amountPaid) {

money += c.amountPaid;

c.check.state = CheckState.done;

}

else {

c.check.state = CheckState.incomplete

c.check.c.msgPunish();

}

}