Banking System

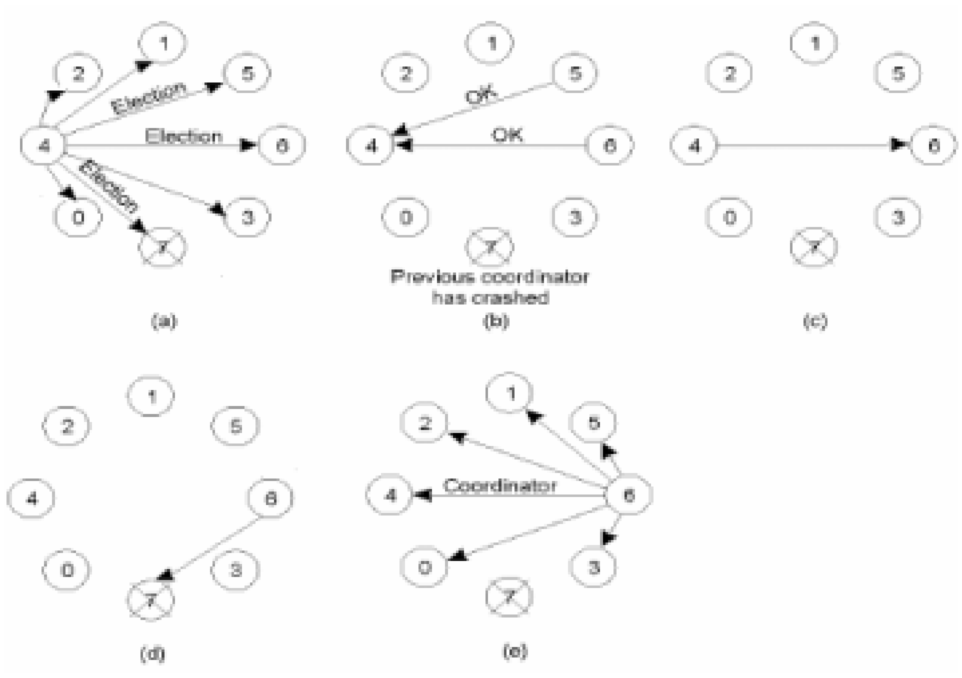
(Experiment 6 - Election Algorithm)

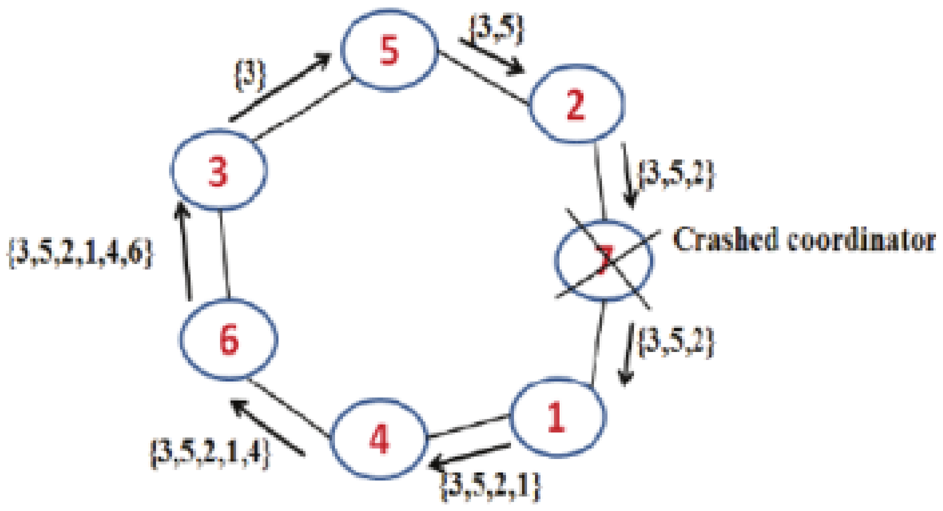
Yash Brid 2019130008
Abhishek Chopra 2019130009
Sumeet Haldipur 2019130018

Aim
To understand and implement election algorithms.

Objective
- To implement Bully algorithm for Election in a distributed system
- To implement Ring algorithm for Election in a distributed system
- To compare both the algorithms

Theory
Election algorithms choose a process from a group of processors to act as a coordinator. If the coordinator process crashes due to some reasons, then a new coordinator is elected on another processor. Election algorithm basically determines where a new copy of the coordinator should be restarted. Election algorithms assume that every active process in the system has a unique priority number. The process with highest priority will be chosen as a new coordinator. Hence, when a coordinator fails, this algorithm elects that active process which has the highest priority number.Then this number is sent to every active process in the distributed system.



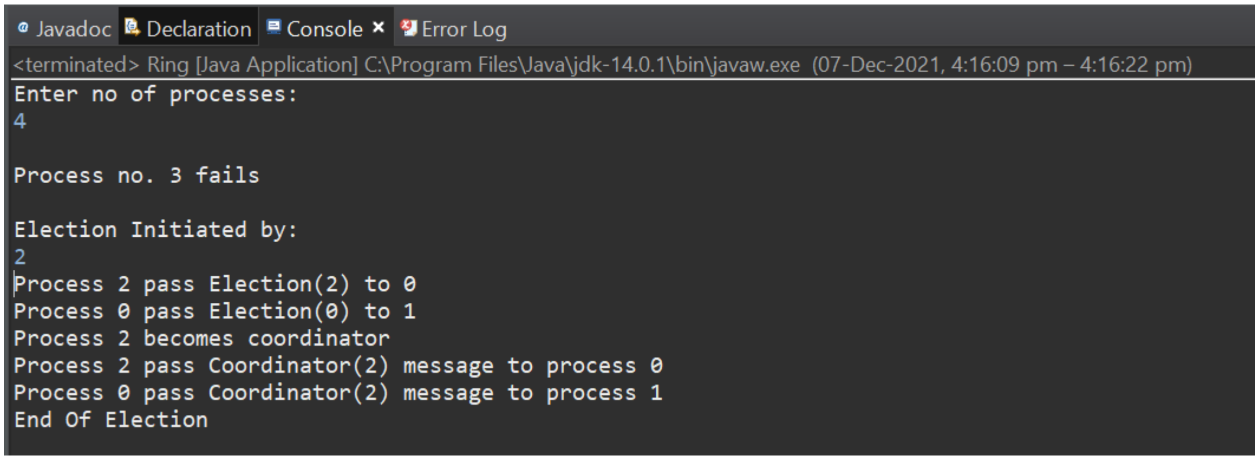


Code

Bully.java

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

class Bully {
    static int n;
    static int priority[] = new int[100];
    static int status[] = new int[100];
    static int co;

    public static void main(String args[]) throws IOException {
        System.out.println("Enter the number of process");
        Scanner in = new Scanner(System.in);
        n = in.nextInt();
        int i;
        for (i = 0; i < n; i++) {
            System.out.println("For process " + (i + 1) + ":");
            System.out.println("Status:");
            status[i] = in.nextInt();
            System.out.println("Priority");
            priority[i] = in.nextInt();
        }
        System.out.println("Which process will initiate election?");
        int ele = in.nextInt();
        elect(ele);
        System.out.println("Final coordinator is " + co);
    }
    static void elect(int ele) {
        ele = ele - 1;
        co = ele + 1;
        for (int i = 0; i < n; i++) {
            if (priority[ele] < priority[i]) {
                System.out.println("Election message is sent from " + (ele + 1)
+ " to " + (i + 1));
                if (status[i] == 1)
                    elect(i + 1);
            }
        }
    }
}
```

Output



Javadoc Declaration Console Error Log

<terminated> Ring [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe (07-Dec-2021, 4:16:09 pm – 4:16:22 pm)

```
Enter no of processes:
4

Process no. 3 fails

Election Initiated by:
2
Process 2 pass Election(2) to 0
Process 0 pass Election(0) to 1
Process 2 becomes coordinator
Process 2 pass Coordinator(2) message to process 0
Process 0 pass Coordinator(2) message to process 1
End Of Election
```

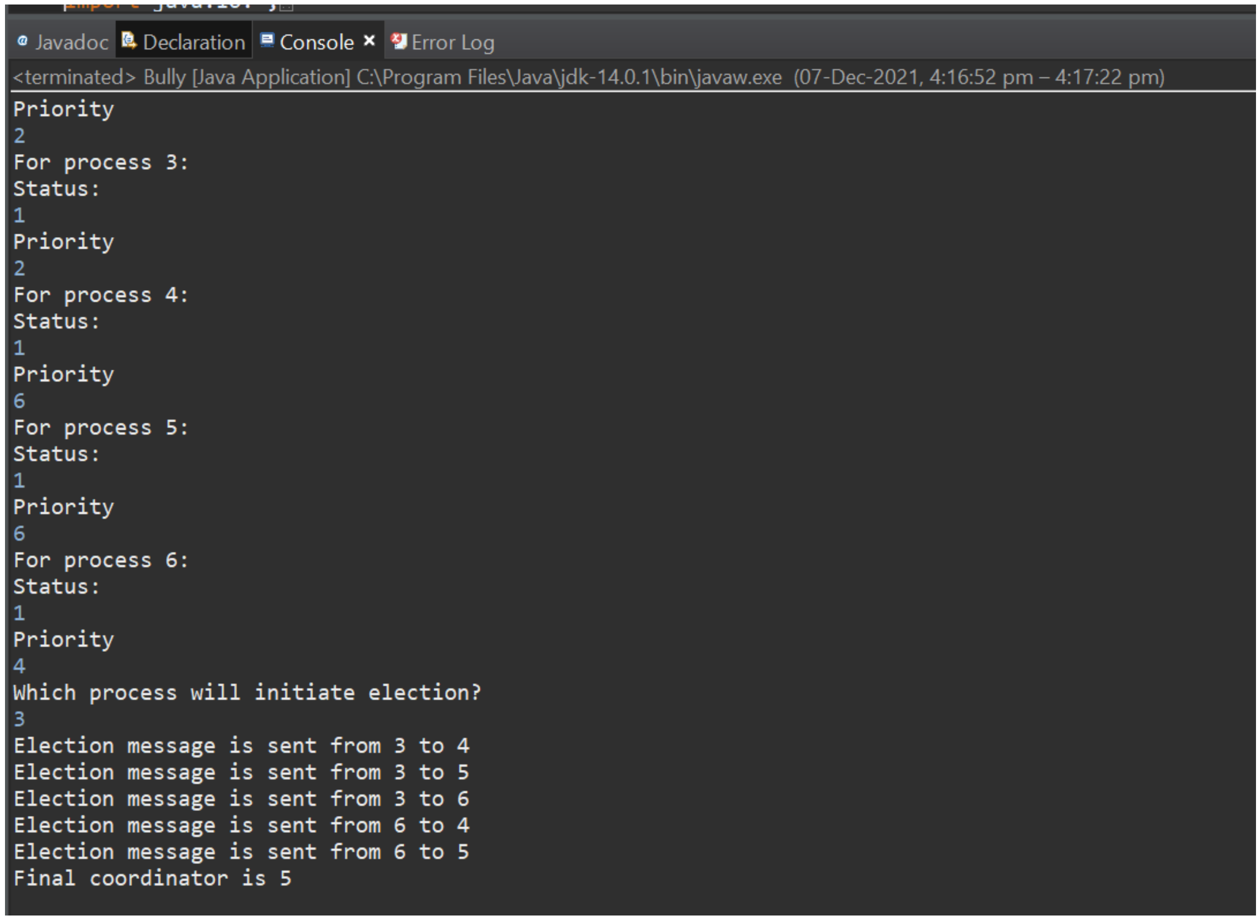
Ring.java

```
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;
```

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

Output



```
<terminated> Bully [Java Application] C:\Program Files\Java\jdk-14.0.1\bin\javaw.exe  (07-Dec-2021, 4:16:52 pm – 4:17:22 pm)
Priority
2
For process 3:
Status:
1
Priority
2
For process 4:
Status:
1
Priority
6
For process 5:
Status:
1
Priority
6
For process 6:
Status:
1
Priority
4
Which process will initiate election?
3
Election message is sent from 3 to 4
Election message is sent from 3 to 5
Election message is sent from 3 to 6
Election message is sent from 6 to 4
Election message is sent from 6 to 5
Final coordinator is 5
```

Conclusion

In this experiment, we understood about the election algorithm in a distributed system for selecting a coordinator in a distributed system. We understood and implemented the Bully and Ring algorithm in Java while comparing and understanding their drawbacks and advantages.