MULTI-AGENT SYSTEMS 2013-2014

ASSESSED COURSEWORK

(Submission by groups of up to three students allowed/welcome)

Due date: Tuesday 18 February 2014

Electronic submission

The main aim of the coursework is to introduce JADE and to increase familiarity with agents, multi-agent systems and agent communication by means of a (simple) **hospital appointment allocation system**. You will define a 'hospital agent' (and associated 'behaviours') that is able to issue appointments and a 'patient agent' (and associated 'behaviours') that is able to request an appointment from a hospital agent as well as swap an appointment it is allocated for another (allocated to another patient agent) more preferable one. Minimal disclosure of information and maximal satisfaction of preferences are key factors.

Note for the electronic submission:

- *Save all files in a directory named 'jadeCW' (note 'j' is lower case) and specify package in source files as such, i.e. 'package jadeCW'.*
- ***Submit 'jadeCW.zip'***

Step 1

Create class **HospitalAgent** that extends the **Agent** class.

Define **setup()** method of **HospitalAgent** to do the following:

- Read in input argument, which is a single integer representing the total number of appointments available;
- Register service of type "allocate-appointments" with the Directory Facilitator (DF) (see yellowPages DFRegisterAgent example in JADE documentation).

Note: *You may assume for the duration of the coursework that there will only be one HospitalAgent instance for any running of the appointment allocation application.*

Step 2

Create class **PatientAgent** that extends the **Agent** class.

Define **setup()** method of **PatientAgent** to do the following:

- Read in and store input arguments, which are positive integers and hyphens separated by whitespace, representing the agent's preferences over appointments, e.g. "2 1 - 3 7 6 - 5 -" represents that the agent prefers appointment 2 or 1 most, then appointment 3, 7 or 6, then appointment 5, and all other appointments thereafter are equally (dis)liked; (an empty set of arguments, i.e. "", represents that the agent is indifferent to all appointments);
- Subscribe with the DF to be notified of any agents that provide the "allocate-appointments" service, and keep record of an agent that provides the "allocate-appointments" service when informed by the DF as such (see yellowPages DFSubscribeAgent example in JADE documentation).

Step 3

Create class **RequestAppointment** that extends the **Behaviour** class.

This behaviour is to be added to the **PatientAgent** class.

Define action() method of RequestAppointment to do the following:

- Check that an agent is known that provides the "allocate-appointments" service.
- Check that this agent (i.e. the parent agent of this behaviour) has not already been allocated an appointment.
- Request an appointment from an agent that provides the "allocate-appointments" service.
- Receive response indicating whether an appointment has been allocated or not.
- In case an appointment is allocated in the response, modify the parent agent's state so that it knows what appointment it has been allocated.

Note: *The RequestAppointment behaviour should not request (or be allocated) more than one appointment.*

Step 4a

Create class **AllocateAppointment** that extends the **CyclicBehaviour** class.

This behaviour is to be added to the **HospitalAgent** class.

Define action() method of AllocateAppointment to do the following:

- Receive a request for an appointment.
- If there are any appointments available (i.e. not all appointments have been allocated), then pick one (any) and agree to the request notifying the requesting agent of the appointment allocated to it.
- If there are no appointments available, then refuse the request.
- In case an appointment is allocated, update the parent agent's state as to which agents have which appointments.

Step 4b

Define **takeDown()** method of **HospitalAgent** such that an instance of HospitalAgent prints out on separate lines which appointments have been allocated to which agents when the agent platform is shut down or when the agent is killed. For example, if a HospitalAgent instance named "hospital1" has 3 appointments, the first two of which have been allocated to "patient1" and "patient2" respectively, and the third has not been allocated to any agent, then the following should be printed:

```
hospital1: Appointment 1: patient1

hospital1: Appointment 2: patient2

hospital1: Appointment 3: null
```

Define **takeDown()** method of **PatientAgent** such that an instance of PatientAgent prints out which appointment it has been allocated, if any, when the agent platform is shut down or when the agent is killed. For example, a PatientAgent instance "patient1" should print out "patient1: Appointment 1" if it has the first appointment, or "patient1: null" if it has no appointment.

Note: *Lack of clear legible (succint!) output (i.e. messages sent and received between agents) as well as source files not containing clear legible code structure and comments will be penalised.*

*****

At this stage you should be able to run JADE with an (instance of a) HospitalAgent and a number of PatientAgents, such that the PatientAgents learn of the HospitalAgent (via the DF), make requests for appointments and are granted appointments.

For example, calling the command below (plus the classpaths) should start up JADE with the GUI, as well as a HospitalAgent instance named 'hospital1' that has three appointments available for allocation, and two PatientAgent instances with names and preferences as specified.

java jade.Boot -gui "hospital1:jadeCW.HospitalAgent(3)" "patient1:jadeCW.PatientAgent(1 2 - 3 -)" "patient2.jadeCW.PatientAgent()"

Note: **patient1:jadeCW.PatientAgent(1 2 - 3 -)** means that **patient1** prefers appointment 1 or 2 to appointment 3 (with appointments 1 and 2 equally preferred).

Note: *no two PatientAgents should get the same appointment, PatientAgents should only get appointments if they are available for allocation, and only appointments within the specified HospitalAgent input range should be allocated.*

You will now add to your definitions so that PatientAgent instances can swap appointments.

*****

Step 5

Create class **FindAppointmentOwner** that extends the **Behaviour** class.

This behaviour is to be added to the **PatientAgent** class.

Define action() method of FindAppointmentOwner to do the following:

- Select an appointment that is more preferred to the one allocated (if any) and query an agent that provides the "allocate-appointments" service as to which agent has the selected appointment.
- Receive response informing the agent as to which agent has the resource, if any, and modify the parent agent's state as such.

Note: *The FindAppointmentOwner behaviour should not continue making the same query infinitely many times.*

Step 6

Create class **RespondToQuery** that extends the **CyclicBehaviour** class.

This behaviour is to be added to the **HospitalAgent** class.

Define action() method of RespondToQuery to do the following:

- Receive a query regarding the owner of a certain appointment.
- Respond with the owner of that appointment, if the appointment is held by an agent, which could be a PatientAgent or the parent agent of this behaviour. Otherwise, notify the querying agent, for example, that the owner of the appointment is not known, or the appointment is not one known of.

*****

At this stage, when JADE is run with an (instance of a) HospitalAgent and a number of PatientAgents, each PatientAgent should be able to request an appointment from the HospitalAgent, be given any available one (if any are available), query the HospitalAgent as to the availability/owner of a more preferred appointment, and be told of the availability/owner of that appointment.

*****

Step 7

Create class **ProposeSwap** that extends the **Behaviour** class.

This behaviour is to be added to the **PatientAgent** class.

Define action() method of ProposeSwap to do the following:

- If an agent is known to have a more preferred appointment, then propose a swap of appointments.
- Receive response, modifying the parent agent's state as necessary.
- Inform the HospitalAgent that the two agents have swapped appointments, if a swap was made with an agent that is not a HospitalAgent.

Note: *The ProposeSwap behaviour should not continue making the same proposal infinitely many times.*

Step 8

Create class **RespondToProposal1** that extends the **CyclicBehaviour** class.

This behaviour is to be added to the **PatientAgent** class.

Define action() method of RespondToProposal1 to do the following:

- Receive a proposal to swap appointments.
- Respond with acceptance if the appointment to be received is at least as preferred as the appointment to be given away. Otherwise, if the appointment to be given away is not owned by this behaviour's parent agent or it is preferred to the one to be received, then respond with rejection.
- In case of acceptance, modify the parent agent's state as necessary and notify the HospitalAgent of the swap.

Step 9

Create class **RespondToProposal2** that extends the **CyclicBehaviour** class.

This behaviour is to be added to the **HospitalAgent** class.

Define action() method of RespondToProposal2 to do the following:

- Receive a proposal to change appointments.
- Respond with acceptance if the appointment asked for is available. Otherwise, if the appointment to be given away is not available, then respond with rejection, notifying the proposing agent of the owner of the requested resource.
- In case the booking is changed, update the parent agent's state as to which agents have which appointments.

Step 10

Create class **UpdateAppointments** that extends the **CyclicBehaviour** class.

This behaviour is to be added to the **HospitalAgent** class.

Define action() method of UpdateAppointments to do the following:

- Receive notifications from agents that have swapped appointments.
- Update the parent agent's state as to which agents have which appointments, but only when certain that two agents have indeed swapped appointments.

*****

At this stage, when JADE is run with an (instance of a) HospitalAgent and a number of PatientAgents, the PatientAgents should be able request/obtain appointments from the HospitalAgent and swap appointments between one another where a swap makes one of the agents more happy and none of the agents less happy. An agent should not cease and settle with its allocated appointment if any such swap is possible. *It is acceptable to focus on two agents' swaps*.

*****

Submit the zipped directory 'jadeCW' containing all your source files.

*****