UTRECHT UNIVERSITY

"Scheduling & Appointments"

Multi Agent Systems,

All questions about this exercise should be emailed to Bas Testerink (b.j.g.testerink@students.uu.nl).

1 INTRODUCTION

For the Jade exercise we are continuing in the spirit of the demo application. The goal is that some agents provide scheduling opportunities for various services (dentist, hairdresser, etc) and other agents want to make appointments.

1.1 SERVICE PROVIDERS

1. They have a service that they provide, which they register at the yellow page service.
2. They have a agenda on which appointments are made.
3. Optionally: price of the service.
4. A service provider cannot have two appointments at the same time.

1.2 CLIENTS

1. They have a series of services that they want to make an appointment with.
2. They find service providers through the yellow pages.
3. They have a agenda on which appointments are made.
4. Optionally: budget and preferred time slots.
5. A client cannot have two appointments at the same time.

1.3 Running process of the MAS

1. Service agents are initialized.
2. Client agents are injected at regular intervals.
3. A client immediately starts looking for the services it wants to make an appointment with and initiates contact with service providers.

1.4 Concrete exercise

1. You have to implement the service and client agent in Jade.
2. Upon initialization of the MAS the service agents have as an argument the service that they provide. The clients have as an argument a list of services they need.
3. After each new appointment an agent outputs to the console the appointment.
4. You must make an initialization agent that uses a TickerBehavior to create clients in the MAS at 3 second intervals. The setup of the initialization agent creates the service provider agents.
5. You must design a simple protocol. For instance: A client contacts a service provider for a free appointment time slot. Then the provider returns the request with a proposal. If the client is free at that time it will accept, otherwise it will request a different time slot. Etc. This protocol must be implemented with behaviors.

1.5 Example scenario

(note that due to distributed computing, events may be shuffled and hence the outcomes may differ)

1. The system starts and the initialization is created.
2. In its setup, it creates two service providers with the arguments "dentist" and "hairdresser" respectively.
3. Also in the setup a TickerBehavior is created and enqueued so that client agents can be spawned.
4. The service providers register themselves at the DF agent of Jade.
5. After 3 seconds the initialization agents creates a client agent A with the argument ["dentist","hairdresser"]
6. Agent A requests the list of agents that provide dentist services from the DF.
7. Agent A requests the list of agents that provide hairdresser services from the DF.
8. Agent A contacts the first result of the dentist list and requests a time slot.

9. The dentist agent replies with timeslot 0.
10. Agent A contacts the first result of the hairdresser list and requests a time slot.
11. The hairdresser agent replies with timeslot 0.
12. Agent A accepts the dentist offer.
13. Agent A requests another time slot from the hairdresser.
14. The dentist makes the appointment, outputs "[dentist] Appointment: Agent A, time: 0" and sends a confirmation.
15. The dentist agent replies with timeslot 1.
16. Agent A makes the appointment, outputs "[A] Appointment: dentist, time: 0".
17. Agent A accepts the hairdresser offer.
18. The hairdresser makes the appointment, outputs "[hairdresser] Appointment: Agent A, time: 1" and sends a confirmation.
19. Agent A makes the appointment, outputs "[A] Appointment: hairdresser, time: 1".

1.6 Resources/tips

All you start with is Jade. Look at the demo files for example code (http://jade.tilab.com/dl.php?file=JADE-examples-4.3.3.zip). As a little head start; here's an implementation for the agenda that you can use:

```
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

public class Agenda {
    private Set<Integer> reservations = new HashSet<Integer>();
    private Map<Integer, String> descriptions = new HashMap<Integer, String>();

    /**
    * Add a reservation.
    * @param timeslot Time slot to set the appointment.
    * @param description Description of the reservation.
    * @return True iff there is not yet an appointment at the time slot.
    */
    public boolean addReservation(Integer timeslot, String description){
        if(reservations.add(timeslot)){
            descriptions.put(timeslot, description);
            return true;
        } else return false;
    }
    /**
    * Check whether a time slot is free.
    * @param timeslot Time slot to check.
    * @return True iff there is no reservation at the time slot.
    */
    public boolean isFree(Integer timeslot){
        return !reservations.contains(timeslot);
    }

    /**
    * Cancel a reservation.
    * @param timeslot The time slot to make available.
    * @return True iff an appointment was canceled as a result of this call.
    */
    public boolean cancelReservation(Integer timeslot){
        if(reservations.remove(timeslot)){
            descriptions.remove(timeslot);
            return true;
        } else return false;
    }

    public String toString(){
        StringBuffer r = new StringBuffer();

        if(reservations.isEmpty()) r.append("no reservations");
        else for(Integer timeslot : reservations)
            r.append(timeslot).append(":").append(descriptions.get(timeslot)).append("\r\n

        return r.toString();
    }
}
```