**Aim –** To implement token ring based mutual exclusion algorithm.

**Name –** Yogita Sunil Girigosavi

**Tokenring.java**

import java.util.\*;

public class Tokenring

{

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

    {

        Scanner scan = new Scanner(System.in);

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

        int n = scan.nextInt();

        int m = n - 1;

        // Decides the number of nodes forming the ring

        int token = 0;

        int ch = 0, flag = 0;

        int criticalNode = -1; // variable to track which node enters the critical section

        for (int i = 0; i < n; i++)

        {

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

        }

        System.out.println(" " + 0);

        do

        {

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

            int s = scan.nextInt();

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

            int r = scan.nextInt();

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

            int a;

            a = scan.nextInt();

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

            for (int i = token, j = token; (i % n) != s; i++, j = (j + 1) % n)

            {

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

            }

            System.out.println(" " + s);

            if (criticalNode == -1)

            { // check if the critical section is empty

                System.out.println("Sender " + s + " enters the critical section");

                criticalNode = s; // assign the critical section to the sender

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

                {

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

                }

                System.out.println("Receiver " + r + " received data: " + a +"\n");

                criticalNode = -1; // release the critical section

            } else

            {

                System.out.println("Sender " + s + " cannot enter the critical section. It is already occupied by Node " + criticalNode);

            }

            System.out.println();

            System.out.println("Critical section is now free");

            token = s;

            System.out.println("Press 1 to continue or any other key to exit");

            ch = scan.nextInt();

        } while( ch == 1 );

    }

}

**Output -**

