# Homework 3 - Linda with Apache ActiveMQ

**Deadline : Friday , 2019/4/26 23:59**

## Overview

In this project, we are going to implement Linda with three operations by using Apache ActiveMQ as the middleware.

## Tutorials

### 1. Linda

Linda is a novel system for communication and synchronization. In Linda, independent process communicate via an abstract tuple space, unlike objects, tuples are pure data, they do not have any associated methods.

In this project you are going to implement three operations that individual workers perform on the tuples and the tuple space:

- **out:** produces a tuple, writing it into tuple space
- **in:** atomically reads and removes—consumes—a tuple from tuple space.
- **read:** non-destructively reads a tuple space.

### 2. ActiveMQ

Apache ActiveMQ is an example of Message Oriented Middleware (MoM). Implementations are designed for the purpose of sending messages between two or more applications.
In this project, we are going to design a Linda server as a tuple space, which use to store tuples, and clients to create tuples. Server and clients communicate through ActiveMQ.

## Specifications

1. Data type

- String : the data between “ and ”, for example “abc”.
- Integer : the data consists of digits and without “”, for example 123.

2. Operation format

- [in/out/read] [value1] [value2]…
- There is a space between each fields.
- The quantity of field will be less than 200.
- The length of each operation will be less than 1024 characters.

3. Operation without ?

- **out “abc” 2 5** : client will send this message as a topic to ActiveMQ, and the server will put tuple ( “abc” , 2 , 5 ) into the tuple space.
- **read “abc” 2 5** : client send this message as a topic to ActiveMQ and receive a tuple ( “abc” , 2 , 5 ) but this tuple in tuple space won’t be removed.
- **in “abc” 2 5** : client will send this message as a topic to ActiveMQ and receive a tuple ( “abc” , 2 , 5 ) and this tuple in tuple space will be removed.

4. Operation with ?

- **read “abc” 2 ?i**
- **in “abc” 2 ?i** : this operation "searches” the tuple space for a tuple consisting of the string “abc", the integer 2, and a third field containing anything. If found, the tuple is removed from the tuple space and the variable i is assigned the value of the third field. The matching and removal are atomic, so if two processes execute the same in operation simultaneously, only one of them will succeed, unless two or more matching tuples are present.

The fields of the in primitive, called the template, are compared to the corresponding fields of every tuple in the tuple space. A match occurs if the following three conditions are all met:

- The template and the tuple have the same number of fields.
- Each constant or variable in the template matches its tuple field.
- The types of the corresponding fields are equal.

5. Suspension

If no matching tuple is present, the calling process is suspended until another process inserts the needed tuple, at which time the called is automatically revived and given the new tuple. The fact that processes block and unblock automatically means that if one process is about to output a tuple and another is about to input it, it does not matter which goes first. The only difference is that if the in is done before the out, there will be a slight delay until the tuple is available for removal.

## Scenarios

**out "abc" 2 5** : put ( "abc" , 2 , 5 ) into the tuple space.

**in "abc" "2" ?j** : there is no match for this request, so this client will be suspended.

**in "abc" 2 ?i** : assign 5 to i, then send tuple ( "abc" , 2 , 5 ) back to the client, and the tuple( "abc" , 2 , 5 ) in the tuple space would be removed.

**in "abc" 2 5** : this is the second client to execute this operation, so this client will be suspended.

**out "abc" "2" 7** : assign 7 to j and the client which is suspended will get the tuple ( "abc" , "2" , 7 ) then remove the tuple from the tuple space.

**out "def" j i** : put ( "def" , 7 , 5 ) into the tuple space.

**read "def" ?a ?b** : assign 7 to a and 5 to b and client would receive a tuple ( "def" , 7 , 5 ) but this tuple won't be removed.

**in "def" a b** : client would receive a tuple ( "def" , 7 , 5 ) and this tuple will be removed.

## Note

1. We have no limitation on the programming language.
2. The ActiveMQ, Linda server/client should be executed on the AWS instance.
3. If tuple space changes, server should print out the tuple space.
4. When In/Read a tuple that doesn't exist, client should be suspend until other client insert a matching tuple.

## File submission

Create a directory named <student_id>.zip
The structure should be :
<student_id>

```
        |—server.xx
        |—client.xx
```

Zip the directory into <student_id>.zip and upload it to the new E3 platform.
TA would validate your source codes by cheating detection. Please finish the assignment by yourself.

## Reference

- Apache ActiveMQ
- Linda
- Linda programming