Magdalena Osiewicz, IO, gr 2.

Rozproszona sztuczna inteligencja.

Sprawozdanie nr 10.

Temat: Yellow Pages.

W ramach zajęć zrealizowałam następujące kroki:

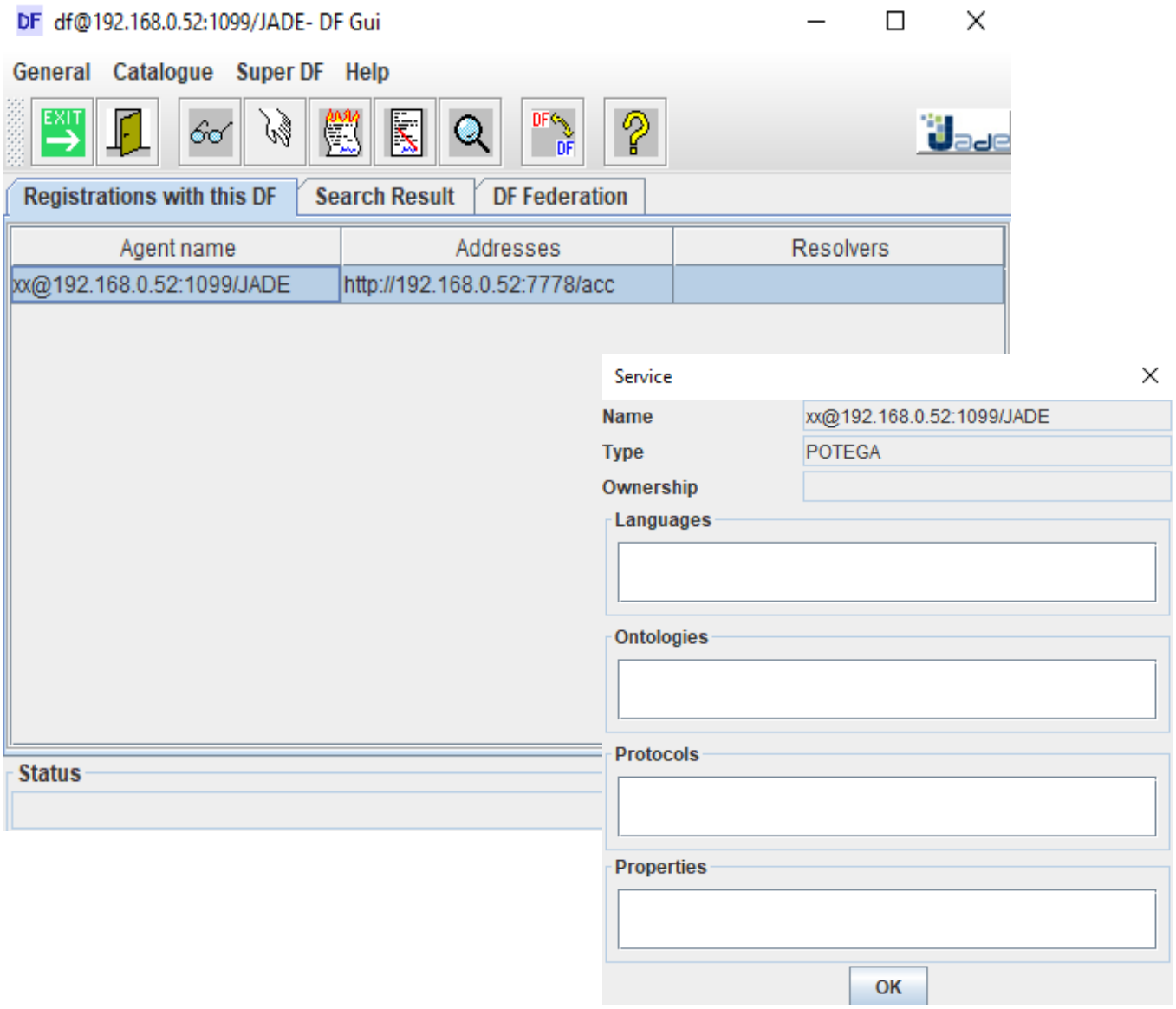
1. Utworzenie klasę agenta o nazwie wykonawca. Agent ten:

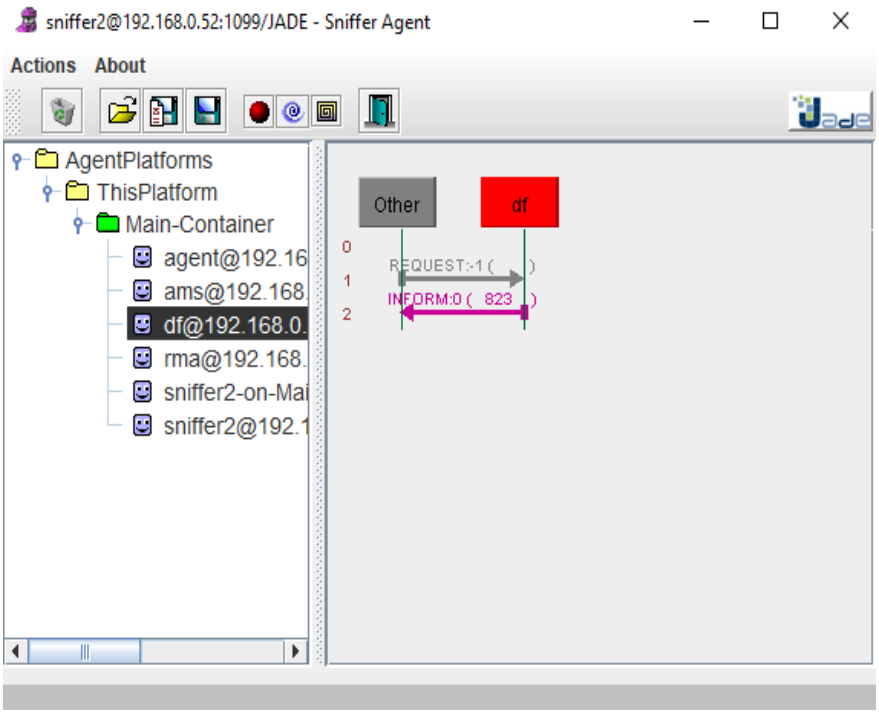
a). Na początku swojego istnienia rejestruje się w serwisie żółtych stron podając typ serwisu „POTEGA” (serwis o dowolnej nazwie).
b). Przed usunięciem wyrejestrowuje się z serwisu żółtych stron.

```
    //Rejestracja usługi:
    // 1. Przygotowanie opisu agenta:
    DFAgentDescription dfd=new DFAgentDescription();
    dfd.setName(getAID());
    ServiceDescription sd=new ServiceDescription();
    sd.setType("POTEGA");
    sd.setName(getName());
    dfd.addServices(sd);
    //2. Rejestracja w serwisie:
    try{
        DFService.register( a: this,dfd);
    }catch (FIPAException fe){
        fe.printStackTrace();
    }
}
protected void takeDown() {
    System.out.println("zaraz sie usune!");
    //3. Wyrejestrowanie z serwisu
    try{
        DFService.deregister( a: this);
    }catch (FIPAException fe){
        fe.printStackTrace();
    }
}
```

Fragment kodu przedstawia dwa kroki niezbędne do rejestracji w serwisie żółtych stron oraz wyrejestrowanie z serwisu przed usunięciem. Agent chcąc się zarejestrować musi podać swój własny AID. Koljenym ważnym etapem jest podanie typu oraz nazwy usługi. Parametry te zostału ustawione, dzięki metom *set* klasy ServiceDescription. Jest to uproszczenie standardowych procedur serwisu żółtych stron które dostarcza szereg statycznych metod, które pozwalają na korzystanie z serwisu bez kłopotliwego przygotywania wiadomości.

Konieczne jest również też wychwycenie możliwych wyjątków takich jak zduplikowane wpisy





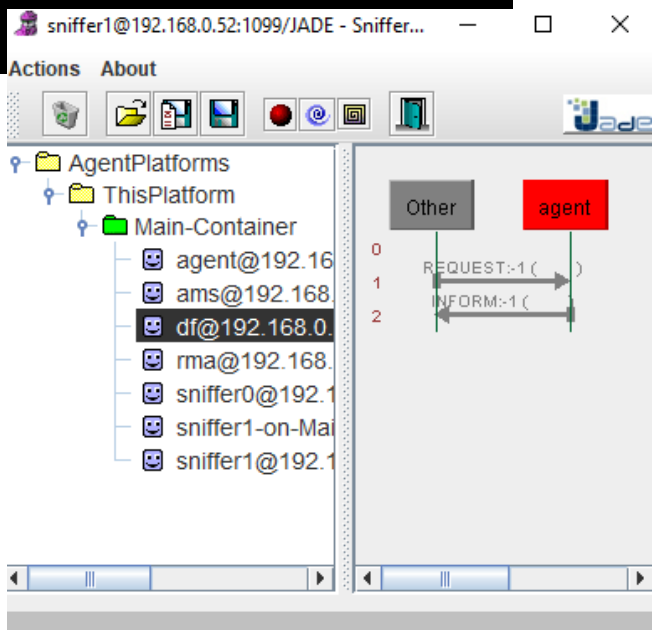
Po uruchomieni agenta i analizie sniffera możemy zaobserwować, że agent komunikuje się z innym agentem o nazwie df@ 192.168.0.52:1099/ JADE. Jest to wyspecjalizowany agent Directory Facilitor, który prowadzi serwis żółtych stron. Zatem korzystanie z tego serwisu jest równoznaczne z komunikacją między utworzonym agentem a agestem DF co widać na załączonym obok zrzucie.

2. Modyfikacja klasy agenta o nazwie Wykonawca: jeśli agent ten otrzyma wiadomość typu REQUEST to wykonuje:

a). Parsuje liczbę podaną w treści wiadomości,
b). Oblicza wynik podnosząc do drugiej potęgi otrzymaną liczbę
c). Odpowiada na wiadomość przesyłając wynik w treści wiadomości z typem INFORM

```
addBehaviour(new CyclicBehaviour() {
    public void action() {
        ACLMessage aclMessage = receive();
        if (aclMessage != null) {
            if (aclMessage.getPerformative() == ACLMessage.REQUEST) {
                String count_message = aclMessage.getContent();
                System.out.println("tresc wiadomosci: " + count_message);
                //Parsuje liczbe podana w treści wiadomości
                double message_number = Double.parseDouble(count_message);
                //Oblicza wynik podnosząc do drugiej potęgi otrzymaną liczbę
                double score = message_number*message_number;
                AID sender = aclMessage.getSender();
                System.out.println("Wyslanie tresci wiadomosci: " + Double.toString(score));
                ACLMessage response = new ACLMessage(ACLMessage.INFORM);
                response.addReceiver(sender);
                response.setContent(Double.toString(score));
                send(response);
            }
            if (aclMessage.getPerformative() == ACLMessage.NOT_UNDERSTOOD) {
                aclMessage.setPerformative(ACLMessage.NOT_UNDERSTOOD);
                aclMessage.setContent("nie_rozumiem");
            }
        }
        else {
            block();
        }
    }
} );
```

```
INFO: MTP addresses:
http://192.168.0.52:7778/acc
cze 19, 2018 5:57:26 PM jade.core.AgentContainerImpl joinPlatform
INFO: ----------------------------------------
Agent container Main-Container@192.168.0.52 is ready.
--------------------------------------------
start!
The content of the message: 13
Sent message: 169.0
The content of the message: 22
Sent message: 484.0
!!!
```



3. Utworzenie klasy agenta o nazwie Zlecajacy. Agent ten powinien:

a). Losować liczbę całkowitą z przedziału od 0 do 100.
b). Co 20 sekund dowiadywać się w serwisie żółtych stron, którzy agenci oferują serwis typu „POTEGA”.
c). Jeśli zwrócono jakiegokolwiek agenta jako wynik przeszukiwania serwisu z punktu (b) to wykonuje punkty (d), (e) i (f).
d). Wysyła do pierwszego agenta z tablicy wyników wiadomość typu REQUEST zawierającą wylosowaną liczbę.
e). Odbiera wiadomość typu INFORM i wypisuje jej treść na ekranie. f). Usuwa się.

```
private void recieve(Behaviour b) {
    MessageTemplate mt = MessageTemplate.MatchPerformative(ACLMessage.INFORM);
    ACLMessage msg1 = blockingReceive(mt);
    if(msg1!= null) {
        if(msg1.getPerformative() == ACLMessage.INFORM){
            System.out.println("Odebranie wiadomosci inform przez zlecajacego  " +  msg1.getContent());
            my.doDelete();
        }
    }
    else {
        b.block();
    }
}
```

W tym przypadku metoda *receive* wygląda nieco inaczej niż w poprzenich zadaniach, ponieważ jako argument przyjmuje obiekt Behaviour. Zachowanie to odpowiada wyszukiwaniu wylosowanej usługi potęgowania co 20 sekund. W tym celu wykorzystano metodę *search* na obiekcie klasy DFService. Metoda ta zwraca listę wszytskich opisów pasujących do tego podanego w szablonie.

```
addBehaviour(new TickerBehaviour( a: this,  period: 2000) {
    protected void onTick() {
        System.out.println("Sprawdzanie uslugi POTEGA");

        DFAgentDescription dfd = new DFAgentDescription();
        ServiceDescription sd = new ServiceDescription();
        sd.setType( "POTEGA" );
        dfd.addServices(sd);

        SearchConstraints ALL = new SearchConstraints();
        ALL.setMaxResults(new Long( value: -1));
        try
        {
            DFAgentDescription[] result = DFService.search(myAgent, dfd, ALL);
            System.out.println("Liczba uslug:    " + result.length);
            System.out.println(" ");
            if(result.length>0) {
                send(result[0].getName());
                recieve( b: this);
            }
        }
        catch (FIPAException fe) {
            System.out.println("ERROR"); }
    }
});
```

```
Wysolowana liczba:   32
Sprawdzanie uslugi POTEGA
Liczba uslug:    0

Sprawdzanie uslugi POTEGA
Liczba uslug:    0

Sprawdzanie uslugi POTEGA
Liczba uslug:    0

Sprawdzanie uslugi POTEGA
Liczba uslug:    0

Sprawdzanie uslugi POTEGA
Liczba uslug:    0

start!
The content of the message: 12
Sent message: 144.0
The content of the message: 12
Sent message: 144.0
The content of the message: 3
Sent message: 9.0
The content of the message: 5
Sent message: 25.0
Sprawdzanie uslugi POTEGA
Liczba uslug:    1

Wysylanie wiadomoĹ?ci REQUEST przez zlecajacego
The content of the message: 32
Sent message: 1024.0
Odebranie wiadomosci inform przez zlecajacego  1024.0
Zakonczenie dzialania przez zlecajacego
```