Multi Agent Programming for Managing Virtual Communities

Ahmad ALIBRAHIM, Mohamed ZEMROUN, Adnan WALAYAT

March 12, 2019

1 Introduction

A community is a group of people that have similar interests. Members of a community can participate, communicate, and share information with each others. The topic of this project is to implement a smart city which is composed of virtual communities. Citizens of the city can participate and communicate through these communities. The city is equipped with community servers where each server has a specific topic. In our project, we implemented one server with the topic "Sports". This project uses the multi-agent-oriented platform JaCaMo in order to implement a multi-agent based virtual community management platform.

2 Structure

2.1 Java

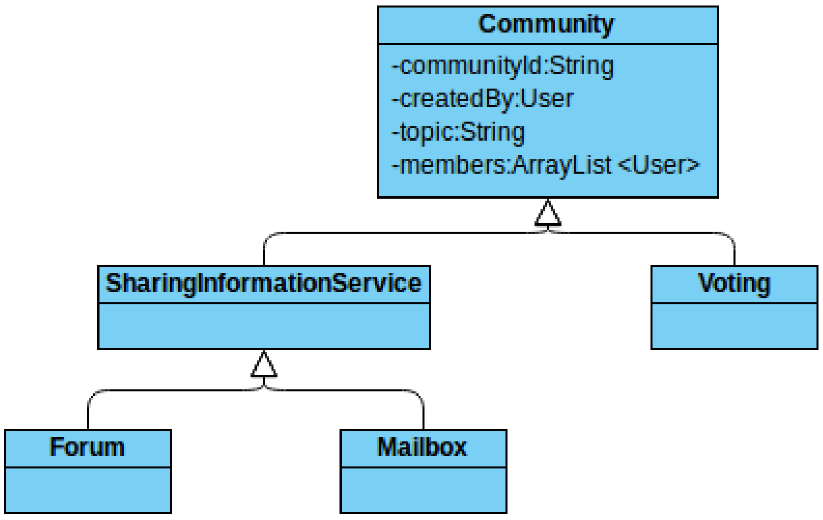


Figure 1: Java Design

In this project we have 2 community types (Voting and Sharing Information Service) where every one of these communities inherits from the Community class,the latter inherited from the artifact class and contain the id,owner,members and topic so that every community that inherits from it have also those attributes.there is also 2 other type of communities (Mailbox and Forum) that inherits from Sharing Information Service.You can see the model in figure below.

2.2 JASON

To implement autonomous smart agents, we used JASON. In our project, we have 3 kinds of agent with 4 different behaviors. The first type of agent is the server agent, this agent receive requests from other agents and achieve their request by dealing with the server. The other three types are agents that corresponds to users, these agents are community assistants, they support the users in the management of their life through achieving their requests. Each type of agents has his own artifact that he's focusing on.

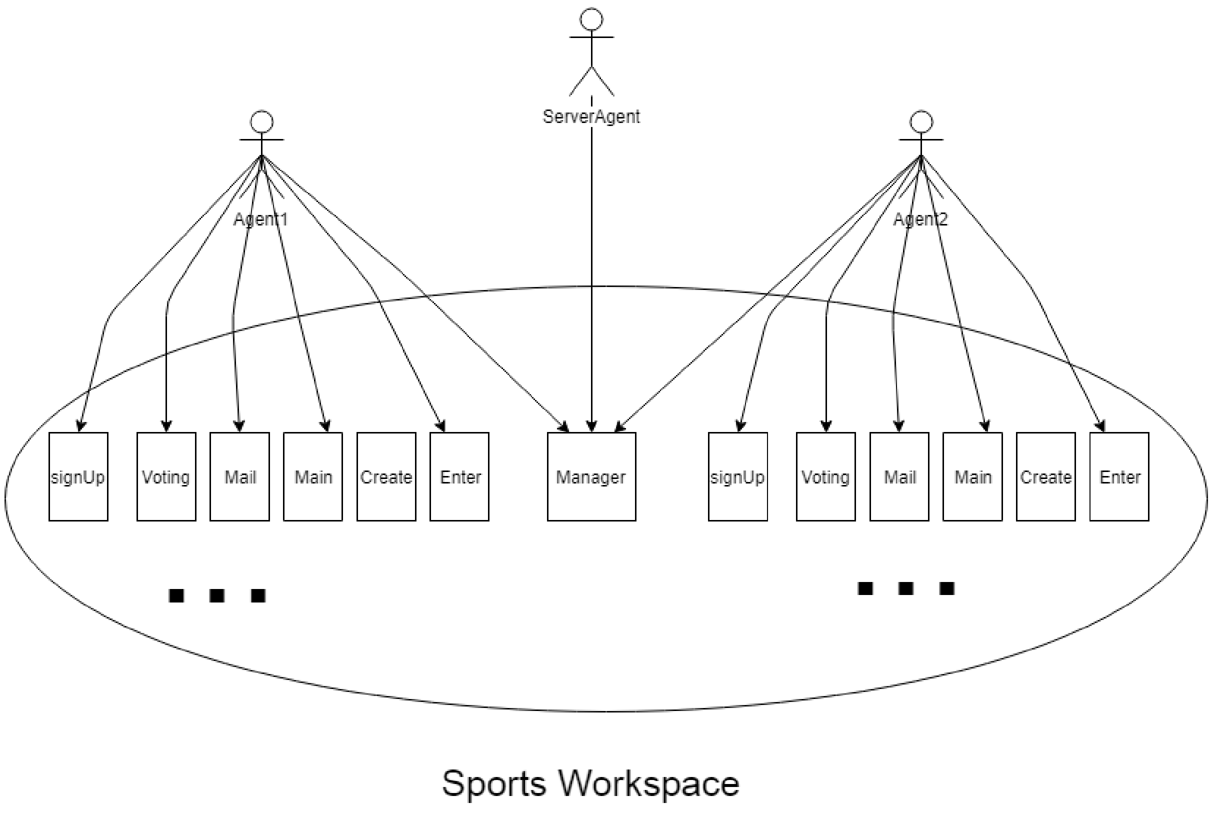


Figure 2: JaCaMo Structure

2.2.1 Server Agent

This agent's main and only role is to initialize the **Manager** artifact. By initializing this artifact, it is initializing the arraylist of communities and the arraylist of users in order for the other agents to act on these arraylists (add, delete, modify).

2.2.2 Hello Agent

As all other agents, this agent can create, delete, join, and leave a community. What's different between this agent and the others is the behavior. As the name of the agent says, **HelloAgent** is an agent responsible for sending hello messages to members of a mailbox. Whenever a mailbox is created, this agent will join it. When the number of participants become greater than N, **HelloAgent** will send a broadcast "Hello" message to all the members of the mailbox, and after that it leaves the mailbox

community.
Technically speaking, when a user create a mailbox, the artifact **ManagerArtifact** will send a signal ***(mailboxCreated(MailboxId))*** with the community id to all agents, saying that a mailbox is created. Only **HelloAgent** agents will consider this signal. When the **HelloAgent** receives this signal, it will add the goal ***(addToCommunity(MyName,MailboxId)***, this goal will add the user to the community created. After that, whenever a member joins the mailbox, the artifact **ManagerArtifact** will send a signal ***numberOfMembersIncreased(CommunityId,Number)*** saying that the number of members of mailbox "**CommunityId**" is "**Number**". When the agent receives this signal, it checks the "**Number**" and compare it to another number **N** initiated by the agent, deppending on the comparison, the agent decides whether to send the broadcast or not. If the number is greater than "**N**", it add 2 goals, the first one is ***sendMessage(MailboxId,MyName,"All","Hello")*** which is the goal of sending a broadcast to all the members of the community named "**Mailbox**" with message "Hello". The second goal is ***!leaveCommunity(MailboxId,MyName)*** which leaves the community "**MailboxId**".

2.2.3 Versatile-egocentric Agent

The main role of this agent is to leave or delete a community if the user change his interests and it those not much with the topic of the community, this agent can also handle as all the other agents to create,join,enter,delete or show a community.
Let take the example of the voting community to create this community the agent focus on the create voting community artifact when the user fill in the information the artifact send a signal to the agent with the information needed to create a new voting community , "**Versatile-egocentric Agent**" focus on the community manager artifact and lunch the goal ***!createVoting(CommunityId,CreatedBy,Topic ,QuestionsIndices)*** this will execute the ***createVoting(CommunityId,CreatedBy,Topic,QuestionsIndices)*** operation defined on the artifact .
But the main role as we said before is to leave communities with topic does not match to our interests so when the user his interests using the "**ChangeInterestsInterface**" the "**ChangeInterests Artifact**" send a the signal ***updateIntrests(UserName,Intrests)*** to this agent ,when he receive it he focus on the **ManagerArtifact** and start the operation **updateIntrests** defined on the artifact this operation will call the method ***leaveCommunity(UserName)*** which leave all the communities with a topic diffrent of the user interests.

2.2.4 CopyAndPaste Agent

2.3 Artifacts

2.3.1 Common Artifacts

Manager Artifact:
All the agents are focusing on this artifact. In our project, this artifact is like a database for storing the communities and the users. Whenever a user wants to createCommunity, leaveCommunity, addHimslef to a community..., his agent must access this artifact and make the appropriate operation.

1. Operations:

    - sendMessage: send message inside a mailbox
    - enterCommunity: to enter a community
    - leaveCommunity: to leave a community
    - createMailbox: to create a mailbox community
    - createVoing: to create a voting community
    - deleteCommunity: for the owner to delete his community
    - updateInterests: for the user to change his interests
    - addUser: when user sign up, add him to the list of users

MainInterfaceGUIArtifact:
The **MainInterfaceGUIArtifact** is the artifact how manage the main GUI in this interface we access to

the show communities ,createMailbox,createVoting,myCommunities,ownedCommunities and changeInterests GUIs so for each of this actions we have an internal operation in our artifact how send a signal to the agent to focus on the artifact of one of them.

SignUp Artifact:
The **SignUpGUIArtifact** is the artifact how manage the Sign Up GUI in this interface the user is sign up and chose his interests, in the artifact we have an internal operation ***signup*** that send a signal to the agent to add the user and we have an operation ***resultOfSigningUp*** which close the GUI if there is no Error or display a the message "User Already Exists".

Show Communities Artifact:
The **ShowCommunitiesGUIArtifact** is the artifact how manage the Show Communities GUI in this interface the user can see all existing communities and can join one of them, in the artifact we have an internal operation ***joinCommunity*** that send a signal to the agent to add the user in the communities members.

Owned Communities Artifact:
The **OwnedCommunitiesGUIArtifact** is the artifact how manage the Owned Communities GUI in this interface the user can see all the communities that he owned and can delete them, in the artifact we have an internal operation ***deleteCommunity*** that send a signal to the agent to delete the community with a specific id .

2.3.2 Mailbox Artifacts

Mailbox Artifacts are the artifacts that are related to mailbox. These artifacts are the following:

1. CreateMailbox: This artifact is a GUIArtifact, so when initializing this artifact, it's basically creating an instance of another java class **CreateMailboxInterface**. The reason of this design is to let the agent to have access to the interface through the artifact. This interface is to help the user to create a mailbox community, so by that he'll become the owner of this community.

    - Internal Operations:
      createMailbox(ActionEvent ev): This internal operation is linked to the action event when pressing the button createCommunityButton. This operation collect the information from the form of the gui class **CreateMailboxInterface** then create a java object mailbox and send a signal ***createMailbox(mailbox)*** with the mailbox object as a parameter, telling the agent to add this mailbox to the server.
    - Operations:
      startInterface(String name): This operation is called by the agent when the user wants to create a mailbox community.
      mailboxCreatedSuccesfully(boolean ok): After adding the community to the server, the agent will receive a feedback from the server telling him if the operation was successfull or not. Depending on the feedback, the agent will call this operation with the corresponding boolean to show a message dialog that informs the user about the result of the creation of the mailbox.

2. EnterMailbox: This artifact is a GUIArtifact, so when initializing this artifact, it's basically creating an instance of another java class **EnterMailboxInterface**. This interface is to help the user to act on a mailbox community. Acting on a mailbox is basically sending messages to members and reading messages sent by members.

    - Internal Operations:
      sendMessage(ActionEvent ev): This internal operation is linked to the action event when the user presses the button sendMessageButton. It will get the receiver and the message from the gui, and send them to the agent through a signal **sendMessage(ComunityId, From, To, Message)** which will be treated by the agent by adding this message to the database.

- Operations:
  startInterface(String name): This operation is called by the agent when the user wants to enter a mailbox community.

2.3.3 Voting Artifacts

Voting Artifacts are the artifacts that are related to voting. These artifacts are the following:

1. CreateVoting: This artifact is a GUIArtifact, so when initializing this artifact, it's basically creating an instance of another java class **CreateVotingInterface**.This interface is to help the user to create a Voting community, so by that he'll become the owner of this community.

   - Internal Operations:
     createVoting(ActionEvent ev): This internal operation is linked to the action event when pressing the button CreateCommunityButton. This operation collect the information from the form of the gui class **CreateVotingInterface** then create a java object Voting and send a signal ***createVoting(Voting)*** with the Voting object as a parameter, telling the agent to add this Voting to the server.
   - Operations:
     startVotingInterface(String name): This operation is called by the agent when the user wants to create a Voting community.
     votingCreatedSuccesfully(boolean ok): if the voting creation succeed the it display that the community had been added and then close the GUI else it display an error message.

2. EnterVoting: This artifact is a GUIArtifact, so when initializing this artifact, it's basically creating an instance of another java class **EnterVotingInterface**. This interface gives the possibility to the user to make a vote.

   - Internal Operations:
     next(ActionEvent ev): This internal operation is linked to the action event when pressing the button nextButton. This operation collect the information from the form of the GUI class **EnterVotingInterface** then send a signal ***addToMap*** with with different information, telling the agent to add this response to the list of the response.
     confirm(ActionEvent ev): This internal operation is linked to the action event when pressing the button confirmButton. This operation confirm the vote and send a signal to the agent with the to display the result of the vote.
   - Operations:
     startVoting(String name): This operation is called by the agent when the user wants to create a enter a voting community.

2.3.4 Forum Artifacts

2.4 Workspace

The workspace named "Sports" in our project is where the agents act. The list of sports in our project are: Football, Basketball, Tennis, and VolleyBall. These are the interests that the user can choose from. Also these are the available topics of the created commmunities.

3 Scenario

1. Scenario 1:

   - 3 users: Bob, Alice, and Eve.
   - All 3 users have "Football" as interest.
   - Bob has HelloAgent agent, while Alice and Eve have another type of agent (for example EgocentricAgent).

- Alice creates a mailbox with ”Football” topic.
- Eve joins the mailbox.

Let’s start: First we will have 3 user interfaces, each one corresponds to an agent. The users will write their name and selects there interests, then click signUp.

1/signUp.png

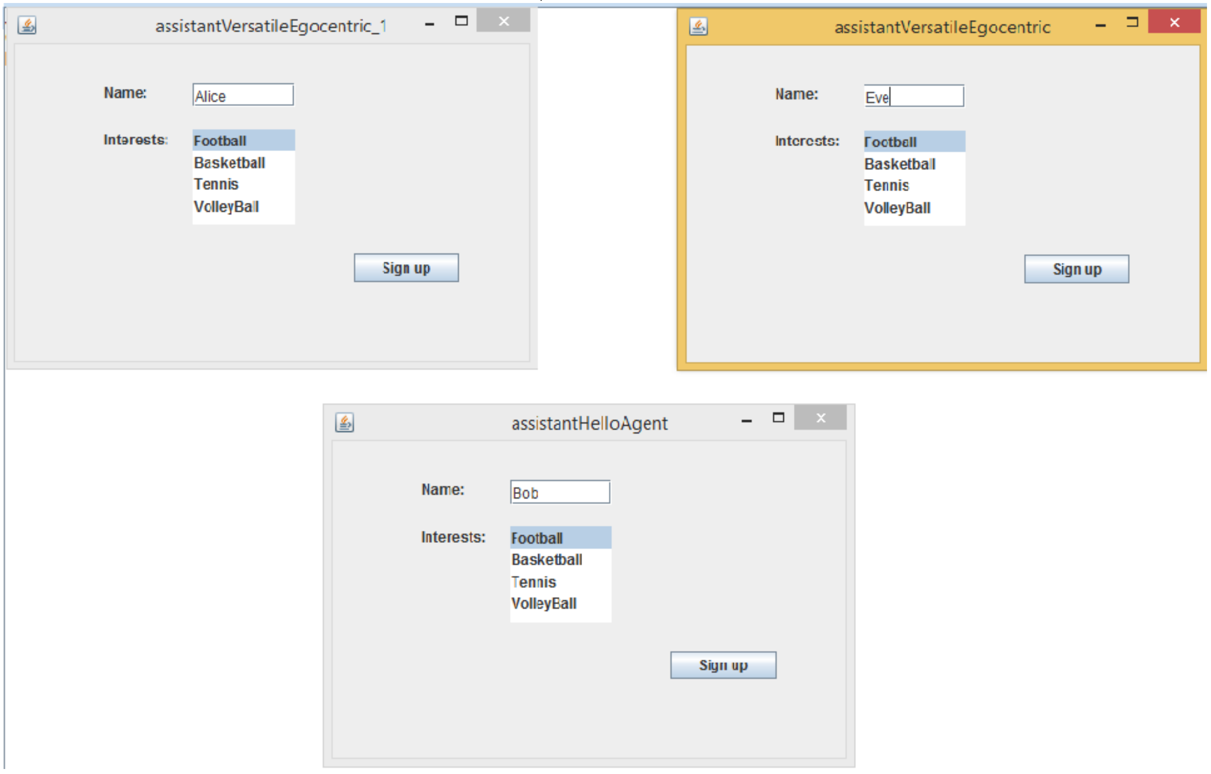


Figure 3: SignUp

The 3 users sign up and the main interface will appear. The main interface helps the users to do the following:

- Create community
- Show communities
- Enter community
- View owned communities
- Change interests

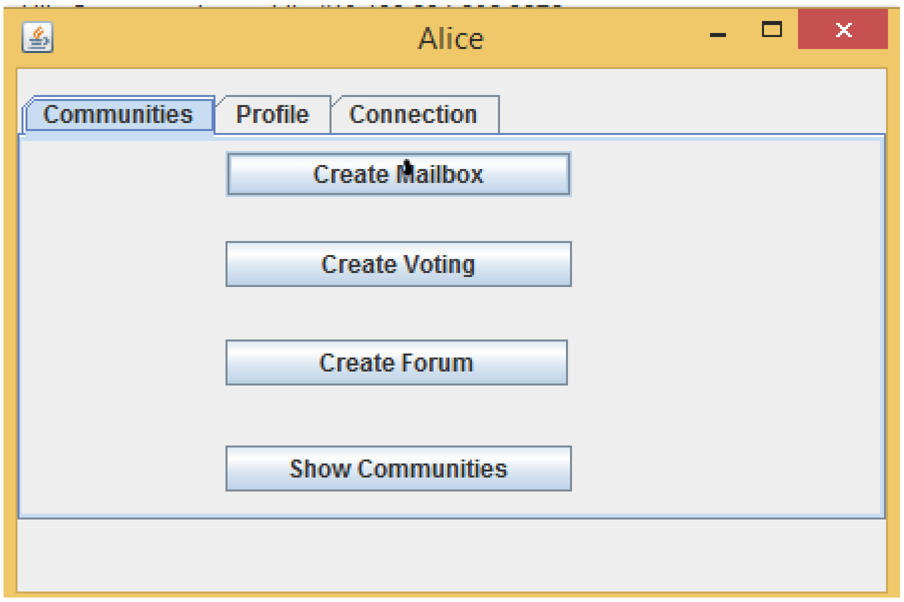
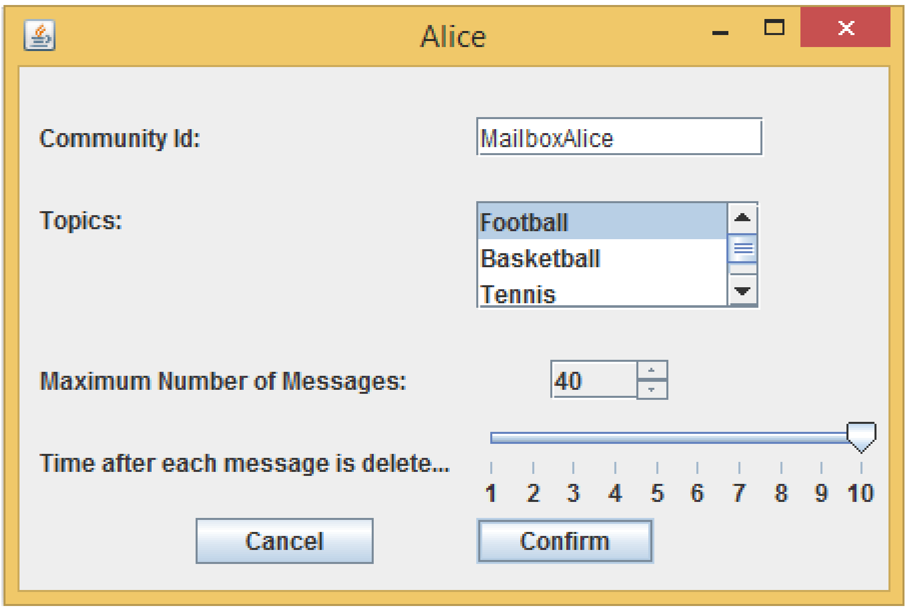


Figure 4: Alice Main

Alice will create the Mailbox community named MailboxAlice and topic "Football" wih the different parameters.



Community Id: MailboxAlice

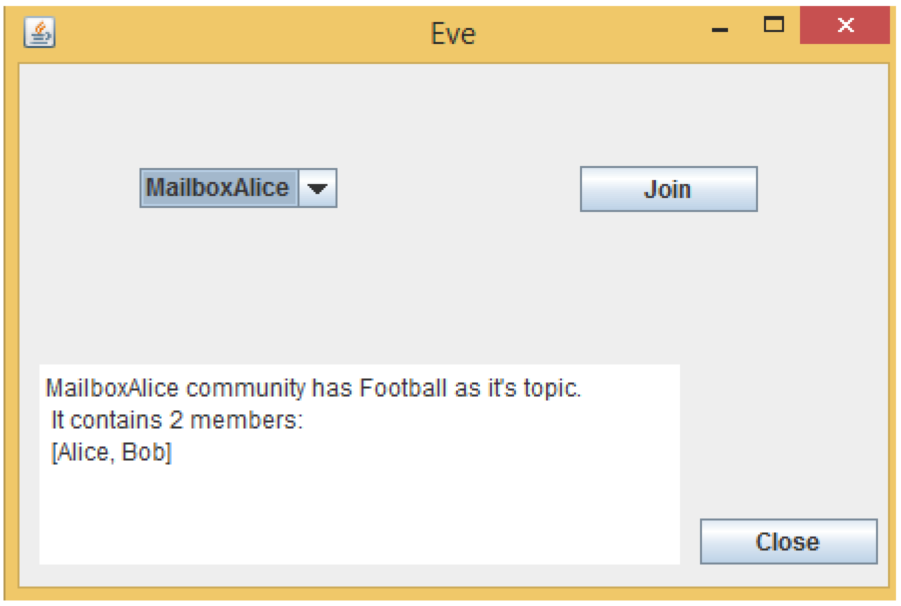
Topics: Football, Basketball, Tennis

Maximum Number of Messages: 40

Time after each message is delete... 1 2 3 4 5 6 7 8 9 10

Figure 5: Alice Creating Mail

Eve will join Alice's community. The information of the selected community will appear in a white box. The information are the topic of the community and the members. We can see that the members are Alice and Bob. Bob is a member although we didn't see that Bob has joined. This is because the agent of Bob is an **HelloAgent** agent, which joined the community automatically after Alice created it.



MailboxAlice community has Football as it's topic.
It contains 2 members:
[Alice, Bob]

Figure 6: Eve joins mail

After Eve joins the community, the agent of Bob will send a broadcast message to all the members (Eve and Alice) saying hello. Alice will enter to her mailbox to see her messages.

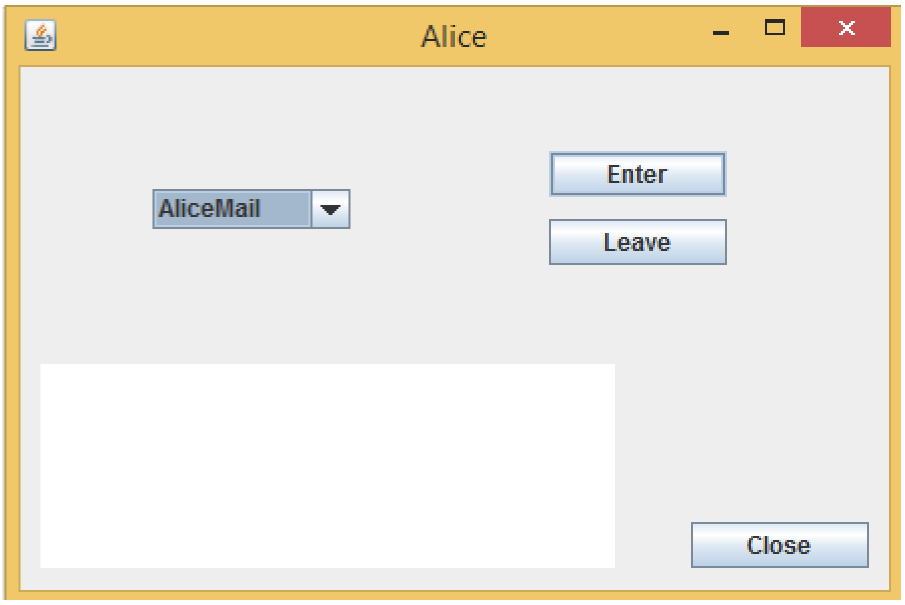
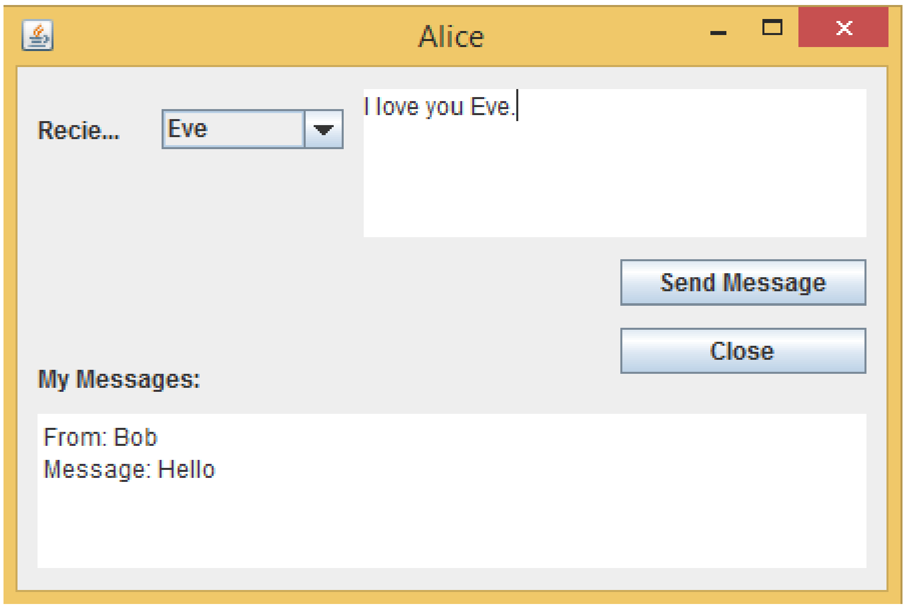


Figure 7: Alice enters her mail

Alice entered her mailbox and found a message from bob, she will now send a message to Eve expressing her love.

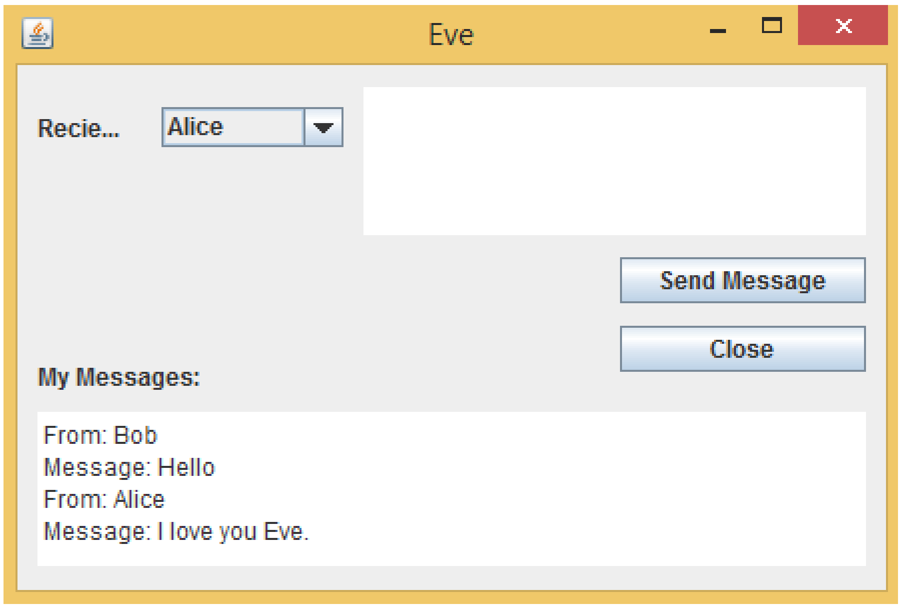


I love you Eve.

From: Bob
Message: Hello

Figure 8: Alice’s messages

Eve opens her mailbox and finds 2 messages from each of Bob and Alice.



From: Bob
Message: Hello
From: Alice
Message: I love you Eve.

Figure 9: Eve’s messages

4 Tasks done by the members

(a) Ahmad ALIBRAHIM
   i. HelloAgent and ServerAgent
   ii. All artifacts related to Mailbox
   iii. Graphical user interface
   iv. Java structure and classes

(b) Mohamed ZEMROUN

   i. Versatile-egocentric Agent
   ii. All artifacts related to Voting
   iii. Graphical user interface related to voting
   iv. Graphical user interface related to changing interests

(c) Adnan WALAYAT