



# **Report of Mini-project**

## **Multi-agent systems**

**Course:** Agent Technology

**Instructor:** Mrs. B. Dellal-Hedjazi

**Submitted by:** SAYOUD Maissa & BOULKABOUL Amira

**ID:** 191931040670 - 202031043294

**Dated:** May 25th, 2024

Department of Artificial Intelligence and Data Science

Masters in Intelligent Computer Systems

Faculty of Computer Science

USTHB

# Table of Contents

Introduction .....	1
First Part: Multi-Agent Negotiation (1 seller- several buyers).....	3
1. Introduction .....	3
2. MAS Framework Description .....	3
3. Auction Scenario Description .....	4
4. Implementation Details .....	5
5. Results .....	7
6. Conclusion.....	9
Second Part: Multi-Criteria Decision and Mobile Agents (1 buyer– several sellers) .....	11
1. Introduction .....	11
Mobile Buyer Agent .....	11
1. Scoring Function.....	11
2. Migration.....	12
3. Proposal Handling.....	13
4. End of Auction .....	13
5. Seller Agents .....	13
6. Results.....	14
4. Conclusion.....	15
Conclusion.....	16

# Introduction

Multi-agent systems (MAS) have emerged as powerful computational frameworks for modeling and simulating complex interactions among autonomous agents in various domains. In this mini-project, we delve into two fundamental aspects of MAS: multi-agent negotiation and multi-criteria decision-making with mobile agents. Through these two parts, we aim to demonstrate the versatility and effectiveness of MAS in modeling and solving real-world problems, ranging from market dynamics to resource allocation and beyond.

# **Part 1: Multi-Agent Negotiation**

# **First Part: Multi-Agent Negotiation**

## **(1 seller- several buyers)**

### **1. Introduction**

In this first part of the project, we were asked to implement a multi-agent negotiation scenario, focusing on an auction between one seller and several buyers within a Multi-Agent System (MAS).

The objective was to simulate the process of auctioning an item, where the seller initiates the auction by offering an item for sale at a specified starting price. Buyers then participate by offering bids higher than the initial price set by the seller. The auction progresses as the seller communicates the highest bid received to all participating buyers. This process will continue until either all buyers stop bidding or the predetermined auction duration elapses. At the end, the item is sold to the highest bidder if their final offer exceeds the reserve price which known only by the seller.

Great! Let's move on to the section where we describe the MAS framework used in your project and how it facilitates the auction scenario. Here's how we can structure this section:

### **2. MAS Framework Description**

The Multi-Agent System (MAS) framework employed in this project provides a platform for modeling and simulating complex negotiation scenarios that we will use for the auctions. The framework encompasses several key components, including agents, communication protocols, and decision-making mechanisms, which collectively enable autonomous interactions and decision-making within the system.

#### **1.1. MAS Architecture**

The MAS architecture is characterized by the presence of autonomous agents, each representing a participant in the negotiation process. Agents operate independently, they possess the ability to perceive their environment, to communicate with other agents, and to make decisions based on their goals and preferences.

## **1.2. Agent Characteristics**

Seller and buyer agents are instantiated within the MAS, each with specific characteristics and behaviors. Seller agents initiate and oversee the auction process, they are responsible for setting the starting price, reserve price, and managing the auction proceedings. While buyer agents, on the other hand, participate in bidding and decision-making activities, assess bidding opportunities, submit bids and adjust their strategies based on incoming information.

## **1.3. Communication Protocols**

Communication among agents is possible through the Agent Communication Language (ACL). Agents exchange messages to convey bids, auction updates, and negotiation proposals to ensure consistent and reliable communication between these latter.

## **1.4. Decision-Making Mechanisms**

Seller agents utilize decision rules to manage the auction process, including determining the starting price, evaluating bids, and enforcing auction termination conditions. Buyer agents employ bidding strategies based on factors such as their maximum bid limit, current highest bid, and risk preferences.

## **1.5. Integration with Auction Scenario**

The MAS framework is integrated with the auction scenario, with seller and buyer agents interacting within a shared environment to simulate the negotiation process. Seller agents instantiate the auction, set initial parameters (starting price, reserve price), and manage the progression of the auction through communication with buyer agents. Buyer agents actively participate in the auction by submitting bids, responding to auction updates, and adapting their strategies based on real-time information received from the seller and other buyers.

# **3. Auction Scenario Description**

The auction scenario simulates a negotiation process where a seller offers a product for sale to a group of potential buyers.

### **a. Initiation of the Auction by the Seller**

- The seller initiates the auction process by offering a product for sale at a specified starting price.

- At the same time, the seller sets a reserve price, which represents the minimum amount at which we are willing to sell the product. This reserve price is not shared and it is only revealed at the end of the auction.

#### **b. Participation of Buyers and Submission of Bids**

Buyers participate in the auction by submitting bids. We defined for each buyer a maximum budget they are willing to spend because we wanted to simulate a real auction. Buyers' bids are submitted concurrently and exceed the starting price set by the seller. Buyer agent evaluates whether the highest bid received is lower than their maximum bid. If this condition is verified, and a random decision is also satisfied, the buyer agent submits a new bid higher than the current highest bid.

#### **c. Progression of the Auction**

The seller regularly communicates to buyers the highest bid currently recorded. This information is used by buyers to adjust their bids and remain competitive in the auction. The auction process continues the expiration of a the time period (10 sec) or by the cessation of bids from all buyers.

#### **d. Determination of the Winning Bid**

At the end of the auction, if the last bid submitted by a buyer exceeds the reserve price set by the seller, the product is awarded to that buyer. However, if no bid exceeds the reserve price, the auction ends without a sale.

## **4. Implementation Details**

We used the JADE (Java Agent Development Framework) platform to implement the multi-agent auction system. The system is composed of a seller agent and multiple buyer agents. The main classes of the implementation are described below:

### **1. Main Class**

In this java class, we set up the JADE environment and initialize the agents involved in the auction. We create the the JADE runtime instance and configure the main container with the following parameters: host, port, and GUI settings. We create then the seller agent and three

buyer agents and launch them with specific arguments, including the starting price and reserve price for seller agent and maximum bid limits for buyer agents.

## **2. Seller Class**

In the Seller class, we extend Agent and manage the auction process, including initiating the auction, receiving bids, and determining the winning bid. The seller agent sets the starting and reserve prices and communicates with buyer agents by sending and receiving messages.

The auction ends when a predefined time limit is reached, and the seller determines if the highest bid meets or exceeds the reserve price.

- **Setup Method:** Initializes the seller's attributes based on the arguments provided and adds the behaviors necessary for the auction process (StartAuctionBehaviour, WakerBehaviour, and AuctionBehaviour).
- **Auction Behaviour:** It defines the main behavior of the seller during the auction. This method receives bids from buyers, updates the highest bid if a higher bid is received, and notifies all buyers of the new highest bid. If the auction has not started, it notifies all buyers of the starting bid.
- **End Auction Method:** This method checks if the highest bid meets or exceeds the reserve price. It announces the winner or says if the auction ended without a sale and deletes the seller agent after the auction ends.

## **3. Buyer Class**

The Buyer class extends Agent and participates in the auction by submitting bids. Each buyer agent has a maximum bid limit and decides whether to place a higher bid based on the current highest bid.

- **Setup Method:** Initializes the buyer's maximum bid based on the arguments provided from the main class and adds the BiddingBehaviour to handle the bidding process.
- **Bidding Behaviour:** It defines the behavior of the buyer during the auction. Then the action method receives updates from the seller about the current highest bid: If the highest bid is lower than the buyer's maximum bid, the buyer places a new higher bid.



## 5. Results

The execution of the auction scenario gave the following results:

```
http://DESKTOP-ITR9E5M:7778/acc
mai 25, 2024 10:20:56 AM jade.core.AgentContainerImpl joinPlatform
INFOS: -----
Agent container Main-Container@192.168.147.1 is ready.
-----

Buyer agent BuyerAgent1 started.
Buyer agent BuyerAgent2 started.
Seller agent started. Starting price: 100.0
Buyer agent BuyerAgent3 started.
BuyerAgent1 received: Current highest bid: 100.0
BuyerAgent2 received: Current highest bid: 100.0
BuyerAgent3 received: Current highest bid: 100.0
BuyerAgent2 bid: 108.0
BuyerAgent1 bid: 107.0
BuyerAgent3 bid: 108.0
BuyerAgent1 received: Current highest bid: 108.0
BuyerAgent2 received: Current highest bid: 108.0
BuyerAgent3 received: Current highest bid: 108.0
BuyerAgent1 bid: 113.0
BuyerAgent2 bid: 117.0
BuyerAgent3 bid: 110.0
BuyerAgent2 received: Current highest bid: 113.0
BuyerAgent1 received: Current highest bid: 113.0
BuyerAgent3 received: Current highest bid: 113.0
BuyerAgent2 bid: 119.0
BuyerAgent3 bid: 122.0
BuyerAgent1 bid: 114.0
BuyerAgent1 received: Current highest bid: 117.0
BuyerAgent2 received: Current highest bid: 117.0
BuyerAgent3 received: Current highest bid: 117.0
BuyerAgent1 bid: 121.0
BuyerAgent3 bid: 125.0
BuyerAgent2 bid: 126.0
BuyerAgent1 received: Current highest bid: 119.0
BuyerAgent3 received: Current highest bid: 119.0
BuyerAgent2 received: Current highest bid: 119.0
```

Figure 1. First part of output of the execution

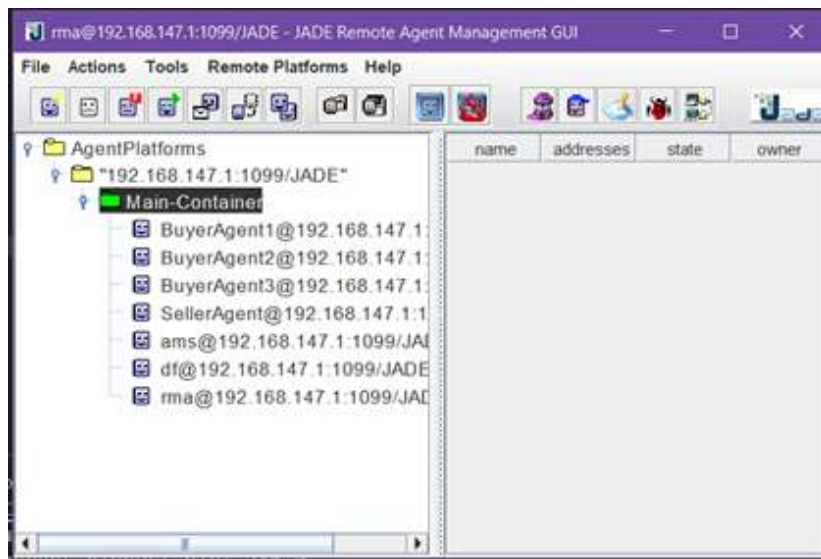


Figure 2. Content of Main-Container before the end of the auction scenario

```

Main (30) [Java Application] C:\Users\ADMIN\p2\pool\plugins\org.eclipse.justj.open
BuyerAgent3 bid: 264.0
BuyerAgent1 received: Current highest bid: 264.0
BuyerAgent2 received: Current highest bid: 264.0
BuyerAgent3 received: Current highest bid: 264.0
BuyerAgent3 bid: 265.0
BuyerAgent1 received: Current highest bid: 265.0
BuyerAgent3 received: Current highest bid: 265.0
BuyerAgent2 received: Current highest bid: 265.0
BuyerAgent3 bid: 273.0
BuyerAgent1 received: Current highest bid: 273.0
BuyerAgent3 received: Current highest bid: 273.0
BuyerAgent2 received: Current highest bid: 273.0
BuyerAgent3 bid: 274.0
BuyerAgent1 received: Current highest bid: 274.0
BuyerAgent2 received: Current highest bid: 274.0
BuyerAgent3 received: Current highest bid: 274.0
BuyerAgent3 bid: 283.0
BuyerAgent2 received: Current highest bid: 283.0
BuyerAgent1 received: Current highest bid: 283.0
BuyerAgent3 received: Current highest bid: 283.0
BuyerAgent3 bid: 284.0
BuyerAgent2 received: Current highest bid: 284.0
BuyerAgent1 received: Current highest bid: 284.0
BuyerAgent3 received: Current highest bid: 284.0
BuyerAgent3 bid: 286.0
BuyerAgent1 received: Current highest bid: 286.0
BuyerAgent2 received: Current highest bid: 286.0
BuyerAgent3 received: Current highest bid: 286.0
BuyerAgent3 bid: 288.0
BuyerAgent1 received: Current highest bid: 288.0
BuyerAgent2 received: Current highest bid: 288.0
BuyerAgent3 received: Current highest bid: 288.0
BuyerAgent3 bid: 296.0
BuyerAgent1 received: Current highest bid: 296.0
BuyerAgent3 received: Current highest bid: 296.0
BuyerAgent2 received: Current highest bid: 296.0
Auction ended. Winner: BuyerAgent3 with bid: 296.0

```

Figure 3. Last part of output of the execution

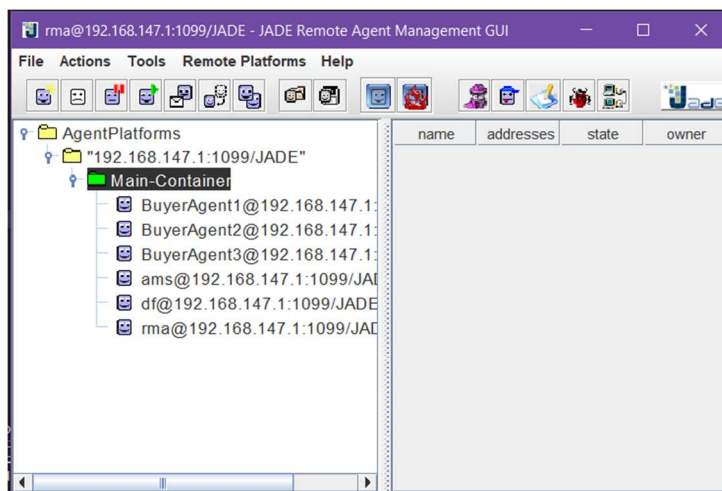


Figure 4. Content of Main-Container after the end of the auction scenario

(Seller Agent Deleted himself)

So, the seller agent started the auction with a starting price of **100.0**. Then BuyerAgent1, BuyerAgent2 and BuyerAgent3 received the notification and started participating in the auction. The auction ended with with **BuyerAgent3** winning the auction with a bid of **296.0**.

## **6. Conclusion**

The auction scenario was executed successfully. Buyers actively participated by making increasing bids, and ultimately, 'BuyerAgent3' won the auction with a bid of 296.0 that exceeds the reserve price set by the seller: 110.0.

## **Part 2: Multi-Criteria Decision and Mobile Agents**

# **Second Part: Multi-Criteria Decision and Mobile Agents**

## **(1 buyer– several sellers)**

### **1. Introduction**

This project implements a multi-agent system based on the JADE (Java Agent DEvelopment Framework) platform to simulate an auction environment. Two types of agents are defined: Seller agents and Mobile Buyer agents. The agents negotiate the sale of products based on several criteria, and the buyer agent migrates between different containers and choose the best criteria depending on the best score.

### **Mobile Buyer Agent**

#### **1. Scoring Function**

The core of the buyer agent's decision-making process is the scoring function, which evaluates each seller's proposal based on three criteria: price, quality, and delivery cost. Each criterion is assigned a weight that reflects its importance to the buyer knowing that the buyer.

##### **1.1. Criteria and Weights:**

- Price (40%): The cost of the product is a significant factor, with a lower price being more favorable.
- Quality (50%): The quality of the product is the most heavily weighted criterion, emphasizing the buyer's preference for higher quality products.
- Delivery Cost (10%): The cost of delivery is the least important criterion but still considered in the evaluation.

##### **1.2. Normalization**

To ensure fair comparison, each criterion is normalized to a scale of 0 to 1, based on predefined minimum and maximum values:

- Price: Normalized using a minimum of 50 and a maximum of 150 units.
- Quality: Normalized using a scale from 1 to 10.

- Delivery Cost: Normalized using a minimum of 5 and a maximum of 20 units.

### 1.3. Scoring Calculation

The scoring function calculates the overall score for each proposal by combining the normalized values of each criterion with their respective weights:

$$\text{Score} = (0.4 \times (1 - \text{Normalized Price})) + (0.5 \times \text{Normalized Quality}) + (0.1 \times (1 - \text{Normalized Delivery Cost}))$$

- Price: The normalized price is subtracted from 1 to ensure that lower prices result in higher scores.
- Quality: The normalized quality is used directly, as higher quality should yield a higher score.
- Delivery Cost: Similar to price, the normalized delivery cost is subtracted from 1 to prefer the lower costs.

## 2. Migration

Migration is a critical behavior of the Mobile Buyer agent, allowing it to move between containers within the JADE platform. This behavior is implemented to simulate a scenario where the buyer agent can access different markets or platforms to gather more offers.

In this auction scenario, we tried to make two (2) types of migration:

### 2.1. Inter-Container Migration

**Initial Migration:** The buyer agent first migrates to a specified container after a delay (the delay is to make the migration visible, so we can notice it) allowing it to interact with Seller agents in that container.

**Delayed Migration:** After collecting proposals in the first container, the buyer agent migrates to a second container after another specified delay, further expanding its search for the best offer.

### 2.2. Inter-platform migration

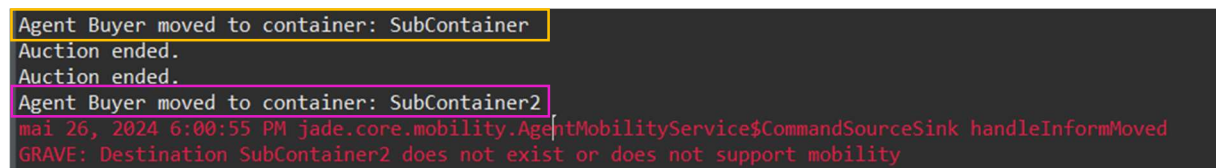
We worked on a program that migrate an agent from a platform to another, so we created a new platform with the same localhost but a different port, we executed it but sadly, after several

useless tries and scanning the documentation, we concluded that it is not possible to migrate agents through platforms (basically this type of mobility ain't supported).

**Error :**

**jade.core.mobility.AgentMobilityService\$CommandSourceSink  
handleInformMoved**

**GRAVE: Destination SubContainer2 does not exist or does not  
support mobility**



```
Agent Buyer moved to container: SubContainer
Auction ended.
Auction ended.
Agent Buyer moved to container: SubContainer2
mai 26, 2024 6:00:55 PM jade.core.mobility.AgentMobilityService$CommandSourceSink handleInformMoved
GRAVE: Destination SubContainer2 does not exist or does not support mobility
```

*Figure 4. Error showing that migration through platforms (not containers) is impossible*

As we can observe the figure, the yellow rectangle shows that the buyer agent has migrated from mainContainer to SubContainer Successfully. While the pink rectangle, shows the message that he migrated but he actually didn't, an error is shown just below the message.

### 3. Proposal Handling

The **buyer agent** continuously listens for proposals from seller agents. When a proposal is received, it is evaluated using the scoring function. The agent keeps track of the highest scoring proposal, updating its best offer if a new proposal has a higher score.

### 4. End of Auction

The auction concludes when the seller agents send an "Auction ended" message. The buyer agent then announces the winner, which is the seller with the highest scoring proposal, or indicates if no valid proposals were received.

### 5. Seller Agents

Seller agents generate proposals based on random values for price, quality, and delivery cost. Each seller sends its proposal to the buyer agent and then signals the end of the auction after a specified delay.

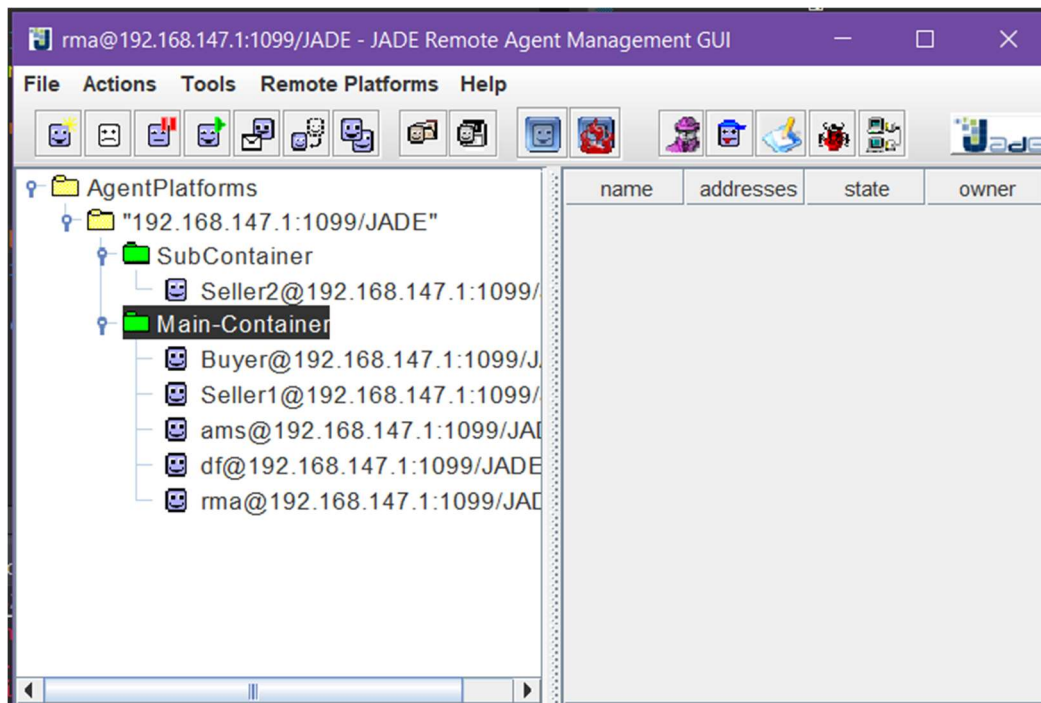


## 6. Results

After Executing the programs, we get:

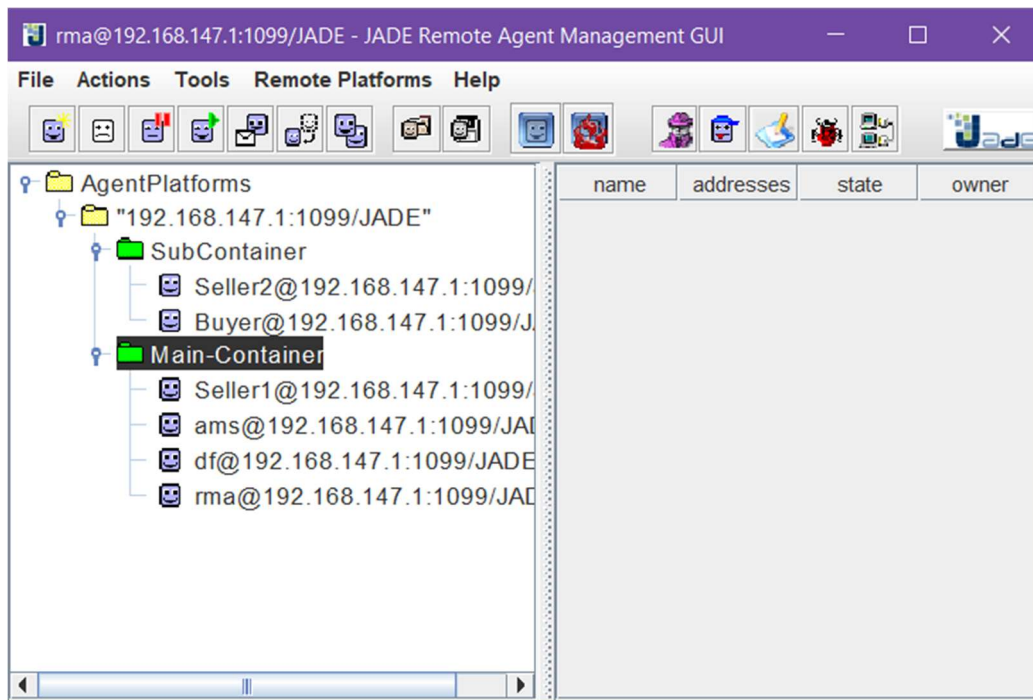
```
Mobile Buyer Agent Buyer started.
Seller agent started. Criteria: {Price=94.28171971531006, Quality=5.900726832027356, DeliveryCost=24.046975874400115}
Seller agent started. Criteria: {Price=119.24895206280603, Quality=5.078734177745091, DeliveryCost=8.475471468438853}
Agent Buyer moved to container: SubContainer
Auction ended.
Auction ended.
Agent Buyer moved to container: SubContainer2
mai 26, 2024 4:28:36 PM jade.core.mobility.AgentMobilityService$CommandSourceSink handleInformMoved
GRAVE: Destination SubContainer2 does not exist or does not support mobility
Received proposal from Seller1 with score: 0.4264307251672442
Auction winner is Seller1 with score: 0.4264307251672442
```

*Figure 6. Result of the auction scenario*



*Figure 7. Buyer Agent before migration to SubContainer*





*Figure8. Buyer Agent after migration to SubContainer*

## 4. Conclusion

This project shows the capabilities of JADE in creating a multi-agent system that simulates an auction environment. The mobile buyer agent's ability to migrate between different containers and evaluate proposals based on a detailed scoring function highlights the flexibility and power of agent-based systems in handling complex negotiation tasks.

# Conclusion

In conclusion, this mini-project has provided valuable insights into the capabilities and applications of multi-agent systems. By simulating scenarios of multi-agent negotiation and multi-criteria decision-making, we have illustrated how MAS can effectively model and solve complex problems in dynamic environments.

Through the auction scenario, we observed how agents engage in negotiation processes to determine optimal prices and facilitate transactions between buyers and sellers. Additionally, the introduction of mobile agents in the second part highlights the adaptability and scalability of MAS in handling diverse tasks across distributed environments.