# Department of Computing

**CS332: Distributed Computing**

**Class: BSCS – 6 & BESE – 7**

# Lab 9: Leader Election

**Date: 2nd December, 2019**

**Time: 09:00 AM – 12:00 PM**

# Instructor: Mr. Suleman Awan

**Lab Engineer: Mr. Ahsan Gul**

**Name: Ahmad Amjad Mughal**

**Reg No: 121672**

**Class: BSCS-6C**

**Lab-Task**

Notice that the process with the greatest identifier is elected. Create a single program that implements this algorithm. Then, instantiate it many times with different process placements i.e. increasing order, decreasing order and random and test how it works in terms of number of messages required in leader selection. You'll need to make some provision for initially assigning each instance a unique identifier.

Note: You can use circular array/ link list

**Solution**

//Ahmad Amjad Mughal 121672 BSCS-6C

**public class Algorithm**

{

//declaring static variables

static int nodes = 6;

public static Node[] arr=new Node[nodes];

public static int ite=0;

//initializing Array of Nodes with values in increasing order

public static void main(String args[])

{

int current;

int next;

System.out.println("------------Initializng Ring of Nodes--------");

for(int i = 0 ;i < nodes ; i++)

{

if(i!= nodes - 1)

{

Node temp = new Node(i,i+1);

arr[i]=temp;

System.out.println("Node = " + temp.getValue());

}

else

//End of a ring where Node 5 is linked with Node 0

{

Node temp = new Node(i,0);

arr[i]=temp;

System.out.println("Node = " + temp.getValue());

}

}

//Getting the value of current and next node and compare them

for(int i = 0 ; i < nodes ; i++)

{

if(i != nodes - 1)

{

current = arr[i].getValue();

next = arr[i+1].getValue();

//if next node has higher value then simply move to next iteration

if(current < next)

{

System.out.println("No Displacement Required for "+ arr[i].getValue() +" and " +arr[i+1].getValue());

ite++ ;

continue;

}

//If current node has higher value then there's displacement of values among them

else

{

int temp = arr[i+1].getValue();

int v = arr[i].getValue();

arr[i+1].setValue(v);

v = temp;

arr[i].setValue(v);

}

}

//it checks for node 5 to make sure it is eligible for leader

else

{

current=arr[i].getValue();

next=arr[0].getValue();

if( current < next) {

continue;

}

else

{

int greater = current;

for(int j=0;j < nodes;j++)

{

int ch=arr[j].getValue();

ite++;

if(ch==greater) {

System.out.println("Leader found. Node = "+Integer.toString(j));

System.out.println("Value of Leader Node = "+ arr[j].getValue());

}

}

}

}

}

}

}

//defined class of Node to define its attributes and functions

**public class Node**

{

private int value;

private int next;

public Node(int v, int n){

this.value=v;

this.next=n;

}

public int getValue() {

return value;

}

public void setValue(int v) {

this.value=v;

}

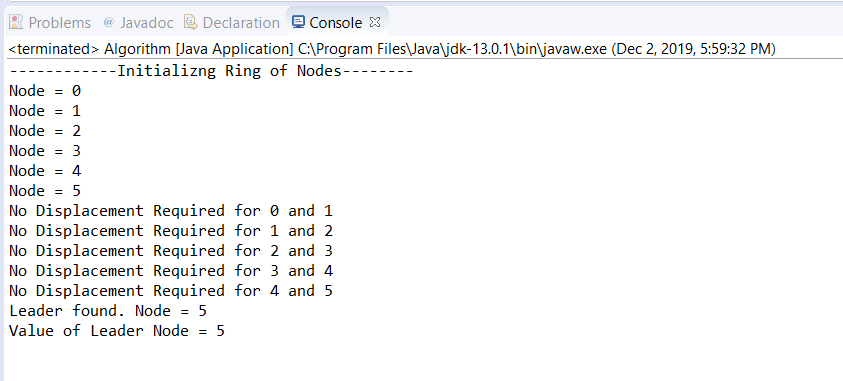
public int getNext() {

return next;

}

}

**Screenshot**

****