Title: Implementation of election algorithm

1) Bully Algorithm

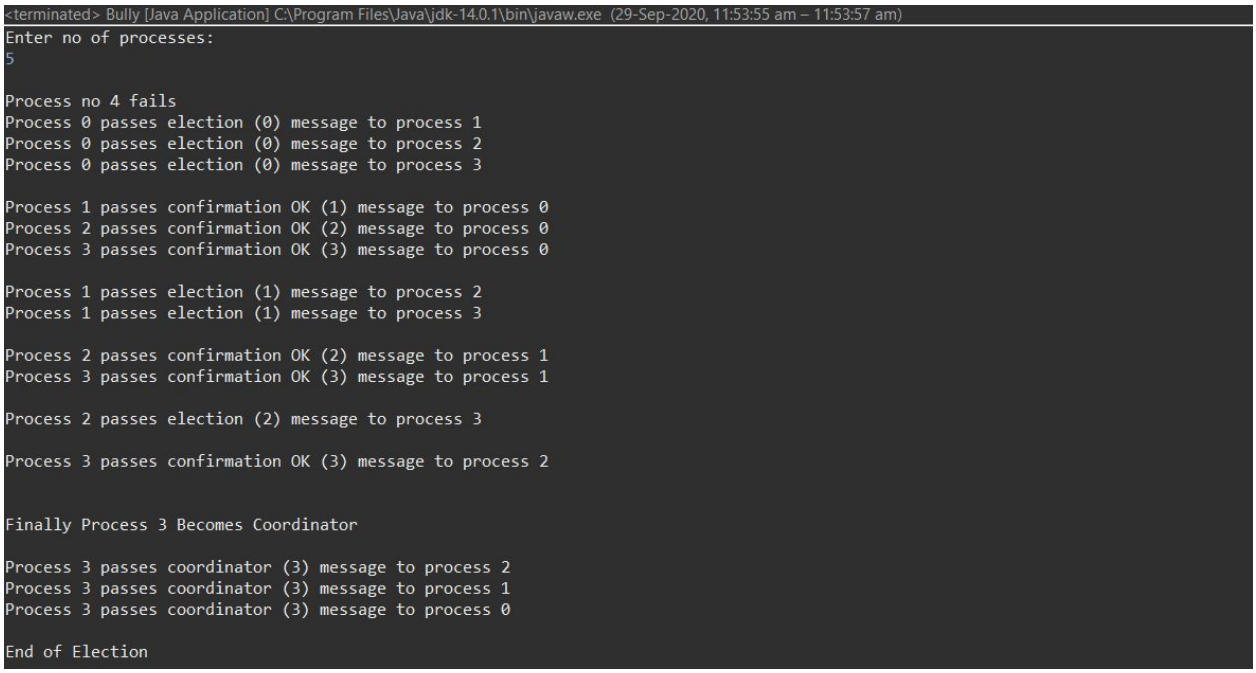
Code:

```
package lab_05;

import java.util.Scanner;

class Process {
	public int id;
	public boolean active;

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

public class Bully {
	Scanner sc;
	Process[] processes;
	int noOfProcess;

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

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

	public void performElection() {

		try {
```

```
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            System.out.println("\nProcess no " + processes[getMax()].id + "
fails");
            processes[getMax()].active = false;

            int InitiatorProcessId = 0;
            boolean notOver = true;
            while (notOver) {

                boolean moreHigherProcesses = false;
                for (int i = InitiatorProcessId + 1; i < noOfProcess; i++) {
                    if (processes[i].active) {
                        System.out.println("Process " +
InitiatorProcessId + " passes election (" + InitiatorProcessId
                                + ") message to process " + i);
                        moreHigherProcesses = true;

                    }
                }
                System.out.println();
                if (moreHigherProcesses) {

                    for (int i = InitiatorProcessId + 1; i < noOfProcess; i++)
{
                        if (processes[i].active) {
                            System.out.println("Process " + i + "
passes confirmation OK (" + i + ") message to process "
                                    + InitiatorProcessId);
                        }

                    }
                    InitiatorProcessId++;
                    System.out.println();
                }
```

```
                        else {
                                int coordinator = processes[getMax()].id;
                                System.out.println("Finally Process " + coordinator + "
Becomes Coordinator\n");
                                for (int i = coordinator - 1; i >= 0; i--) {
                                        if (processes[i].active) {
                                                System.out.println("Process " +
coordinator + " passes coordinator (" + coordinator
                                                                + ") message to process "
+ i);
                                        }
                                }

                                System.out.println("\nEnd of Election");
                                notOver = false;
                                break;
                        }
                }

        }

        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 static void main(String[] args) {
                Bully b = new Bully();
                b.initialiseBully();
                b.performElection();
        }
}
```

Screenshot:



2) Ring Algorithm:
Code:

```
package lab_05;

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 initiatorProcess = sc.nextInt();

		int prev = initiatorProcess;;
```

```
                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 == initiatorProcess)

                        {
                                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: