# HOST AGENT

## Data

```
static final int NTABLES = 3;//a global for the number of tables.
public List<CustomerAgent> waitingCustomers
= new ArrayList<CustomerAgent>();

public List<Boolean> messaged = new ArrayList<Boolean>();

public List<MyWaiters> allWaiters
= new ArrayList<MyWaiters>();
public Collection<Table> tables;
public RestaurantGui RestGUI;

public enum AgentState
{DoingNothing, SeatingCustomer};
private AgentState state = AgentState.DoingNothing;//The start state

private String name;

public HostGui hostGui = null;

private int LeastBusyWaiter=0;
private int WaiterOnBreak=-1;

private class MyWaiters {
        WaiterAgent waiter;
        boolean WantABreak;
        boolean OnBreak;
        int NumberOfCustomers;

        MyWaiters (WaiterAgent w) {
                waiter = w;
                WantABreak =false;
                OnBreak = false;
                NumberOfCustomers = 0;
        }

        public WaiterAgent GetWaiter() {
                return waiter;
        }
```

```
	public int GetNumberOfCustomers() {
		return NumberOfCustomers;
	}

	public boolean GetOnBreak() {
		return OnBreak;
	}

	public boolean GetWantABreak() {
		return WantABreak;
	}

	public void IWantABreak() {
		WantABreak = true;
	}

	public void TakeABreak() {
		OnBreak = true;
		WantABreak = false;
	}

	public void BackFromBreak() {
		OnBreak = false;
	}

	public void AddCustomer() {
		NumberOfCustomers++;
	}

	public void CustomerDone() {
		NumberOfCustomers--;
	}
}


private class Table {
	CustomerAgent occupiedBy;
	int tableNumber;

	Table(int tableNumber) {
		this.tableNumber = tableNumber;
	}
```

```
        public int getTable() {
                return tableNumber;
        }

        void setOccupant(CustomerAgent cust) {
                occupiedBy = cust;
        }

        void setUnoccupied() {
                occupiedBy = null;
        }

        CustomerAgent getOccupant() {
                return occupiedBy;
        }

        boolean isOccupied() {
                return occupiedBy != null;
        }

        public String toString() {
                return "table " + tableNumber;
        }
}
```

## Messages

```
public void msgNewWaiter(WaiterAgent w) {
        allWaiters.add(new MyWaiters(w));
        stateChanged();
}

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


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

```
	}

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

		for ( int i =0; i<allWaiters.size();i++) {
			if (w == allWaiters.get(i).GetWaiter()) {
				allWaiters.get(i).CustomerDone();
			}
		}
		stateChanged();

	}

	public void msgIWantABreak(WaiterAgent w) {
		for ( int i =0; i<allWaiters.size();i++) {
			if (w == allWaiters.get(i).GetWaiter()) {
				allWaiters.get(i).IWantABreak();
				print(allWaiters.get(i).GetWaiter().getName() + " wants a break");
			}
		}
		stateChanged();
	}

	public void msgBackFromBreak(WaiterAgent w) {
		for ( int i =0; i<allWaiters.size();i++) {
			if (w == allWaiters.get(i).GetWaiter()) {
				allWaiters.get(i).BackFromBreak();
				print(allWaiters.get(i).GetWaiter().getName() + " is back from
break");
				WaiterOnBreak = -1;
			}
		}
		stateChanged();
	}

	public void msgIWontWait(CustomerAgent c) {
		print(c + " does not want to wait and has left");
```

```
            waitingCustomers.remove(c);
            stateChanged();
        }
```

## Scheduler

1.)
```
for ( int i =0; i<allWaiters.size();i++) {
                if (allWaiters.get(i).GetWantABreak() == true &&
allWaiters.get(i).GetNumberOfCustomers()==0) {
                        if (WaiterOnBreak == -1 && allWaiters.size()>1) {
                                Break(i);
                                return true;
                        }
                }
        }
```

2.)
```
if(!allWaiters.isEmpty())
        {
        for (Table table : tables) {
                if (!table.isOccupied()) {
                        if (!waitingCustomers.isEmpty()) {
                                LeastBusyWaiter = 0;
                                if (WaiterOnBreak == 0) LeastBusyWaiter=1;
                                for ( int i =0; i<allWaiters.size();i++) {
                                        if (WaiterOnBreak == i) i++;
                                        if (allWaiters.get(i).GetNumberOfCustomers() <
allWaiters.get(LeastBusyWaiter).GetNumberOfCustomers() && i != WaiterOnBreak) {
                                                LeastBusyWaiter = i;
                                        }
                                }
                                seatCustomer( allWaiters.get(LeastBusyWaiter).GetWaiter(),
waitingCustomers.get(0), table);
                                allWaiters.get(LeastBusyWaiter).AddCustomer();

                                return true;
                        }
                }
        }
```

```
}
```

3.)
```
if(!waitingCustomers.isEmpty()){
        for (Table table : tables) {
                if (!table.isOccupied()) {
                        return false;
                }
        }

        for (int i = 0; i<waitingCustomers.size();i++) {
                if (messaged.get(i) == false) {
                        messaged.set(i, true);
                        RestaurantFull(i);
                }
        }
}
```

4.)
```
return false;
```

## Actions

```
private void Break(int i) {
        allWaiters.get(i).TakeABreak();
        allWaiters.get(i).GetWaiter().msgTakeABreak();
        WaiterOnBreak=i;
        RestGUI.OnABreak(allWaiters.get(i).GetWaiter());
        print(allWaiters.get(i).GetWaiter().getName() + " is taking a break");
}

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);
        messaged.remove(0);
}

private void RestaurantFull(CustomerAgent c) {
```

```
            waitingCustomers.get(i).msgRestaurantFull();
    }
```