Code:

```
package com.muthadevs;

import java.util.ArrayList;
import java.util.Collections;
import java.util.Scanner;

public class Main{

   public static void main(String[] args) throws InterruptedException{

      Scanner sc1 = new Scanner(System.in);
      int rootChoice;

      while(true){
         System.out.println(
               "|----|-------------------------|"+
                  "\n| No |        Menu          |"+
                  "\n|----|-------------------------|"+
                  "\n| 1] | Ring Election Algorithm  |"+
                  "\n| 2] | Bully Election Algorithm |"+
                  "\n| 3] | Terminate              |"+
                  "\n|----|-------------------------|"+
                  "\nEnter your choice : ");rootChoice = sc1.nextInt();

         switch (rootChoice){
            case 1:
               //Ring Code Here

               int temp, i, j;

               Ring_Process[] proc = new Ring_Process[10];

               //object initialisation
               for (i = 0; i < proc.length; i++)
                  proc[i] = new Ring_Process();

               //scanner used for getting input from console
               Scanner in = new Scanner(System.in);
               System.out.print("Enter the number of process : ");
               int numOfProcesses = in.nextInt();
               System.out.println();

               // getting input from users
               for (i = 0; i < numOfProcesses; i++) {
                  proc[i].index = i;
                  System.out.print("Enter the id of process "+(i)+" : ");
                  proc[i].id = in.nextInt();
                  System.out.println("");
                  proc[i].state = "active";
               }
```

```
            // sorting the processes on the basis of id
            for (i = 0; i < numOfProcesses - 1; i++) {
               for (j = 0; j < numOfProcesses - 1; j++) {
                  if (proc[j].id > proc[j + 1].id) {
                     temp = proc[j].id;
                     proc[j].id = proc[j + 1].id;
                     proc[j + 1].id = temp;
                  }
               }
            }

            for (i = 0; i < numOfProcesses; i++) {
               System.out.print("["+i+"]"+proc[i].id+" ");
            }

            int init,ch,temp1,temp2,max = -1;
            int[] active_list = new int[10];

            System.out.println("\nProcess " + proc[numOfProcesses - 1].id + " selected as co-ordinator");

            max=proc[numOfProcesses-1].id;

            ch = 1;

            while (ch!=5) {
               for(int k : active_list){
                  active_list[k] = 0;
               }
               System.out.print
                     ("|----|---------------------------|\n" +
                           "| No |         MENU          |" +
                           "\n|----|---------------------------|"+
                           "\n| 1] | Crash Process           |" +
                           "\n| 2] | Conduct Election        |" +
                           "\n| 3] | Print co-ordinator      |" +
                           "\n| 4] | Send MSG to co-ordinator  |" +
                           "\n| 5] | Terminate from here     |" +
                           "\n|----|---------------------------|" +
                           "\nEnter Your Choice : ");
               ch = in.nextInt();

               switch (ch) {
                  case 1:
                     System.out.print("Enter Process Number to Crash : ");
                     int p=in.nextInt();
                     System.out.println();
                     if(p==proc[numOfProcesses-1].id)
                     {
                        proc[numOfProcesses - 1].state = "inactive";
                        System.out.println("Process "+proc[numOfProcesses - 1].id+" is crashed\nPerform
Election to choose new coordinator !");
                     }
                     else
```

```
        {
            proc[p-1].state="inactive";
            System.out.println("Process "+p+" is crashed");
        }
        max = -1;
        break;

    case 2:
        System.out.print("\nEnter the Process number who initialised election : ");
        init = in.nextInt();
        temp2 = init;
        temp1 = init + 1;

        i = 0;
        max = -1;

        while (temp2 != temp1) {
            if ("active".equals(proc[temp1].state)) {

                System.out.print("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);
                active_list[i] = proc[init].id;
                System.out.println("");
                for(int t=0;t< active_list.length;t++){
                    System.out.print((active_list[t]!=0)?active_list[t]+" ":"");
                }
                init = temp1;
                i++;

            }
            if (temp1 == numOfProcesses) {
                temp1 = 0;
            } else {
                temp1++;
            }
        }

        System.out.println("\nProcess " + proc[init].id + " send message to " + proc[temp1].id);
        active_list[i] = proc[init].id;
        for(int t=0;t< active_list.length;t++){
            System.out.print((active_list[t]!=0)?active_list[t]+" ":"");
        }
        i++;

        // finding maximum for co-ordinator selection
        for (j = 0; j < i; j++) {
            if (max < active_list[j]) {
                max = active_list[j];
            }
        }

        //Co-ordinator is found then printing on console
        System.out.println("\nProcess " + max + " selected as coordinator");
```

```
                for (i = 0; i < numOfProcesses; i++) {

                    if (proc[i].id == max) {
                        proc[i].state = "inactive";
                    }
                }
                break;
            case 3:
                if (max != -1) {
                    System.out.print("Current Co-ordinator : Process with id " +max+"\n");
                } else {
                    System.out.print("No Co-ordinator present\n");
                }
                break;
            case 4:
                if(max != -1){
                    System.out.print("Enter the process id which will send msg to co-ordinator : ");
                    int p_id = in.nextInt();
                    boolean flag=false,flag2=false;
                    for(int k=0;k<proc.length;k++){
                        if(p_id == proc[k].id){
                            flag=true;
                            if(proc[k].state.equalsIgnoreCase("active")){
                                flag2=true;
                            }
                        }
                    }
                    if (!flag){
                        System.out.println("No Such process found !");
                        break;
                    }
                    if(!flag2){
                        System.out.println("The process is INACTIVE !");
                        break;
                    }
                    System.out.print("Response from co-ordinator (i.e process id "+max+") received\n");
                }else {
                    System.out.println("No co-ordinator present, please initiate the election");
                }
                break;
            case 5:
                System.out.println("Program terminated ...");
                break;
            default:
                System.out.println("\nInvalid response \n");
                break;
        }

    }

    break;
case 2:
    //Bully Code Here
```

```
        ArrayList<Process> processes = new ArrayList<>();
        Scanner sc=new Scanner(System.in);
        Process currentCoo, p;

        System.out.print("Enter a number of processes: ");
        int n = sc.nextInt();
        System.out.println(String.format("Enter a %d priorities: ",n));
        for(int q = 0; q < n; q++)
          processes.add(new Process(q, (int) (Math.random()*(11)), sc.nextInt()));

        for(int i2 = 0; i2<processes.size(); i2++)
          System.out.println(processes.get(i2));

        Collections.sort(processes);
        currentCoo = processes.get(0);
        p = processes.get((int)(Math.random()*(processes.size()-1)+1));


        boolean cooChanged = false;

        while(processes.size() > 1) {
          if (cooChanged || p == currentCoo) {
            p = processes.get((int) (Math.random() * processes.size()));
            cooChanged = false;
          }

          System.out.print("Current Process (ID): ");
          System.out.println(p.getID());
          System.out.print("Current Co-ordinator (ID): ");
          System.out.println(currentCoo.getID());
          System.out.println(String.format("Process %d sent message to Co-ordinator %d", p.getID(),
currentCoo.getID()));


          if (p.getTimeout() >= currentCoo.getTimeout()) {
            System.out.println(String.format("Current Co-ordinator with ID %d is responding",
currentCoo.getID()));
          } else {
            System.out.println(String.format("Co-ordinator %d Didn't respond.", currentCoo.getID()));
            System.out.println("Election Started!");
            if (p != currentCoo) {
              processes.remove(processes.indexOf(currentCoo));
              cooChanged = true;
            }


            int processIndex = processes.indexOf(p);
            updatetimeouts(processes);
            p = processes.get(processIndex);

            int mintime = p.getTimeout();
            boolean flag = false;
```

```
System.out.print("Sending message to Priorities higher than ");
System.out.println(p.getPriority());

for (int z = 0; z < processIndex; z++) {
   if (processes.get(z).getTimeout() < mintime) {
      currentCoo = processes.get(z);
      mintime = currentCoo.getTimeout();
      flag = true;
   }
}

if (!flag) {
   currentCoo = p;
   System.out.println("No one responded.");
} else {
   System.out.print("New Co-ordinator is (ID): ");
   System.out.println(currentCoo.getID());
   System.out.println("message conveyed to lower Processes.");
   System.out.println("Restarting Loop... \n\n");
   updatetimeouts(processes);

   continue;
}

int minIndex = processes.indexOf(currentCoo);
updatetimeouts(processes);
currentCoo = processes.get(minIndex);

System.out.print("Sending message to Priorities Lower than ");
System.out.println(currentCoo.getPriority());

int timeout = currentCoo.getTimeout();
ArrayList<Process> responses = new ArrayList<>();
ArrayList<Integer> responseIndices = new ArrayList<>();
for (int x = minIndex + 1; x < processes.size(); x++) {
   if (processes.get(x).getTimeout() < timeout) {
      responses.add(processes.get(x));
      responseIndices.add(x);
   }
}
updatetimeouts(processes);
currentCoo = processes.get(minIndex);

for (int h = 0; h < responseIndices.size(); h++) {
   responses.get(h).incrementTimeout(processes.get(responseIndices.get(h)).getTimeout());
}

int maxtimeout = 2 * currentCoo.getTimeout();

flag = false;
for (int v = 0; v < responses.size(); v++) {
   System.out.println(String.format("Process %d responded waiting for another reply",
```

```
responses.get(v).getID()));
                        if (responses.get(v).getTimeout() < maxtimeout) {
                            currentCoo = responses.get(v);
                            maxtimeout = responses.get(v).getTimeout();
                            flag = true;
                        }
                    }
                    if (flag) {
                        System.out.print("new Co-ordinator is (ID): ");
                        System.out.println(currentCoo.getID());
                    } else {
                        System.out.println("no one Responded!");
                    }
                }
                System.out.println("Restarting Loop... \n\n");
                updatetimeouts(processes);
            }

            break;
        case 3:
            System.out.println("Terminated...!");
            System.exit(0);
        default:
            System.out.println("Enter Valid Choice..!");
        }

    }

  }

  static void updatetimeouts(ArrayList<Process> p){
    for(int i = 0; i < p.size(); i++){
      p.get(i).setTimeout((int) (Math.random()*(11)));      }

  }
}
class Ring_Process {

  public int index;   // to store the index of process
  public int id;     // to store id of process
  String state;      // indicates whether process is in active or inactive state

}
class Process implements Comparable<Process>{
  private int mId;
  private int mTimeout;
  private int mPriority;
  Process(int id, int timeout, int prior){
    mId = id;
    mTimeout = timeout;
    mPriority = prior;
  }
```

```
  public int getID(){
    return mId;
  }

  public int getTimeout(){
    return mTimeout;
  }

  public int getPriority(){
    return mPriority;
  }

  public void setTimeout(int tm){
    mTimeout = tm;
  }

  public void incrementTimeout(int t){
    mTimeout += t;
  }

  @Override public String toString(){
    return ("ID: "+ Integer.toString(mId)+ "\tPriority: "+ Integer.toString(mPriority));//+ "\tTimeout: "+
Integer.toString(mTimeout);
  }

  @Override public int compareTo(Process p){
    return -1*(this.mPriority - ((Process)p).getPriority());
  }
}
```

Output:

(RING ALGORITHM OUTPUT)

```
|----|-------------------------|
| No |         Menu            |
|----|-------------------------|
| 1] | Ring Election Algorithm  |
| 2] | Bully Election Algorithm |
| 3] | Terminate                |
|----|-------------------------|
```

Enter your choice :

1

Enter the number of process : 5

Enter the id of process 0 : 4

Enter the id of process 1 : 1

Enter the id of process 2 : 3

Enter the id of process 3 : 5

Enter the id of process 4 : 2

[0]1 [1]2 [2]3 [3]4 [4]5

Process 5 selected as co-ordinator

|----|---------------------------|

| No | MENU |

|----|---------------------------|

| 1] | Crash Process |

| 2] | Conduct Election |

| 3] | Print co-ordinator |

| 4] | Send MSG to co-ordinator |

| 5] | Terminate from here |

|----|---------------------------|

```
Enter Your Choice : 3
Current Co-ordinator : Process with id 5
|----|--------------------------|
| No |         MENU             |
|----|--------------------------|
| 1] | Crash Process            |
| 2] | Conduct Election         |
| 3] | Print co-ordinator       |
| 4] | Send MSG to co-ordinator |
| 5] | Terminate from here      |
|----|--------------------------|
Enter Your Choice : 4
Enter the process id which will send msg to co-ordinator : 1
Response from co-ordinator (i.e process id 5) received
|----|--------------------------|
| No |         MENU             |
|----|--------------------------|
| 1] | Crash Process            |
| 2] | Conduct Election         |
| 3] | Print co-ordinator       |
| 4] | Send MSG to co-ordinator |
| 5] | Terminate from here      |
|----|--------------------------|
Enter Your Choice : 1
```

Enter Process Number to Crash : 5

Process 5 is crashed

Perform Election to choose new coordinator !

```
|----|--------------------------|
| No |         MENU             |
|----|--------------------------|
| 1] | Crash Process            |
| 2] | Conduct Election         |
| 3] | Print co-ordinator       |
| 4] | Send MSG to co-ordinator |
| 5] | Terminate from here      |
|----|--------------------------|
```

Enter Your Choice : 3

No Co-ordinator present

```
|----|--------------------------|
| No |         MENU             |
|----|--------------------------|
| 1] | Crash Process            |
| 2] | Conduct Election         |
| 3] | Print co-ordinator       |
| 4] | Send MSG to co-ordinator |
| 5] | Terminate from here      |
|----|--------------------------|
```

Enter Your Choice : 2

Enter the Process number who initialised election : 0

Process 1 send message to 2

1

Process 2 send message to 3

1 2

Process 3 send message to 4

1 2 3

Process 4 send message to 1

1 2 3 4

Process 4 selected as coordinator

|----|----------------------------|

| No | MENU |

|----|----------------------------|

| 1] | Crash Process |

| 2] | Conduct Election |

| 3] | Print co-ordinator |

| 4] | Send MSG to co-ordinator |

| 5] | Terminate from here |

|----|----------------------------|

Enter Your Choice : 3

Current Co-ordinator : Process with id 4

```
|----|--------------------------|
| No |          MENU            |
|----|--------------------------|
| 1] | Crash Process            |
| 2] | Conduct Election         |
| 3] | Print co-ordinator       |
| 4] | Send MSG to co-ordinator |
| 5] | Terminate from here      |
|----|--------------------------|
Enter Your Choice : 4
Enter the process id which will send msg to co-ordinator : 2
Response from co-ordinator (i.e process id 4) received
|----|--------------------------|
| No |          MENU            |
|----|--------------------------|
| 1] | Crash Process            |
| 2] | Conduct Election         |
| 3] | Print co-ordinator       |
| 4] | Send MSG to co-ordinator |
| 5] | Terminate from here      |
|----|--------------------------|
Enter Your Choice : 5
Program terminated ...
```

(BULLY ALGORITHM OUTPUT)

```
|----|-------------------------|
| No |         Menu            |
|----|-------------------------|
| 1] | Ring Election Algorithm  |
| 2] | Bully Election Algorithm |
| 3] | Terminate               |
|----|-------------------------|
```

Enter your choice :

2

Enter a number of processes: 6

Enter a 6 priorities:

5

6

3

2

1

4

ID: 0 Priority: 5

ID: 1 Priority: 6

ID: 2 Priority: 3

ID: 3 Priority: 2

ID: 4 Priority: 1

ID: 5 Priority: 4

Current Process (ID): 5

Current Co-ordinator (ID): 1

Process 5 sent message to Co-ordinator 1

Current Co-ordinator with ID 1 is responding

Restarting Loop...

Current Process (ID): 5

Current Co-ordinator (ID): 1

Process 5 sent message to Co-ordinator 1

Current Co-ordinator with ID 1 is responding

Restarting Loop...

Current Process (ID): 5

Current Co-ordinator (ID): 1

Process 5 sent message to Co-ordinator 1

Current Co-ordinator with ID 1 is responding

Restarting Loop...

Current Process (ID): 5

Current Co-ordinator (ID): 1

Process 5 sent message to Co-ordinator 1

Current Co-ordinator with ID 1 is responding

Restarting Loop...

Current Process (ID): 5

Current Co-ordinator (ID): 1

Process 5 sent message to Co-ordinator 1

Current Co-ordinator with ID 1 is responding

Restarting Loop...

Current Process (ID): 5

Current Co-ordinator (ID): 1

Process 5 sent message to Co-ordinator 1

Co-ordinator 1 Didn't respond.

Election Started!

Sending message to Priorities higher than 4

No one responded.

Sending message to Priorities Lower than 4

Process 2 responded waiting for another reply

no one Responded!

Restarting Loop...

Current Process (ID): 5

Current Co-ordinator (ID): 5

Process 5 sent message to Co-ordinator 5

Current Co-ordinator with ID 5 is responding

Restarting Loop...

Current Process (ID): 2

Current Co-ordinator (ID): 5

Process 2 sent message to Co-ordinator 5

Co-ordinator 5 Didn't respond.

Election Started!

Sending message to Priorities higher than 3

No one responded.

Sending message to Priorities Lower than 3

Process 3 responded waiting for another reply

Process 4 responded waiting for another reply

new Co-ordinator is (ID): 4

Restarting Loop...

Current Process (ID): 2

Current Co-ordinator (ID): 4

Process 2 sent message to Co-ordinator 4

Co-ordinator 4 Didn't respond.

Election Started!

Sending message to Priorities higher than 3

New Co-ordinator is (ID): 0

message conveyed to lower Processes.

Restarting Loop...

Current Process (ID): 2

Current Co-ordinator (ID): 0

Process 2 sent message to Co-ordinator 0

Current Co-ordinator with ID 0 is responding

Restarting Loop...

Current Process (ID): 2

Current Co-ordinator (ID): 0

Process 2 sent message to Co-ordinator 0

Co-ordinator 0 Didn't respond.

Election Started!

Sending message to Priorities higher than 3

No one responded.

Sending message to Priorities Lower than 3

Process 3 responded waiting for another reply

new Co-ordinator is (ID): 3

Restarting Loop...

Current Process (ID): 2

Current Co-ordinator (ID): 3

Process 2 sent message to Co-ordinator 3

Co-ordinator 3 Didn't respond.

Election Started!

Sending message to Priorities higher than 3

No one responded.

Sending message to Priorities Lower than 3

no one Responded!

Restarting Loop...