Title: Implementation of election algorithm

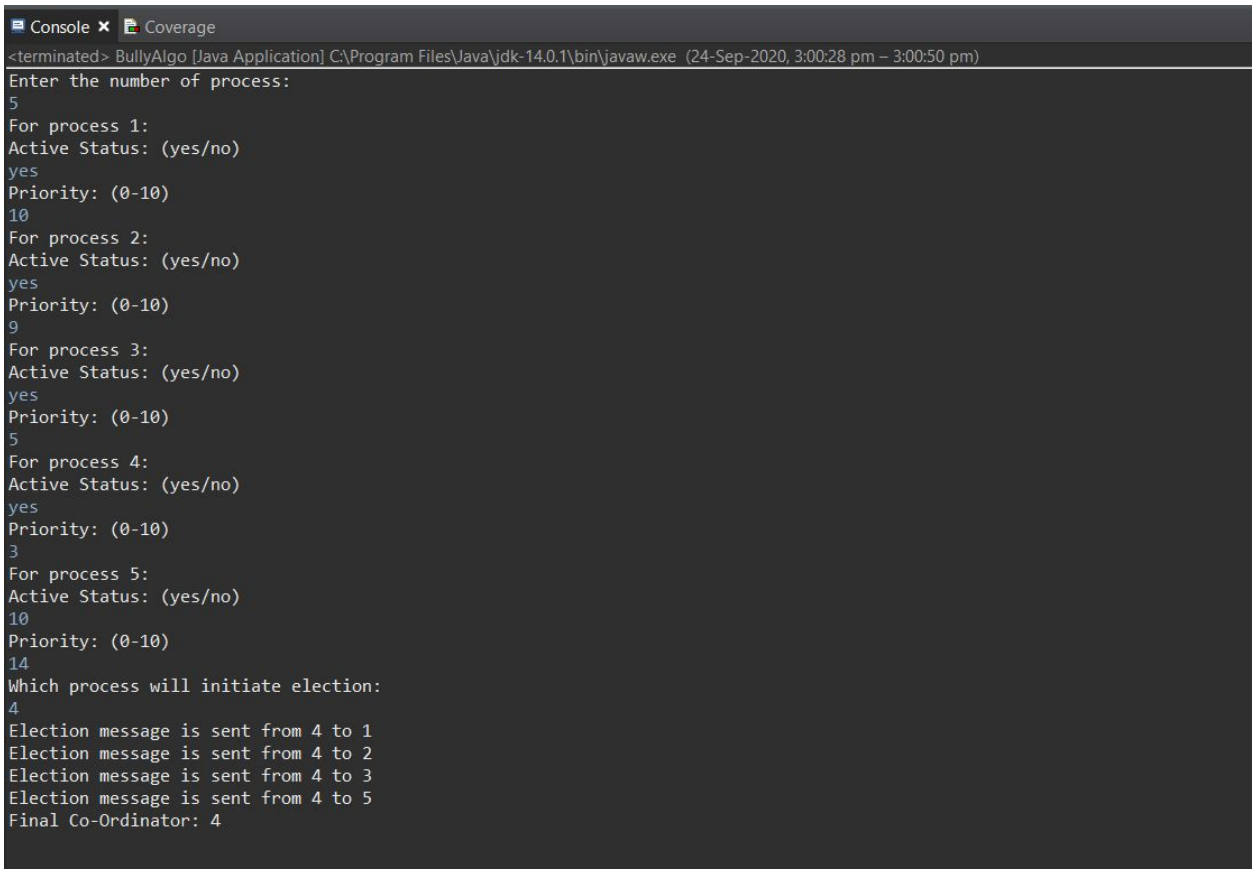
1) Bully Algorithm

Code:

```
package lab_03;

import java.io.*;
import java.util.ArrayList;
import java.util.Scanner;

public class BullyAlgo {
	static int n, co;
	static ArrayList<Integer> process = new ArrayList<Integer>();
	static ArrayList<String> status = new ArrayList<String>();

	BullyAlgo() {
		n = 0;
		co = 0;
	}

	static void findCoordinator(int x) {
		x = x - 1;
		co = x + 1;
		for (int i = 0; i < n; i++) {
			if (process.get(x) < process.get(i)) {
				System.out.println("Election message is sent from " +
(x + 1) + " to " + (i + 1));
				if (status.get(i) == "yes")
					findCoordinator(i + 1);
			}
		}
	}

	public static void main(String args[]) throws IOException {
		System.out.println("Enter the number of process:");
		Scanner sc = new Scanner(System.in);
		n = sc.nextInt();

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

```
                System.out.println("For process " + (i + 1) + ":");
                System.out.println("Active Status: (yes/no)");
                status.add(sc.next());
                System.out.println("Priority: (0-10)");
                process.add(sc.nextInt());
            }
            System.out.println("Which process will initiate election:");
            findCoordinator(sc.nextInt());
            System.out.println("Final Co-Ordinator: " + co);
            sc.close();
        }

    }
```

Screenshot:



2) Ring Algorithm:

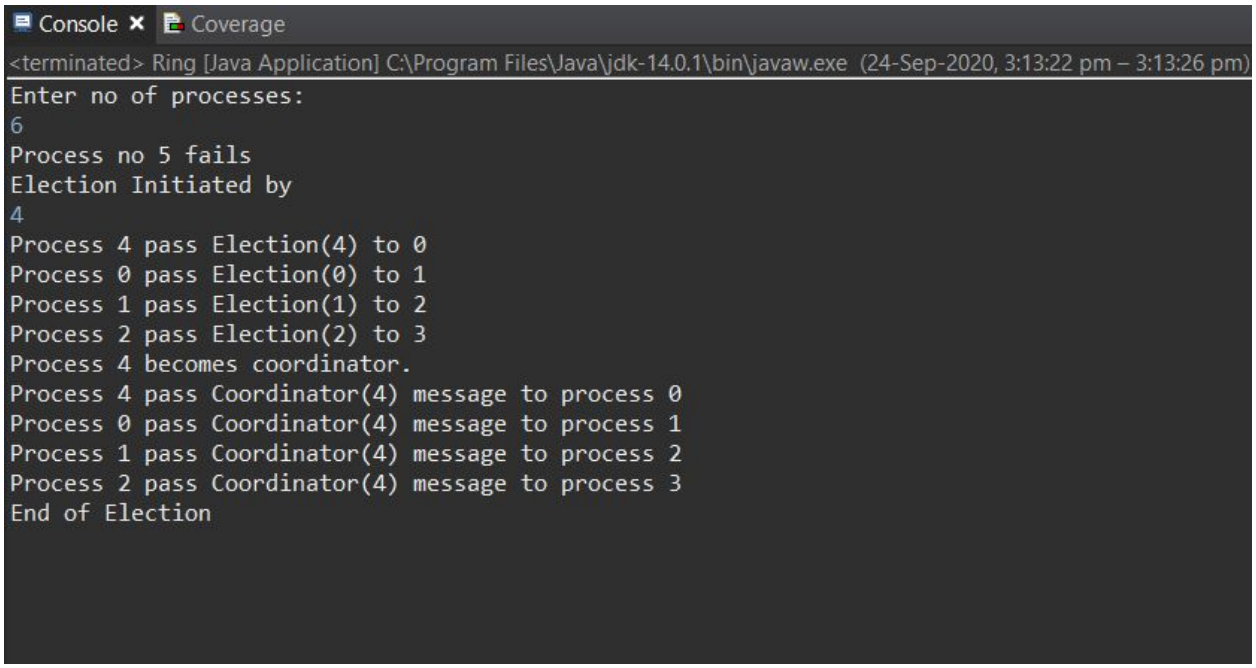
Code:

```
package lab_03;

import java.util.Scanner;

class Process {
	public int id;
	public boolean active;

	public Process(int id) {
		this.id = id;
		active = true;
	}

}

public class Ring {
	int noOfProcesses;
	Process[] processes;
	Scanner sc;

	public Ring() {
		sc = new Scanner(System.in);
	}

	public void initialiseRing() {
		System.out.println("Enter no of processes:");
		noOfProcesses = sc.nextInt();
		processes = new Process[noOfProcesses];
		for (int i = 0; i < processes.length; i++) {
			processes[i] = new Process(i);
		}
	}

	public int getMax() {
		int maxId = -99;
		int maxIdIndex = 0;
		for (int i = 0; i < processes.length; i++) {
```

```
                        if (processes[i].active && processes[i].id > maxId) {
                                maxId = processes[i].id;
                                maxIdIndex = i;

                        }
                }
                return maxIdIndex;
        }

        public void performElection() {

                System.out.println("Process no " + processes[getMax()].id + "
fails");
                processes[getMax()].active = false;
                System.out.println("Election Initiated by");
                int initiatorProcesss = sc.nextInt();

                int prev = initiatorProcesss;
                int next = prev + 1;

                while (true) {
                        if (processes[next].active) {
                                System.out.println("Process " + processes[prev].id + "
pass Election(" + processes[prev].id + ") to "
                                                + processes[next].id);
                                prev = next;
                        }

                        next = (next + 1) % noOfProcesses;
                        if (next == initiatorProcesss)

                        {
                                break;
                        }
                }

                System.out.println("Process " + processes[getMax()].id + "
becomes coordinator.");
                int coordinator = processes[getMax()].id;
```

```
                prev = coordinator;
                next = (prev + 1) % noOfProcesses;

                while (true) {

                        if (processes[next].active) {
                                System.out.println("Process " + processes[prev].id + "
pass Coordinator(" + coordinator
                                                + ") message to process " +
processes[next].id);
                                prev = next;
                        }
                        next = (next + 1) % noOfProcesses;
                        if (next == coordinator)

                        {
                                System.out.println("End of Election ");
                                break;
                        }
                }

        }

        public static void main(String arg[]) {
                Ring r = new Ring();
                r.initialiseRing();
                r.performElection();
        }

}
```

Screenshot:


Console
Coverage
<terminated> Ring [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (24-Sep-2020, 3:13:22 pm – 3:13:26 pm)
Enter no of processes:
6
Process no 5 fails
Election Initiated by
4
Process 4 pass Election(4) to 0
Process 0 pass Election(0) to 1
Process 1 pass Election(1) to 2
Process 2 pass Election(2) to 3
Process 4 becomes coordinator.
Process 4 pass Coordinator(4) message to process 0
Process 0 pass Coordinator(4) message to process 1
Process 1 pass Coordinator(4) message to process 2
Process 2 pass Coordinator(4) message to process 3
End of Election