1. **Introduction**

In this assignment, we try to develop an agent to negotiate the other agents. We have a case which I want to organize a party with my two friends but as being normal everybody has different preferences. I decided to code an agent who will be known my preferences and it will negotiate with others in behalf of me and try to reach a conclusion with high utility depends on my preferences.

We designed an agent to decide party options in behalf of us. In this project, we used a software environment is named as Genius which can run agents that are written in Java. It provides a testbed for negotiating agents.

First, we create three preferences profiles. Then, we started to thinking about our strategy. We applied many different strategies but at the end of we decide on one which will be explained below. There are many reasons to change our strategies. We changed our strategies after negotiating the other agents because we see our weaknesses against the other. We consider the win-win approach and most of our strategies depends on it. Also, we planned and had concrete strategies. As I said before, we changed our strategy many times to improve its stability and reliability. Moreover, we decide some reservation values which are time limit and minimum utility.

1. **Description and Structure**

We selected our reservation values as bellows ***Figure 1***;



Figure 1.Reservation values

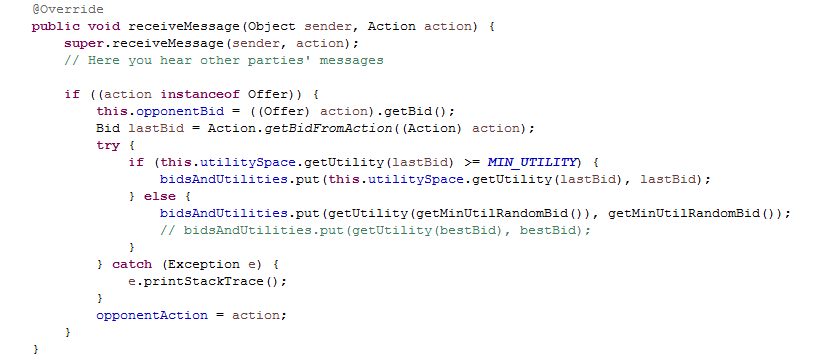
Our main strategy depends on that we collect all offered bids to us and put them to hash map with their utility amounts. ***Figure 2***

Figure 2. ReceiveMessage Method

As it is shown above that if last offered bid for us is bigger than or equal to our minimum utility amount we put them to hash map which is named **bidsAndUtilities**. If it is not bigger than or equal to minimum utility, we generate random bid with bigger than minimum utility condition and put it to **bidsAndUtilities** Hash Map.

**getMinUtilRandomBid()** method is shown below ***Figure 3:***

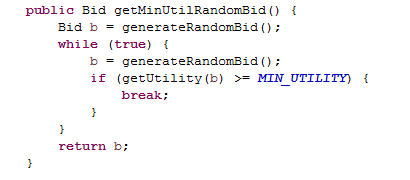


Figure 3. getMinUtilRandomBid Method

Agent’s **chooseAction()** method is in ***Figure 4***.:

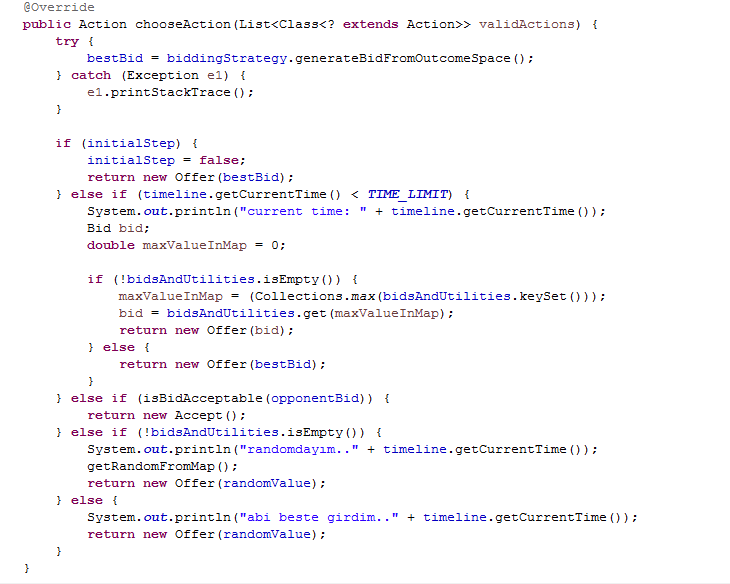


Figure 4.chooseAction Method

As it is seen that we have a best initial offer with **bestBid**. We initially offer best bid. It is generated in **generateBidFromOutcomeSpace()** method **of Group4BiddingStrategy** class. ***Figure 5.*** After initial bid offered our strategy depends on time limit. If it is smaller than limit control the bidsAndUtilities hash map and retrieve the maximum bid than offer it. If bidsAndUtilities map is empty, it offers to bestBid. If time limit is exceeded, now agent controls the offered bids to me are they acceptable. It checks it with **isBidAcceptable()** method ***Figure 6.*** It if is not acceptable to my preferences, agent start to offer randomly bid from my bidsAndUtilities hash map.

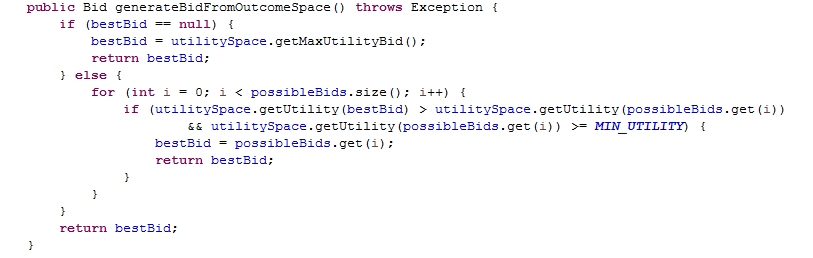


Figure 5.generateFromOutcomeSpace Method

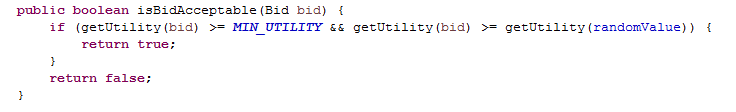


Figure 6. isBidAcceptable Method

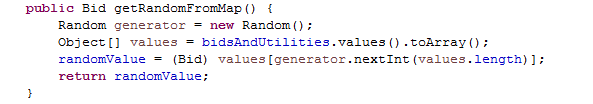


Figure 7.getRandomFromMap Method

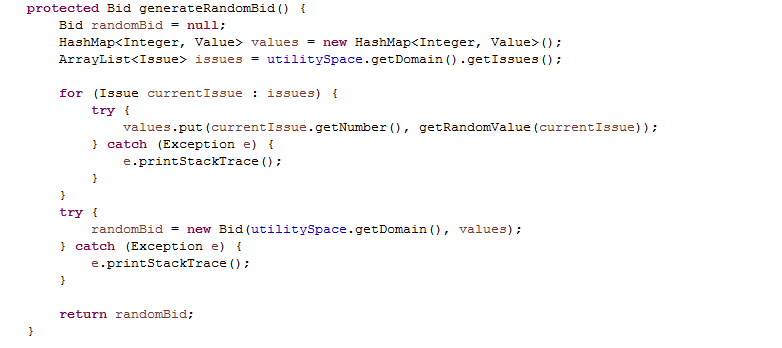


Figure 8.generateRandomBid Method

**2.1. Methods Descriptions**

**generateRandomBid():** This method generates random bid. ***Figure 8.***

**getRandomFromMap():** Retrieve a bid randomly from hash map which all old bids are stored. ***Figure 7.***

**isBidAcceptable(Bid bid):** This checks that is the utility of bid bigger than minimum utility and utility of coming up offered bid. ***Figure 6.***

**generateFromOutcomeSpace():** It creates bids which has maximum utility. ***Figure 5.***

**getMinUtilRandomBid():** This generate random bids but it considers on the minimum utility.***Figure 3.***