**Aim –** To implement Bully and Ring algorithm for leader election.

**Name –** Yogita Sunil Girigosavi

**BullyAlgoExample2.java**

import java.util.Scanner;

// create process class for creating a process having id and status

class Process

{

    // declare variables

    public int id;

    public String status;

    // initialize variables using constructor

    public Process(int id)

    {

        this.id = id;

        this.status = "active";

    }

}

// create class BullyAlgoExample2 for understanding the concept of Bully algorithm

public class BullyAlgoExample2

{

    // initialize variables and array

    Scanner sc;

    Process[] processes;

    int n;

    // initialize Scanner class object in constructor

    public BullyAlgoExample2()

    {

        sc= new Scanner(System.in);

    }

    // create ring() method for initializing the ring

    public void ring()

    {

        // get input from the user for processes

        System.out.println("Enter total number of processes of Processes");

        n = sc.nextInt();

        // initialize processes array

        processes = new Process[n];

        for(int i = 0; i< n; i++)

        {

            processes[i]= new Process(i);

        }

    }

    // create election() method for electing process

    public void performElection()

    {

        // we use the sleep() method to stop the execution of the current thread

        try

        {

            Thread.sleep(1000);

        }

        catch (InterruptedException e)

        {

            e.printStackTrace();

        }

        // show failed process

        System.out.println("Process having id "+processes[getMaxValue()].id+" fails");

        // change status to Inactive of the failed process

        processes[getMaxValue()].status = "Inactive";

        // declare and initialize variables

        int idOfInitiator = 0;

        boolean overStatus = true;

        // use while loop to repeat steps

        while(overStatus)

        {

            boolean higherProcesses = false;

            // iterate all the processes

            for(int i = idOfInitiator + 1; i< n; i++)

            {

                if(processes[i].status == "active")

                {

                    System.out.println("Process "+idOfInitiator+" passes election message to process " +i);

                    higherProcesses = true;

                }

            }

            // check for higher process

            if(higherProcesses)

            {

                // use for loop to again iterate processes

                for(int i = idOfInitiator + 1; i< n; i++)

                {

                    if(processes[i].status == "active")

                    {

                        System.out.println("Process "+i+" passes Ok message to process " +idOfInitiator);

                    }

                }

                // increment initiator id

                idOfInitiator++;

            }

            else

            {

                // get the last process from the processes that will become coordinator

                int coord = processes[getMaxValue()].id;

                // show process that becomes the coordinator

                System.out.println("Finally Process "+coord+" Becomes Coordinator");

                for(int i = coord - 1; i>= 0; i--)

                {

                    if(processes[i].status == "active")

                    {

                        System.out.println("Process "+coord+" passes coordinator message to process " +i);

                    }

                }

                System.out.println("End of Election");

                overStatus = false;

                break;

            }

        }

    }

    // create getMaxValue() method that returns index of max process

    public int getMaxValue()

    {

        int mxId = -99;

        int mxIdIndex = 0;

        for(int i = 0; i<processes.length; i++)

        {

            if(processes[i].status == "active" && processes[i].id >mxId)

            {

                mxId = processes[i].id;

                mxIdIndex = i;

            }

        }

        return mxIdIndex;

    }

    // main() method start

    public static void main(String[] args)

    {

        // create instance of the BullyAlgoExample2 class

        BullyAlgoExample2 bully = new BullyAlgoExample2();

        // call ring() and performElection() method

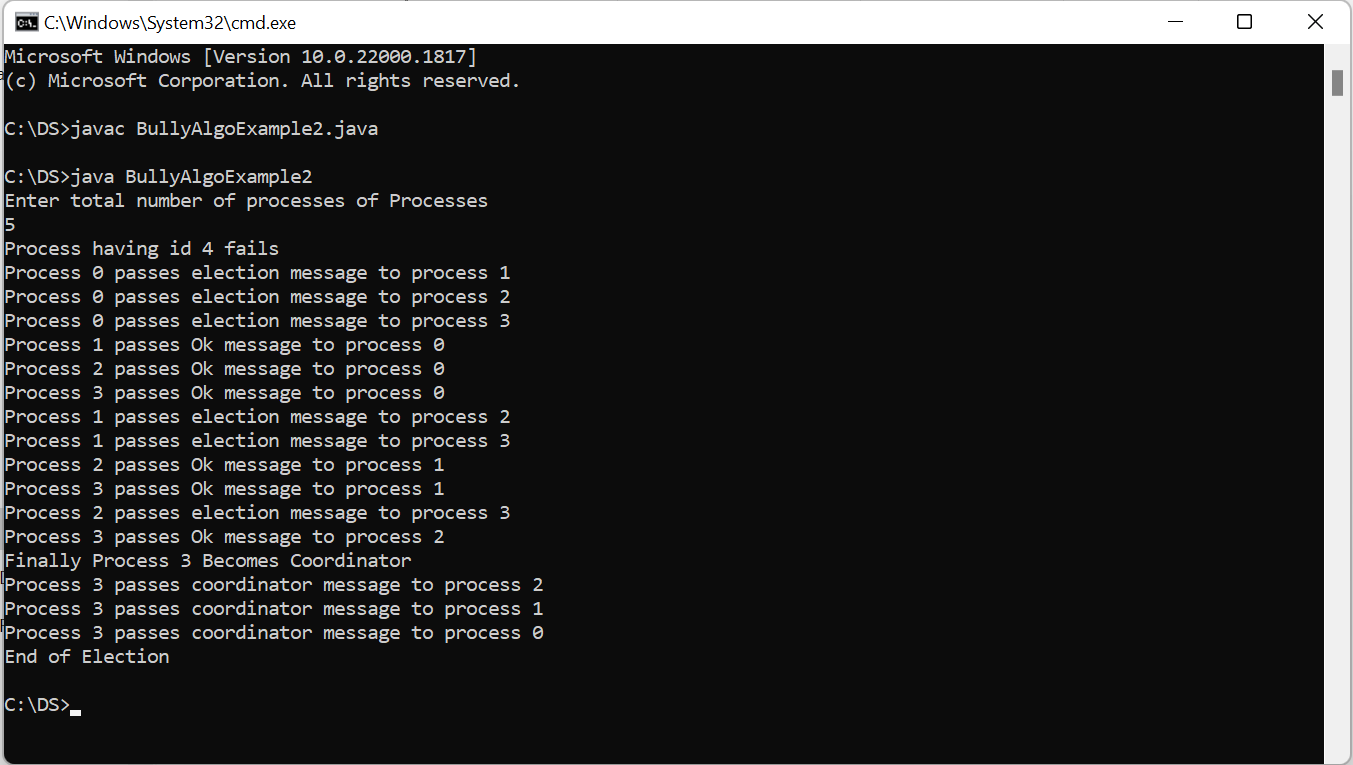
        bully.ring();

        bully.performElection();

    }

}

**Output –**



**Ring.java**

import java.util.Scanner;

public class Ring

{

    class Pro

    {

        int id;

        boolean act;

        Pro(int id)

        {

            this.id = id;

            act = true;

        }

    }

    int TotalProcess;

    Pro[] process;

    public Ring() { }

    public void initialiseRing()

    {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the number of processes: ");

        TotalProcess = scanner.nextInt();

        process = new Pro[TotalProcess];

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

        {

            process[i] = new Pro(i);

        }

    }

    public void Election() {

        System.out.println("Process no " + process[FetchMaximum()].id + " fails");

        process[FetchMaximum()].act = false;

        System.out.println("Election Initiated by 2");

        int initializedProcess = 2;

        int old = initializedProcess;

        int newer = old + 1;

        while (true)

        {

            if (process[newer].act)

            {

                System.out.println("Process " + process[old].id + " pass election message to " + process[newer].id);

                old = newer;

            }

            newer = (newer + 1) % TotalProcess;

            if (newer == initializedProcess)

            {

                break;

            }

        }

        System.out.println("Process " + process[FetchMaximum()].id + " becomes coordinator");

        int coord = process[FetchMaximum()].id;

        old = coord;

        newer = (old + 1) % TotalProcess;

        while (true) {

            if (process[newer].act)

            {

                System.out.println("Process " + process[old].id + " pass Coordinator message to process " + process[newer].id);

                old = newer;

            }

            newer = (newer + 1) % TotalProcess;

            if (newer == coord)

            {

                System.out.println("End Of Election ");

                break;

            }

        }

    }

    public int FetchMaximum()

    {

        int Ind = 0;

        int maxId = -9999;

        for (int i = 0; i < process.length; i++)

        {

            if (process[i].act && process[i].id > maxId)

            {

                maxId = process[i].id;

                Ind = i;

            }

        }

        return Ind;

    }

    public static void main(String arg[])

    {

        Ring object = new Ring();

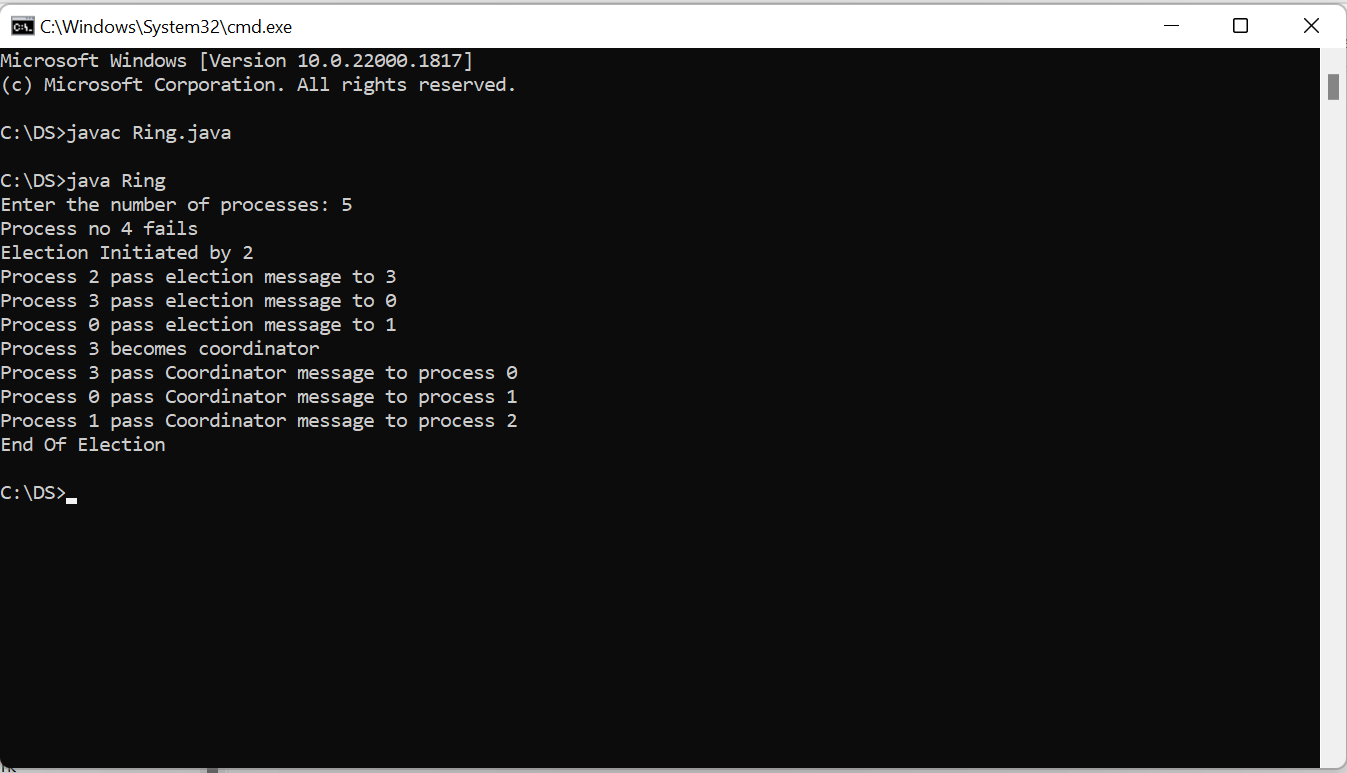
        object.initialiseRing();

        object.Election();

    }

}

**Output –**

****