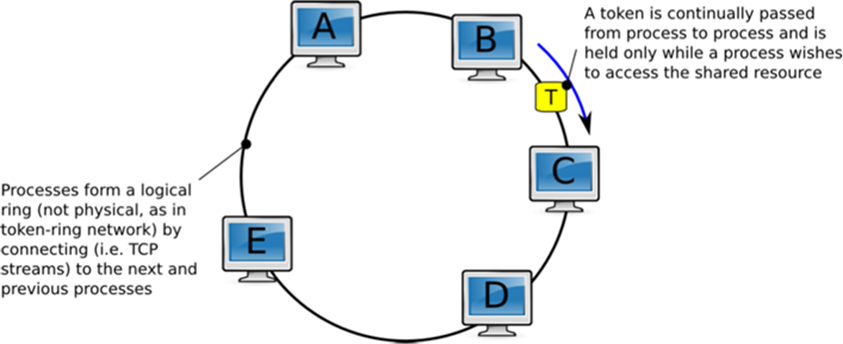
**Experiment No: 02**

**Aim: To implement mutual exclusion in Distributed System (Token Ring Algorithm)**

**Theory :**

Token Ring Algorithm:

****

* A completely different approach to achieving mutual exclusion in a distributed system. Here we have a bus network (for e.g. ethernet).
* A logical ring is constructed in which each process is assigned a position in the ring. The ring positions may be allocated in numerical order of network addresses or some other means. It does not matter what the ordering is.
* All that matters is that each process knows who is next in line after itself.
* When the ring is initialized, process A (say) is given a token. The token circulates around the ring. It is passed from process k to process k+1 in point-to-point messages.
* When a process acquires the token from its neighbor, it checks to see if it is attempting to enter a CS. If so, the process enters the CS, does all the work it needs to, and leaves the section.
* After it has exited, it passes the token along the ring. It is not permitted to enter a second CS using the same token.
* If a process is handed the token by its neighbor and is not interested in entering a CS, it just passes it along. As a consequence, when no processes want to enter any CS, the token just circulates at high speed around the ring.

**Advantages:**

* The correctness of this algorithm is evident. Only one process has the token at any instant, so only one process can be in a CS.
* Since the token circulates among processes in a well-defined order, starvation cannot occur.

**Disadvantages:**

* Once a process decides it wants to enter a CS, at worst it will have to wait for every other process to enter and leave one critical region.
* If the token is ever lost, it must be regenerated. In fact, detecting that it is lost is difficult, since the amount of time between successive appearances of the token on the network is not a constant. The fact that the token has not been spotted for an hour does not mean that it has been lost; some process may still be using it.
* The algorithm also runs into trouble if a process crashes, but recovery is easier than in the other cases. If we require a process receiving the token to acknowledge receipt, a dead process will be detected when its neighbor tries to give it the token and fails. At that point the dead process can be removed from the group, and the token holder can pass the token to the next member down the line.

**Code:**

import java.io.\*;

import java.lang.\*;

public class Tring {

 public static void main(String args[]) throws Throwable {

  DataInputStream dis = new DataInputStream(System.in);

  System.out.println("Enter the num of nodes:");

  int n = Integer.parseInt(dis.readLine());

  // Decides the number of nodes forming the ring

  int token = 0;

  int ch = 1;

  for (int i = 0; i < n; i++)

   System.out.print(" " + i);

  System.out.println(" " + 0);

  try {

   while (ch == 1) {

    System.out.println("Enter sender:");

    int s = Integer.parseInt(dis.readLine());

    System.out.println("Enter receiver:");

    int r = Integer.parseInt(dis.readLine());

    System.out.println("Enter Data:");

    String d = dis.readLine();

    System.out.print("Token passing:");

    for (int i = token; i != s; i++)

     System.out.print(" " + i + "->");

    System.out.println(" " + s);

    System.out.println("Sender " + s + "sending data: " + d);

    for (int i = s + 1; i != r; i = (i + 1) % n)

     System.out.println("data  " + d + " forwarded by " + i);

    System.out.println("Receiver " + r + "received data: " + d);

    token = s;

 }

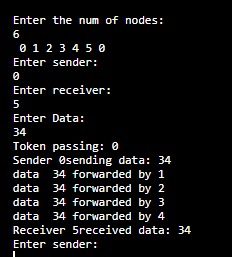
  } catch (Exception e) {

  }

 }

}

**Output:**



**Conclusion:**

Hence we have successfully implemented Token Ring algorithm using java.