DATA

```
List<MyCustomer> customers;

List<MyWaiter> waiters;

class MyCustomer {
	Customer c;
	boolean waiting = true;
	boolean toldRestaurantIsFull // Give customers a chance to leave if all tables are full
}

class MyWaiter {
	Waiter w;
	int numCustomers = 0; // Keeps track of how many customers the waiter is serving
	bool iWantABreak;
	waiterState state = working;
}

enum waiterState { working, onBreak };

List<Table> tables;

int numberOfWorkingWaiters = 0; // To make sure there's always at least 1 working waiter

class Table {
	int tableNumber;
	boolean occupied;
}
```

MESSAGES

```
ImHungry(Customer c) {
	if c is not in customers
		customers.add(new MyCustomer(c));
	else
		c.s = waiting;
}

IwantABreak(WaiterAgent w) {
	MyWaiter mw = waiters.find(w);
	mw.iWantABreak = true;
}
```

```
ImDoneWithMyBreak(WaiterAgent w) {
        MyWaiter mw = waiters.find(w);
        mw.iWantABreak = false;
        mw.state = working;
        numberOfWorkingWaiters++;
}

ImLeaving(CustomerAgent c) {
        MyCustomer mc = customers.find(c);
        mc.waiting = false;
}

TableIsFree(int table, Waiter w) {
        Table table = tables.find(table);
        table.occupied = false;

        MyWaiter waiter = waiters.find(w);
        w.numCustomers--;
        // decrease number of customers that waiter is dealing with by 1
}
```

SCHEDULER

```
if there is a w in waiters such that w.iWantABreak
        giveWaiterABreak(w);

if there is a c in customers such that c.s = waiting;
        if there is a t in tables such that t.occupied = false;
                if waiters is not empty
                        seatCustomer(c, t)

if there is a c in customers such that c.s = waiting and !mc.toldRestaurantIsFull
        tellCustomerRestaurantIsFull(mc);
```

ACTIONS

```
seatCustomer(MyCustomer c, Table table) {
        MyWaiter w = waiters.findLeastBusyWaiter(w)
        // This part finds the waiter with the lowest numCustomers who is not on a break

        // Assigns the customer to the least busy waiter
        w.PleaseSeatCustomer(c.c, table.tableNumber);
        w.numCustomers++; // Waiter has one more customer
```

```
        c.s = seated;
        table.occupied = true;
}

giveWaiterABreak(MyWaiter w) {
        w.iWantABreak = false;
        if there is only 1 waiter or numberOfWorkingWaiters = 1
                w.w.SorryNoBreakNow();

        w.w.FinishUpAndTakeABreak;
        w.s = onBreak;
        numberOfWorkingWaiters--;

tellCustomerRestaurantIsFull(MyCustomer c) {
        c.toldRestaurantIsFull = true;
        c.c.msgRestaurantFull();
```