Homework 3

Mattias Cederlund, mcede@kth.se

Task #1

First off I created a helper function to set up the agent container with N QueenAgents. Each QueenAgent is provided their own row-index and the total number of rows in the board. The QueenAgents are named q0, q2, … so every QueenAgent easily can find and send messages to their predecessor and successor. All agents use a CyclicBehaviour to receive messages from other agents.

The first agent, q0, will start generating a solution to the N-queens problem by creating an empty representation of the board (array with N size, filled with zeros), and start the placement-finding algorithm.

The algorithm will try to make a placement, and check if it is valid based on previously placed queens. If it finds a valid placement, it will send a message to the next agent containing information about the current placements for all queens. The agents will continue until the Nth agent found a valid placement. If it finds a solution, it will present the result.

If a QueenAgent can't find any valid placement, it will send a message to its predecessor and ask it to re-position. That agent will try to find another placement and if it succeeds it will send a message to the next queen which will try positioning again. In the worst case, if the first queen can't find a valid placement, there does not exist any solution to the problem.

To create multiple solutions for the same N, I manually send a message to the last queen to re-position itself. It will start the backtracking chain and end with a new solution (if there exists any).

To distinguish the forward and backward messages I used two performatives, INFORM and FAILURE. The only difference in the message handling is that when a FAILURE message is received it should continue the placement algorithm from the previous placement + 1 instead of starting over on the row.

Task #2

For the second task I started the Jade runtime with three containers. One containing the ArtistManagerAgent and two containing one CuratorAgent each.

I modified the ArtistManagerAgent to fetch the available containers, using the snipped provided by the Jade programming guide. Upon receiving the containers I cloned one ArtistManagerAgent into each of the containers. The original agent stayed in its container and started listening for messages sent by its clones, collecting the final sell prices.

In ArtistManagerAgents afterClone I added the AuctioneerBehaviour, with only a slight modification: The behaviours onEnd was overridden to send a message to the original agent with the END_OF_AUCTION message containing the price before, just before moving back to the original container. Also, upon moving back the clone would terminate shortly.

When the original ArtistManagerAgent received messages from all its clones it will tell the highest price.

The only difference to the CuratorAgent, compared to the HW2 implementation, is the fact that it will clone itself a few times before initializing its BuyerBehaviour.