HOST AGENT

Data

```
static final int NTABLES;

public List<CustomerAgent> waitingCustomers
= new ArrayList<CustomerAgent>();

public List<WaiterAgent> allWaiters
= new ArrayList<WaiterAgent>();

public Collection<Table> tables;

public enum AgentState
{DoingNothing, SeatingCustomer};
private AgentState state;

private String name;

public HostGui hostGuiI;

private int LeastBusyWaiter;

private class Table {
        CustomerAgent occupiedBy;
        int tableNumber;
}
```

Messages

```
public void msgImFree(WaiterAgent w) {
        stateChanged();
}

public void msgIWantFood(CustomerAgent cust) {
        waitingCustomers.add(cust);
        stateChanged();
}
```

```
public void msgLeavingTable(CustomerAgent cust) {
        for (Table table : tables) {
                if (table.getOccupant() == cust) {
                        print(cust + " leaving " + table);
                        table.setUnoccupied();
                        stateChanged();
                }
        }
}
```

Scheduler

```
1.)     for (Table table : tables) {
                if (!table.isOccupied()) {
                        if (!waitingCustomers.isEmpty()) {

        seatCustomer( allWaiters.get(LeastBusyWaiter), waitingCustomers.get(0), table);

                                LeastBusyWaiter++;

                                if(LeastBusyWaiter >= allWaiters.size()) {
                                        LeastBusyWaiter=0;
                                }
                                return true;
                        }
                }
        }
2.)     return false;
```

Actions

```
private void seatCustomer(WaiterAgent waiter, CustomerAgent customer, Table table) {
        waiter.msgNewCustomerToSeat(customer, table.getTable());
        print(waiter.getName() + " seating " + customer + " at " + table);
        table.setOccupant(customer);
        waitingCustomers.remove(customer);
}
```