# ASSIGNMENT-4

**Aim and Objective:** To develop Token Ring distributed algorithm for leader election

**Tools / Environment:** Java Programming Environment, JDK 1.8, EclipseNeon(EE).

**CODES**

**package** As4;

**import** java.io.\*;

**import** java.util.Scanner;

**publicclass** bully {

## staticint*n*;

**staticint***pro*[] = **newint**[100]; **staticint***sta*[] = **newint**[100]; **staticint***co*;

**publicstaticvoid** main(String args[]) **throws** IOException { System.***out***.println("Enter the number of process");

@SuppressWarnings("resource") Scanner in = **new** Scanner(System.***in***); *n* = in.nextInt();

**int**i;

**for** (i = 0; i<*n*; i++) {

System.***out***.println("For process " + (i + 1) + ":"); System.***out***.println("Status:");

*sta*[i] = in.nextInt(); System.***out***.println("Priority"); *pro*[i] = in.nextInt();

}

System.***out***.println("Which process will initiate election?");

**int**ele = in.nextInt();

*elect*(ele);

System.***out***.println("Final coordinator is " + *co*);

}

**staticvoid** elect(**int**ele) {

ele = ele - 1;

*co* = ele + 1;

**for** (**int**i = 0; i<*n*; i++) {

**if** (*pro*[ele] <*pro*[i]) {

System.***out***.println("Election message is sent from " +

(ele + 1) + " to " + (i + 1));

}

**if** (*sta*[i] == 1)

*elect*(i + 1);

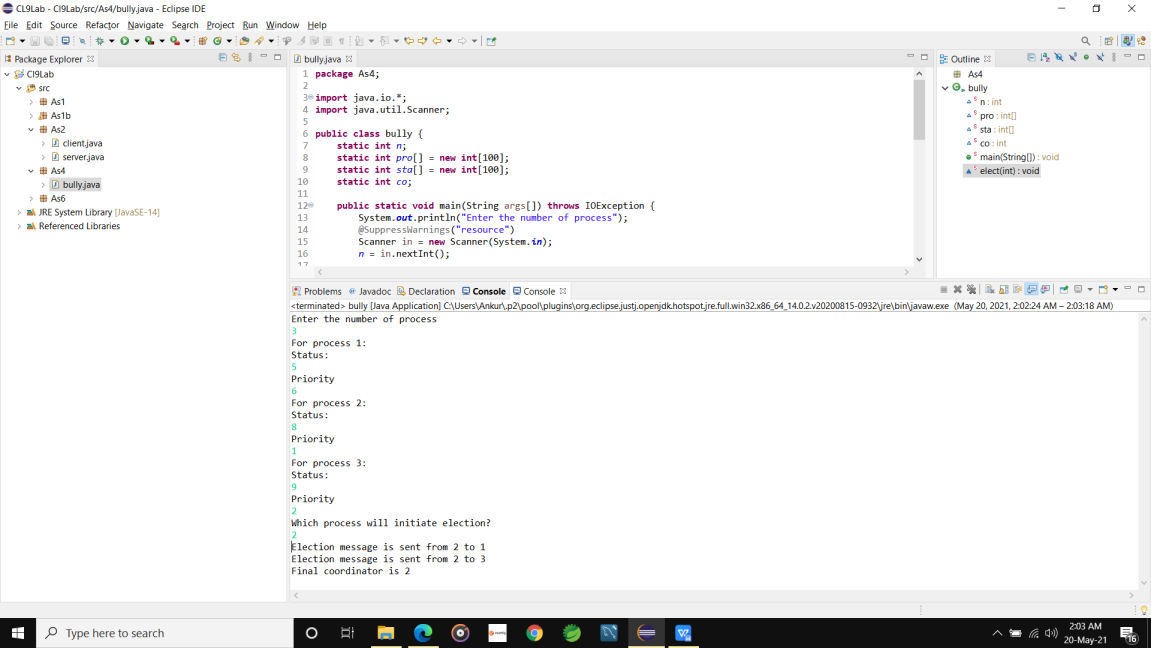
}

}

}

## Related Theory:

**Outputs**



**Election Algorithm:**

1. Many distributed algorithms require a process to act as a coordinator.
2. The coordinator can be any process that organizes actions of other processes.
3. *A* coordinator may fail.
4. How is a new coordinator chosen or elected?

## Assumptions:

Each process has a unique number to distinguish them. Processes know each other's process number.

There are two types of Distributed Algorithms:

1. Bully Algorithm
2. Ring Algorithm

## BullyAlgorithm:

* 1. **When a process, P, notices that the coordinator is no longer responding to requests, itinitiates an election.**
     1. P sends an ELECTION message to all processes with higher numbers.
     2. If no one responds, P wins the election and becomes a coordinator.
     3. If one of the higher-ups answers, it takes over. P’s job is done.

## When a process gets an ELECTION message from one of its lower-numbered colleagues:

* + 1. Receivers ends an OK message back to the sender to indicate that he is a live and will take over.
    2. Eventually, all processes giveup apart of one, and that one is the new coordinator.
    3. The new coordinator announce s*its* victory by sending all processesa **CO- ORDINATOR**message telling them that it is the new coordinator.

## If a process that *was* previously down comes back:

* + 1. It holds an election.

If it happens to be the highest process currently running, it will win the election and takeover the coordinators job.

## Conclusion:

Election algorithms **are designed to choose a coordinator.** We have two election algorithms for two different configurations of distributed system. **The Bully** algorithm applies to system where every process can send a message to every other process in the system and **The Ring** algorithm m applies to systems organized as a ring (logically or physically). In this algorithm we assume that the link between the process are unidirectional and every process can message to the process on its right only.