Proiect Final Programare Orientata pe Agenti

Pîrcalaboiu Andrei-Marius 242-2

Toderuțiu Radu Ioan 242-1

1. Tema si descrierea proiectului:

Folosind Platforma Android dezvoltati un sistem multiagent.

Sa se creeze doua containere, primul (principal) va contine:

a) un agent DisplayAgent in containerul principal care aseaza informatiile primite de la alti agenti intr-o fereastra.

b) un agent mobil ReadFileMobileAgent in containerul principal care citeste o lista de preturi de produse dintr-un fsier si le transmite apoi la agentul DisplayAgent. Containerul doi (secundar) sa contina un agent FilePathInfoAgent care stie din ce fisier trebuie citite informatiile cu lista de produse.

Cerinte: Sa se implementeze urmatorul comportament: Agentul Read-File MobileAgent se va muta in containerul secundar si va cere prin intermediul mesajelor agentului FilePathInfoAgent locatia in care se afla fisierul cu lista de produse si va trimite aceasta lista la agentul DisplayAgent.

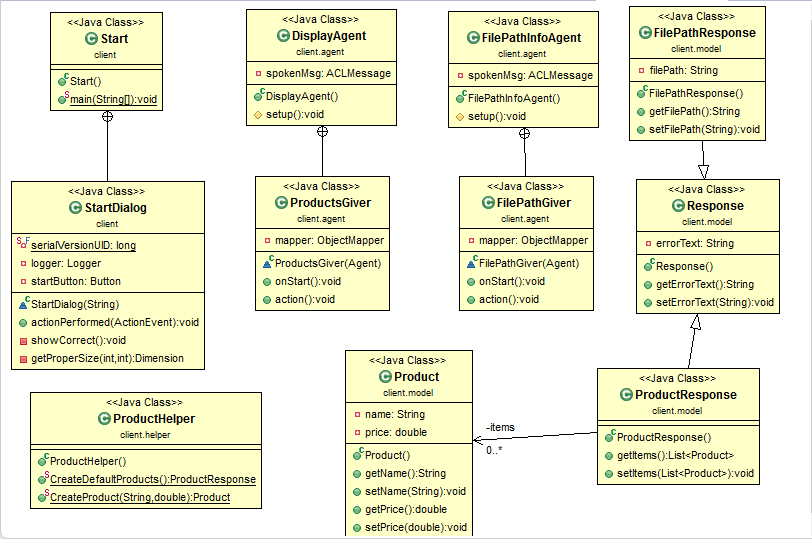
1. Cerinte technice

Proiectul este realizat în mediul de programare Eclipse si Android Studio, utilizând platform JADE. Pentru a rula proiectul, este necesară instalarea platformei JADE și adaugarea librariilor necesare: jade.jar, http.jar, iiop.jar, jadeTools.jar .

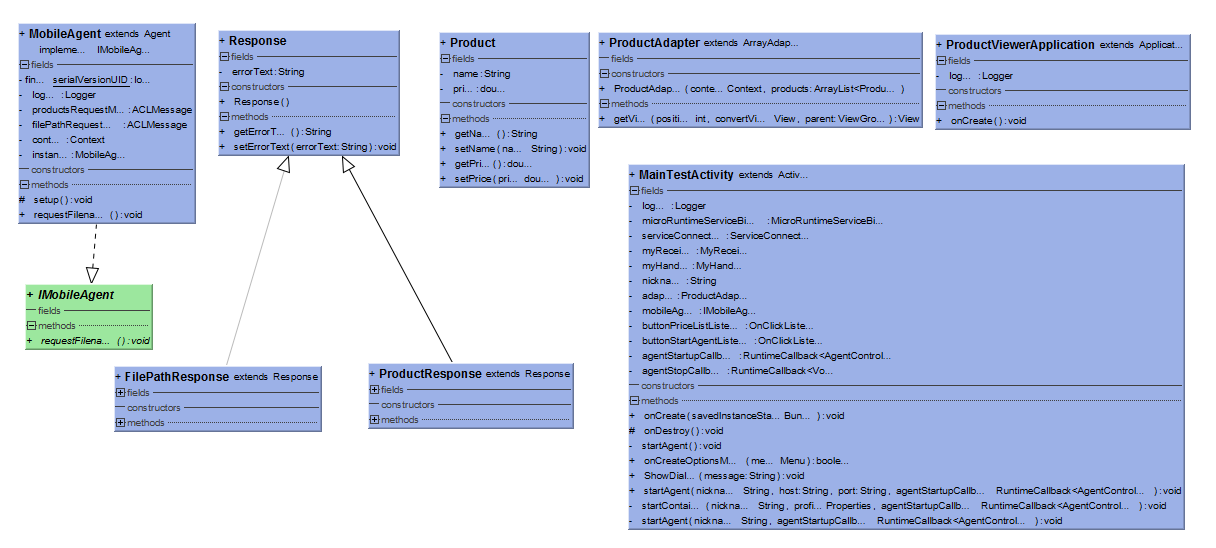
1. Arhitectura interna a programului

Programul este impartit in doua:

* Proiectul Java realizat in mediul Eclipse care are ca si scop principal pornirea platformei JADE si pornirea agentilor DisplayAgent si FilePathInfoAgent.
* Proiectul Android realizat in mediul Android Studio care are scop realizarea unei aplicatii ce se va conecta la o platforma JADE si va trimite prin intermediul agentului creeat o cerere catre alti agenti din containere diferite si va primi de la acestia un raspuns pe care il va procesa si il va afisa pe ecran.
* Descrierea Claseselor din fiecare proiect:
  + In continuare se vor descrie clasele din proiectul Eclipse folosind diagram uml:

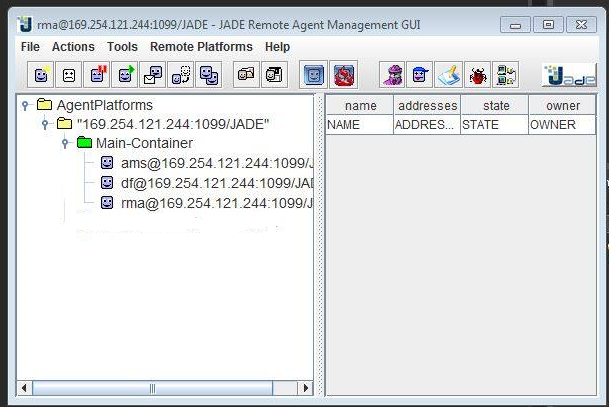


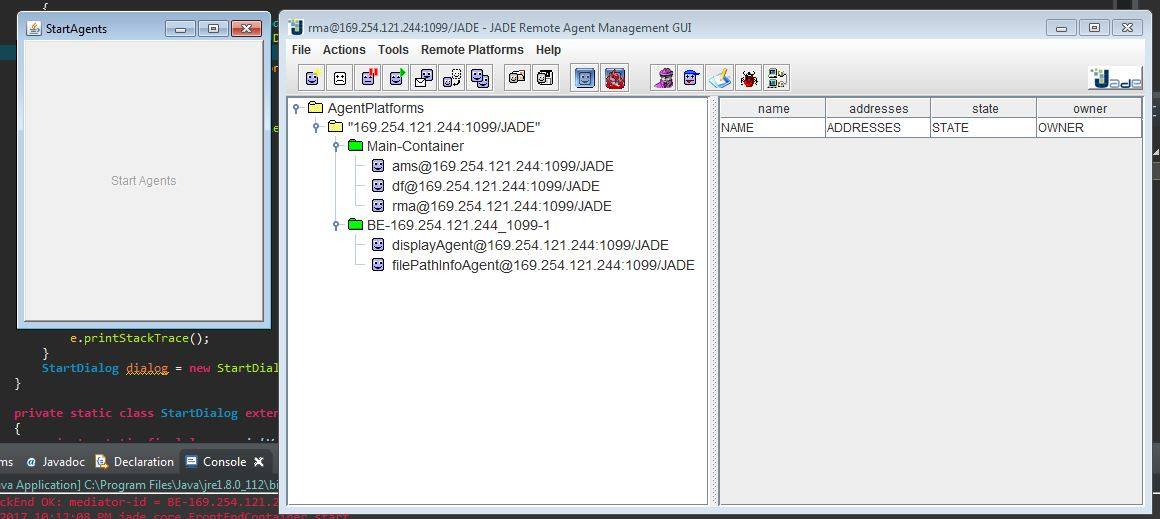
* + In continuare se vor descrie clasele din proiectul Android prin diagrama uml:



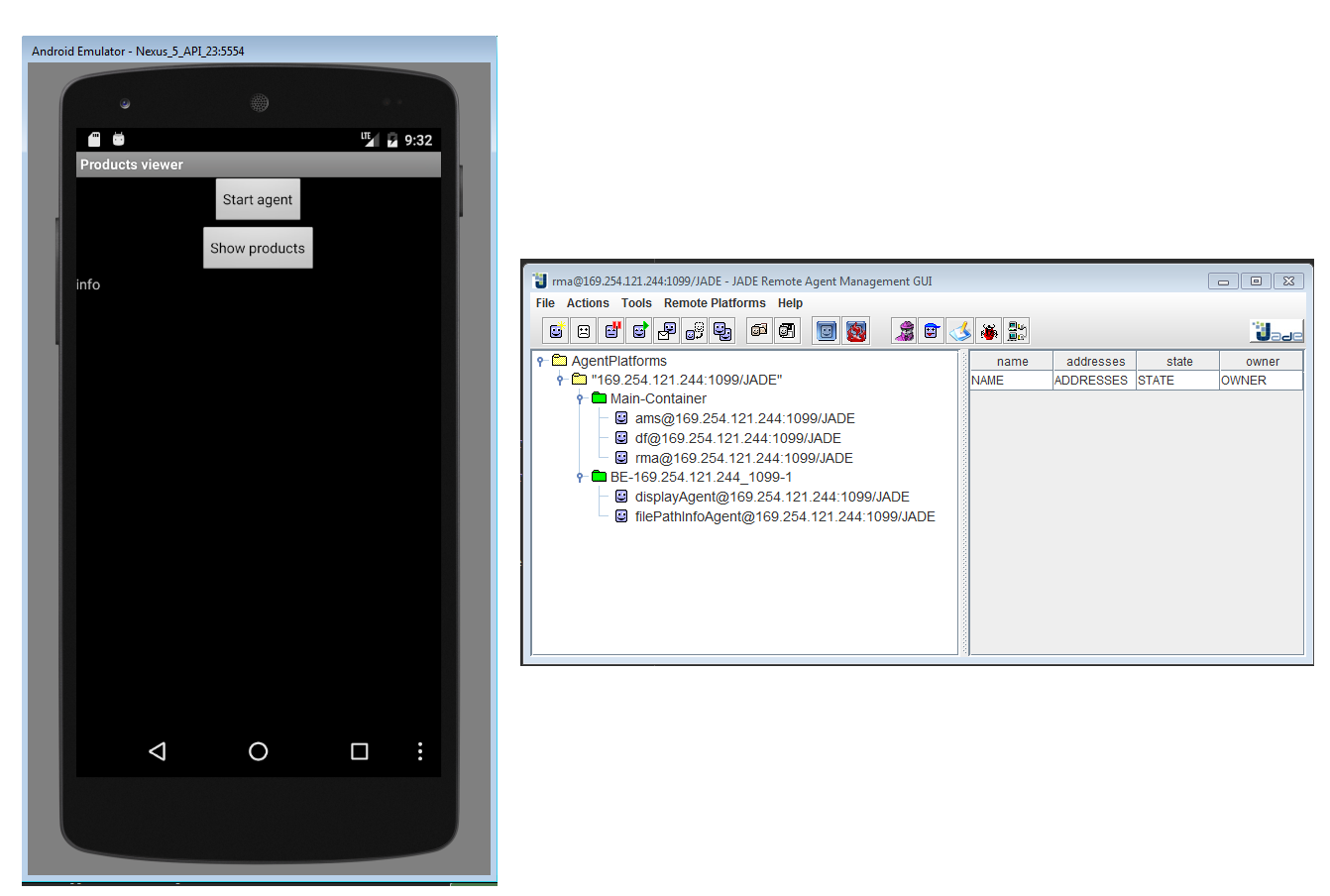
1. Modul de utilizare al programului

Pentru proiectul java care porneste platforma jade si contine agentii care trimit date catre aplicatia de Android. Au fost configurate doua confiuratii de run. Prima configuratie “jade start” va porni platforma jade care va contine MainContainer-ul si agentii acestuia (ams,df si rma).

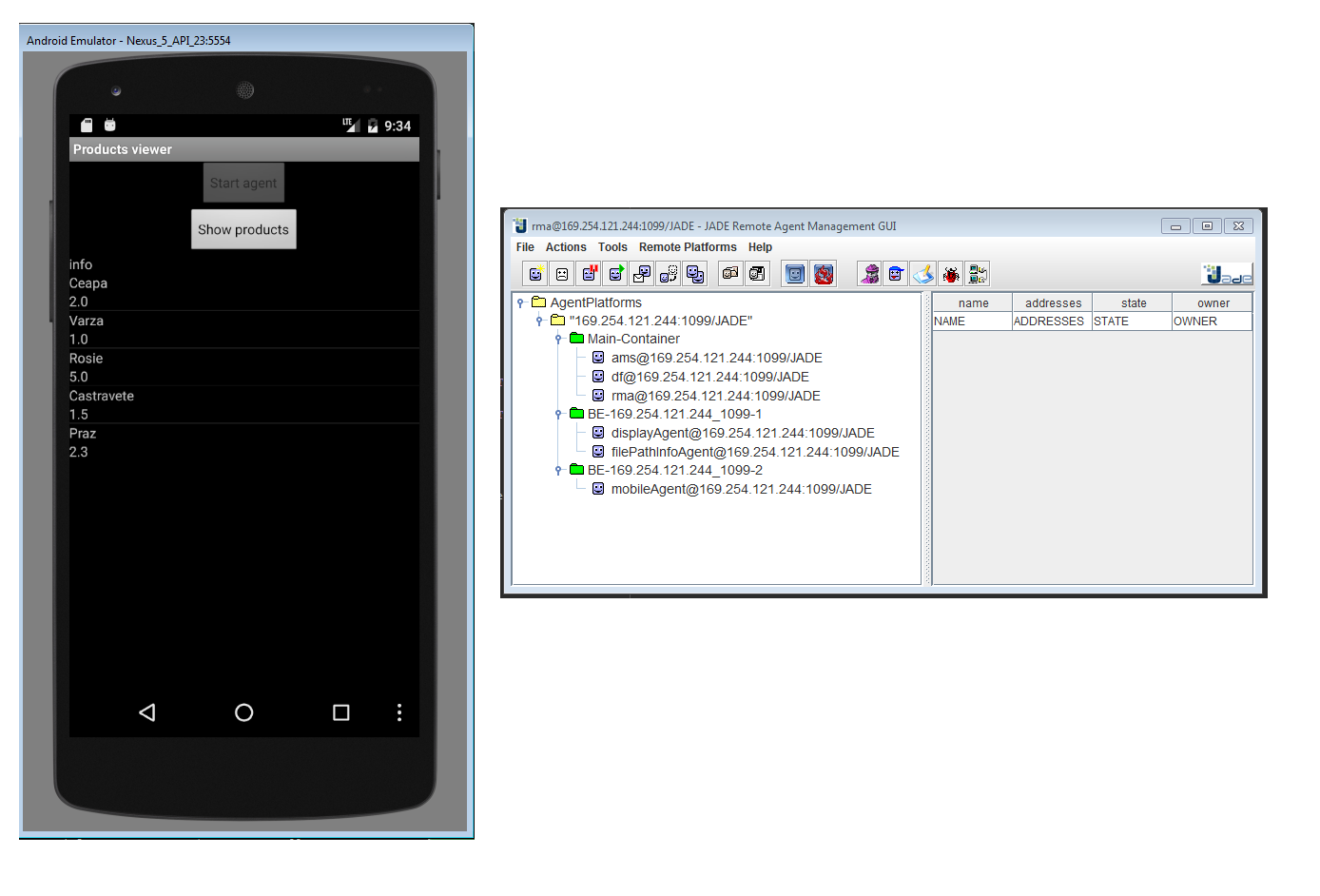


Dupa ce platforma a fost pornita se poate rula a doua configuratie de run a proiectului “start gui” , configuratie in care se va porini o mini interfata grafica care are doar un buton (Start Agents). Cand acest buton va fi actionat in platforma jade va aparea un nou container cu doi agenti filePathIfoAgent respectiv DisplayAgent. Primul agent contine informatia despre locatia unde se va gasi pe hard disk lista cu preturi care va fi ulterior la cere afisata in aplicatia android. DisplayAgent trimite catre agentul pornit de aplicatia android lista de produse serializata in fosrmat json.

Pentru ca aplicatia de android sa poata fi rulata este nevoie sa se instaleze sdk-ul corespunzator si o masina virtuala de android pe care sa ruleze. Se va rula aplicatia de android iar pe ecran vor aparea doua butoane. Primul buton, “Start Agent” care va conecta aplicatia de android la platforma jade prin intermediul unui agent jade care porneste intr-un container separate. Agentul se numeste mobile agent iar functia acestuia este de a comunica cu agentii din primul container. Al doilea buton “Show Products” va afisa pe ecran lista de produse preluata de locatia data de catre agentul filepathinfo.



In aceasta imagine este pornita aplicatia android, dar agentul nu a fost pornit inca. Dupa cum se poate vedea acesta nu este vizibil in platforma.



In aceasta imagine se poate observa ca agentul a fost pornit, butonul fiind dezactivat, nu mai avem optiunea de a porni din nou agentul. Butonul “Show Products” a fost apasat, iar preturile au aparut pe ecranul telefonului.

1. Identificarea Agentilor. Rolurile lor

Dupa cum a fost mentionat si mai sus, proiectul este impartit intr-un proiect java scris in mediul Eclipse care are ca scop initial pornirea platformei JADE si pornirea a doi agenti intr-un container separat de cel principal. Agentii sunt FilePathInfoAgent si DisplayAgent.

FilePathInfoAgent contine un comportament de tip CyclicBehaviour numit FilePathGiver in care se va seta locatia de pe hard disk de la care se va citi lista de preturi a produselor. Acest comportament se poate este exemplificat mai jos:

**public** **void** action()

{

ACLMessage msg = myAgent.receive();

**if**(msg != **null** && msg.getPerformative() == ACLMessage.***REQUEST*** && msg.getProtocol().equals(InteractionProtocol.***FIPA\_REQUEST***))

{

**try**

{

FilePathResponse response = **new** FilePathResponse();

response.setFilePath("C:\\Users\\Radu2T\\Downloads\\produse.json");

String json = mapper.writeValueAsString(response);

spokenMsg = **new** ACLMessage(ACLMessage.***ACCEPT\_PROPOSAL***);

spokenMsg.setContent(json);

spokenMsg.addReceiver(msg.getSender());

send(spokenMsg);

}

**catch** (Exception e)

{

Logger.*println*(e.toString());

e.printStackTrace();

}

}

**...**

DisplayAgent contine la fel ca si FilePathInfoAgent, cel mentionat mai sus, un CyclicBehaviour numit ProductsGiver acesta valideaza citirea de pe hardisk , verificand daca mesajul primit este acelasi ca si cel pe care agentul doreste sa il primeasca. Daca se trece de aceasta conditie, se va citi de pe hard disk si mesajul va fi trimis.

**public** **void** action()

{

ACLMessage msg = myAgent.receive();

**if** (msg != **null** && msg.getPerformative() == ACLMessage.***REQUEST*** && msg.getProtocol().equals(InteractionProtocol.***FIPA\_REQUEST***))

{

**if** (msg.getContent().equals("C:\\Users\\Radu2T\\Downloads\\produse.json"))

{

**try**

{

String products = **new** String(Files.*readAllBytes*(Paths.*get*(msg.getContent())));

spokenMsg = **new** ACLMessage(ACLMessage.***ACCEPT\_PROPOSAL***);

spokenMsg.setContent(products);

spokenMsg.addReceiver(msg.getSender());

send(spokenMsg);

}

**catch** (Exception e)

{

Logger.*println*(e.toString());

e.printStackTrace();

}

} ...

Pentru aplicatia de Android se poate identifica agentul MobileAgent care contintine doua comportamente de tip CyclicBehaviour. Primul comportament numit FilePathReciever va primi locatia de la care se vor citi datele de pe hard disk.

public void action() {  
 ACLMessage msg = myAgent.receive();  
 if (msg != null && msg.getPerformative() == ACLMessage.*ACCEPT\_PROPOSAL*) {  
 try {  
 FilePathResponse response = mapper.readValue(msg.getContent(),FilePathResponse.class);  
 addBehaviour(new MobileAgent.ProductsRequester(instance,response.getFilePath()));  
 } catch (Exception e) {  
 Logger.*println*(e.toString());  
 e.printStackTrace();  
 }  
 } else if (msg == null) {  
 block();  
 }  
}

Al doilea comportament ProductsReciever va primi lista de produse si preturile acestora in format json.

public void action() {  
 ACLMessage msg = myAgent.receive();  
 if (msg != null && msg.getPerformative() == ACLMessage.*ACCEPT\_PROPOSAL*) {  
 try {  
 Intent broadcast = new Intent();  
 broadcast.setAction("jade.demo.chat.PRODUCT\_LIST");  
 broadcast.putExtra("productResponse",msg.getContent());  
 context.sendBroadcast(broadcast);  
 } catch (Exception e) {  
 Logger.*println*(e.toString());  
 e.printStackTrace();  
 }  
 } else if (msg == null) {  
 block() } }}

Celelate doua comportamente sunt de tip OneShotBehaviour. Primul comportament, FilenameRequester va cere de la agentul filePathInfoAgent locatia de la care trebuie sa citeasca lista de preturi.

public void action() {  
 filePathRequestMsg.clearAllReceiver();  
 AID aid = getAID("filePathInfoAgent");  
 filePathRequestMsg.addReceiver(aid);  
 filePathRequestMsg.setContent(sentence);  
 send(filePathRequestMsg);  
}

Al doilea comportament de tip OneShotBehaviour numit ProductsRequester va cere de la agentul displayAgent lista de produse si preturile acestora in format json pentru a le putea afisa pe ecran.

public void action() {  
 productsRequestMsg.clearAllReceiver();  
 AID aid = getAID("displayAgent");  
 productsRequestMsg.addReceiver(aid);  
 productsRequestMsg.setContent(sentence);  
 send(productsRequestMsg);  
}