CS249 - Homework 5

Group 6: Mayuri Wadkar

Eric Han

Sonali Mishra

Instructions to run:

$ git clone https://github.com/Mayuri-Wad-012447851/CS-249-Distributed-Computing

Cloning into 'CS-249-Distributed-Computing'...

remote: Counting objects: 36, done.

remote: Compressing objects: 100% (32/32), done.

remote: Total 36 (delta 4), reused 35 (delta 3), pack-reused 0

Unpacking objects: 100% (36/36), done.

Instructions to run the program from command prompt:

1. On command prompt, change directory to location of file (cloning location).

2. To compile and run, issue following commands:

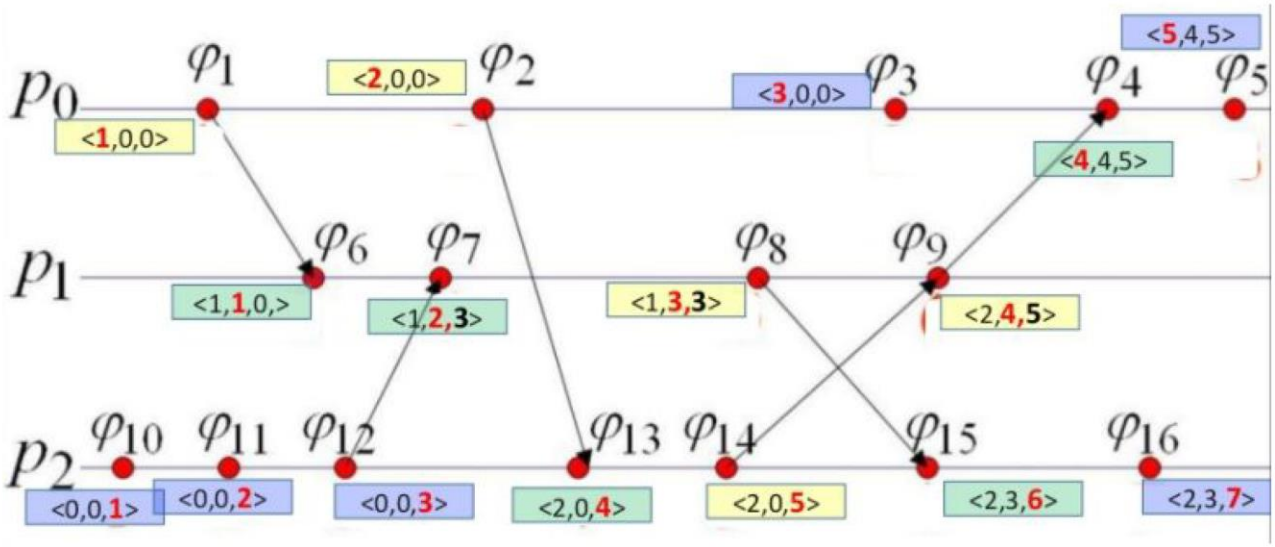
Example command 1:

$..\CS-249-Distributed-Computing\HW5-Vector-clock\src>"C:\Program Files\Java\jdk1.8.0_121\bin\javac" *.java

Example command 2:

$..\CS-249-Distributed-Computing\HW5-Vector-clock\src >"C:\Program Files\Java\jdk1.8.0_121\bin\java" Algorithm

Input:



Question 1 Implement the design of a simulation that will Calculate vector clocks

1- Interactions

Compute Event- Increment the vector clock at processor's own index by one.

Send Event- Processor i sends an event to Processor j and increments its vector clock at its position by one.

Receive Event- Upon receiving a send event from Processor i, Processor j will do three things:

a) compares its VC element at position i to the VC element at position i received from processor i, and update its VC array element at index i to the greater value

b) update its VC array element corresponding to itself by one.

c) when looking at the VC received from processor i, if there are any elements in that VC that do not exist in its own VC, processor j will copy those values received from processor i over to its own VC (processor j).

2- What we need to store

We first need to store the events that each process has for execution. We then need to store the vector clock for each process as it gets updated with events within each process. While a send message we need to store and send the message as well as the current state of vector clock of the process with the send event.

4- What needs to change

The VectorClocks of the Processors will change depending on the event message received, which can be a compute event, send event or receive event.

5- What will be the input? [The input will be different each time, user should have the ability to control input and get different results]

The input for this implementation will be the processors, and execution plans for them. The execution plan for each processor will include the events of the processor and the event type of each event, if it is a compute, send or receive event.

6- What will be the output?

Implementation updates integer array of vector clocks after each computation, send or receive event. Output of the implementation will be integer array of VCs (vector clocks) with updated timestamps.

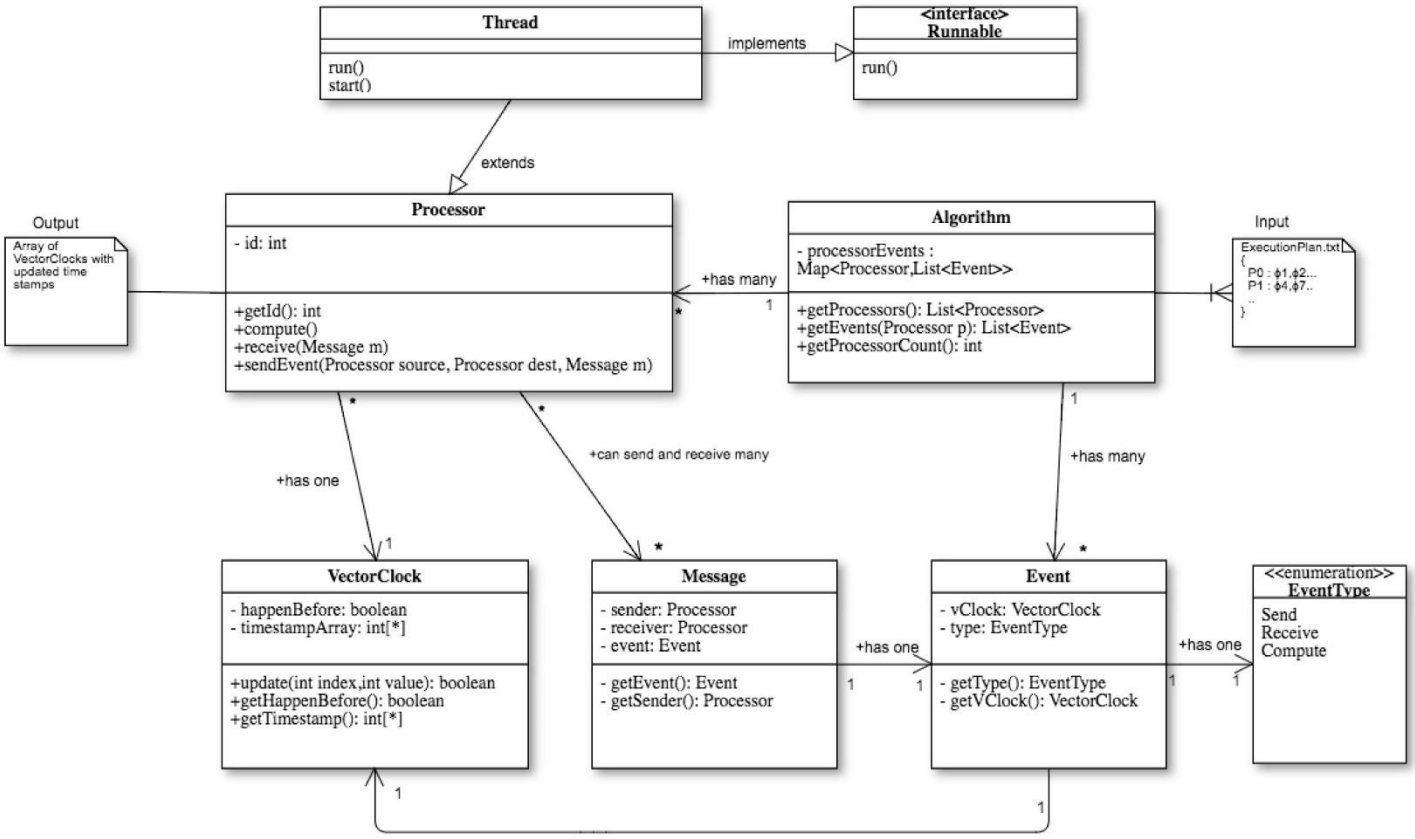
7- Decide which event happened Before, which event is concurrent.

If an event is the sending of the message by one process and another event is the receiving of the same message by another process, then the sender event would happen before receive event. Else all events are concurrent events.

Hence if Processor P0 sends message to Processor P1, and if a is the send event of P0, b is the receive event of P1, then a happen before (-->) b. In all other situations event a and b will be concurrent.

So we check the sender event and receiver event of a message to decide the happened before or concurrent relationship.

Class Diagram:



Sequence Diagram: