WAITER AGENT

Data

```
public enum AgentState
{DoingNothing, SeatingCustomer, GetOrder, TakeOrderToCook, ServeFood,
CleanTable};

public AgentState state;

public enum Event
{NewCustomerToSeat, DoneSeating, customerReady, GotOrder, FoodReady,
customerDone};

private Event event = Event;
private Event currentEvent;

public enum CustState {Seating, Seated, ReadyToOrder, Ordered, WaitingForFood,
OrderOut, Eating, Done, Leaving, Gone};

public List<MyCustomers> myCustomers
= new ArrayList<MyCustomers>();  //see below

private List<Event> allEvents = new ArrayList<Event>();

public int tablenumber;

private HostAgent host;

private CookAgent cook;

public boolean busy;

private MyCustomers CurrentCustomer;

private Semaphore atTable = new Semaphore(0,true);
private Semaphore atCook = new Semaphore(0,true);
private Semaphore working = new Semaphore(0,true);
String name;

private Menu currentMenu = new Menu();
```

```
public WaiterGui waiterGuil;

public class Menu {
        String optionOne;
        String optionTwo;
        String optionThree;
        String optionFour;

}

private class MyCustomers {
        CustomerAgent customer;
        int tableNumber;
        String order;
        CustState currentState;

        MyCustomers(CustomerAgent customer, int tableNumber) {
                this.tableNumber = tableNumber;
                this.customer = customer;
        }

}
```

Messages

```
public void msgNewCustomerToSeat(CustomerAgent cust, int table){
        event = Event.NewCustomerToSeat;
        allEvents.add(event);
        CurrentCustomer= new MyCustomers(cust,table);
        CurrentCustomer.setState(CustState.Seating);
        myCustomers.add(CurrentCustomer);
        tablenumber=table;
        stateChanged();
}
```

```
public void msgLeavingTable(CustomerAgent cust) {
	for (MyCustomers myCustomer : myCustomers) {
		if (myCustomer.getCustomer() == cust) {
			myCustomer.setState(CustState.Done);
		}
	}
	event=Event.customerDone;
	allEvents.add(event);
	stateChanged();
}

public void msgAtTable() {	//from animation
	atTable.release();
	stateChanged();
}

public void msgReadyToOrder(CustomerAgent cust) {
	for (MyCustomers myCustomer : myCustomers) {
		if (myCustomer.getCustomer() == cust) {
			myCustomer.setState(CustState.ReadyToOrder);
		}
	}

	event=Event.customerReady;
	allEvents.add(event);
	stateChanged();
}

public void msgOrderFood(CustomerAgent cust, String food) {
	for (MyCustomers myCustomer : myCustomers) {
		if (myCustomer.getCustomer() == cust) {
			myCustomer.setState(CustState.Ordered);
			myCustomer.setOrder(food);
		}
	}
	event=Event.GotOrder;
	allEvents.add(event);
	stateChanged();
 }
```

```
public void msgAtFront(){

        this.state = AgentState.DoingNothing;
        busy=false;
        host.msgImFree(this);
        stateChanged();
}

public void msgAtCook(){
        atCook.release();
        stateChanged();
}

public void msgOrderReady(String food, int table) {
        print ("Order Ready");
        for (MyCustomers myCustomer : myCustomers) {
                if (myCustomer.getTableNumber() == table) {
                        myCustomer.setState(CustState.Eating);
                        CurrentCustomer = myCustomer;
                        }
        }

        event=Event.FoodReady;
        allEvents.add(event);
        stateChanged();
}
```

Scheduler

1. if there exists an Event E within allEvents such that E == Event.GotOrder && state ==AgentState.GetOrder
        Then: busy = true;
        currentEvent = E;
        state = AgentState.TakeOrderToCook;
        TakeOrderToKitchen(CurrentCustomer);
        allEvents.remove(currentEvent);
        currentEvent=null;
        return true;

2. if there exists an Event E within allEvents such that E == Event.FoodReady && state ==AgentState.DoingNothing
    Then: busy = true;
    currentEvent = E;
    state = AgentState.ServeFood;
    DeliverFoodToTable(CurrentCustomer);
    allEvents.remove(currentEvent);
    currentEvent=null;
    return true;
3. if there exists an Event E within allEvents such that E == Event.NewCustomerToSeat && state ==AgentState.DoingNothing
    Then: busy = true;
    currentEvent = E;
    state = AgentState.SeatingCustomer;
    seatCustomer();
    allEvents.remove(currentEvent);
    currentEvent=null;
    return true;
4. if there exists an Event E within allEvents such that E == Event.customerReady && state ==AgentState.DoingNothing
    Then: busy = true;
    currentEvent = E;
    state = AgentState.GetOrder;
    GetCustomerOrder();
    allEvents.remove(currentEvent);
    currentEvent=null;
    return true;
5. if there exists an Event E within allEvents such that E == Event.customerDone&& state ==AgentState.DoingNothing
    Then: busy = true;
    currentEvent = E;
    state = AgentState.CleanTable;
    PrepareTable();
    allEvents.remove(currentEvent);
    currentEvent=null;
    return true;
6. else return false;

Actions

```
	private void seatCustomer() {
CurrentCustomer=GetCurrentCustomer(); //unsubstanited here

CurrentCustomer.getCustomer().msgFollowMeToTable(this, currentMenu);
DoSeatCustomer(CurrentCustomer.getCustomer(), CurrentCustomer.getTableNumber());
		try {
			atTable.acquire();
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
		busy=true;
		this.state=AgentState.SeatingCustomer;
//semaphore call clears state and busy calls, recalled at correct values
		waiterGui.DoLeaveCustomer();
	}

	// The animation DoXYZ() routines
	private void DoSeatCustomer(CustomerAgent customer, int table) {
			waiterGui.DoBringToTable(customer, table);

	}

	public void GetCustomerOrder() {
		CurrentCustomer=GetCurrentCustomer();
		waiterGui.DoGoToTable(CurrentCustomer.getCustomer(),
	CurrentCustomer.getTableNumber()); //animation
				try {
			atTable.acquire();
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		busy=true;
		this.state=AgentState.GetOrder;

		CurrentCustomer.setState(CustState.Ordered);

		CurrentCustomer.getCustomer().msgHereForOrder();
		print("Here for order");
	}
```

```
	public void TakeOrderToKitchen(MyCustomers current) {
		print("Taking order to chef");
		waiterGui.BringOrderToCook(current.getOrder());
		try {
			atCook.acquire();
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		busy=true;
		this.state=AgentState.TakeOrderToCook;
		current.setState(CustState.WaitingForFood);
		cook.msgNewOrder(this,current.getTableNumber(), current.getOrder());
		waiterGui.DoLeaveCustomer();
		print("Chef has order");
	}


   public void DeliverFoodToTable(MyCustomers current) {
	waiterGui.DoGoToCook();
	try {
			atCook.acquire();
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	waiterGui.BringFoodToCustomer(current.getCustomer(),
current.getTableNumber(),current.getOrder());

	try {
			atTable.acquire();
		} catch (InterruptedException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}

		busy=true;
		this.state=AgentState.ServeFood;
		current.setState(CustState.Eating);
		current.getCustomer().msgDeliveredFood();
	waiterGui.DoLeaveCustomer();
	print("Delivered Food");
```

```
    }

        public void PrepareTable() {
                print("Table empty");
                CurrentCustomer=GetCurrentCustomer();
                CurrentCustomer.setState(CustState.Leaving);
                        waiterGui.DoGoToTable(CurrentCustomer.getCustomer(),
CurrentCustomer.getTableNumber());
                try {
                        atTable.acquire();
                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                busy=true;
                this.state=AgentState.CleanTable;
                host.msgLeavingTable(CurrentCustomer.getCustomer());
                CurrentCustomer.setState(CustState.Gone);
                print("Table clean");
                waiterGui.DoLeaveCustomer();
         }
```