Bully.java 1 | Ring.java 1

Bully.java > ...

```java
import java.util.*;

public class Bully {
    int coordinator;
    int max_processes;
    boolean processes[];

    public Bully(int max) {
        max_processes = max;
        processes = new boolean[max_processes];

        System.out.println(x:"Creating processes...");
        for (int i = 0; i < max; i++) {
            processes[i] = true;  // All processes are initially active
            System.out.println("P" + (i + 1) + " created");
        }
        coordinator = max;  // Highest process is initially the leader
        System.out.println("Process P" + coordinator + " is the coordinator");
    }

    void displayProcesses() {
        for (int i = 0; i < max_processes; i++) {
            if (processes[i]) {
                System.out.println("P" + (i + 1) + " is up");
            } else {
                System.out.println("P" + (i + 1) + " is down");
            }
        }
        System.out.println("Current Coordinator: P" + coordinator);
    }

    void upProcess(int process_id) {
        if (process_id < 1 || process_id > max_processes) {
            System.out.println(x:"Invalid process ID.");
            return;
        }
        if (!processes[process_id - 1]) {
            processes[process_id - 1] = true;
            System.out.println("Process " + process_id + " is now up.");
        } else {
            System.out.println("Process " + process_id + " is already up.");
        }
    }

    void downProcess(int process_id) {
        if (process_id < 1 || process_id > max_processes) {
            System.out.println(x:"Invalid process ID.");
            return;
        }
        if (!processes[process_id - 1]) {
            System.out.println("Process " + process_id + " is already down.");
        } else {
            processes[process_id - 1] = false;
            System.out.println("Process " + process_id + " is down.");
        }
    }

    void runElection(int process_id) {
        if (process_id < 1 || process_id > max_processes || !processes[process_id - 1]) {
            System.out.println(x:"Invalid process ID or process is down.");
            return;
        }

        System.out.println("Process " + process_id + " started election.");
        int newCoordinator = process_id;

        // Send election messages to higher processes
        for (int i = process_id; i < max_processes; i++) {
            if (processes[i]) {
                System.out.println("Election message sent from P" + process_id + " to P" + (i + 1));
                newCoordinator = i + 1;  // Update the highest active process
            }
        }

        // The highest active process becomes the new coordinator
        coordinator = newCoordinator;
        System.out.println("New Coordinator elected: P" + coordinator);
    }

}
```

```
Run | Debug
80      public static void main(String args[]) {
81          Bully bully = null;
82          int max_processes = 0, process_id = 0;
83          int choice;
84          Scanner sc = new Scanner(System.in);
85
86          while (true) {
87              System.out.println(x:"\nBully Algorithm");
88              System.out.println(x:"1. Create processes");
89              System.out.println(x:"2. Display processes");
90              System.out.println(x:"3. Up a process");
91              System.out.println(x:"4. Down a process");
92              System.out.println(x:"5. Run election algorithm");
93              System.out.println(x:"6. Exit Program");
94              System.out.print(s:"Enter your choice: ");
95              choice = sc.nextInt();
96
97              switch (choice) {
98                  case 1:
99                      System.out.print(s:"Enter the number of processes: ");
100                     max_processes = sc.nextInt();
101                     bully = new Bully(max_processes);
102                     break;
103                 case 2:
104                     if (bully != null) bully.displayProcesses();
105                     else System.out.println(x:"Create processes first.");
106                     break;
107                 case 3:
108                     System.out.print(s:"Enter the process number to up: ");
109                     process_id = sc.nextInt();
110                     if (bully != null) bully.upProcess(process_id);
111                     else System.out.println(x:"Create processes first.");
112                     break;
113                 case 4:
114                     System.out.print(s:"Enter the process number to down: ");
115                     process_id = sc.nextInt();
116                     if (bully != null) bully.downProcess(process_id);
117                     else System.out.println(x:"Create processes first.");
118                     break;
119                 case 5:
120                     System.out.print(s:"Enter the process number to start election: ");
121                     process_id = sc.nextInt();
122                     if (bully != null) {
123                         bully.runElection(process_id);
124                         bully.displayProcesses();
125                     } else {
126                         System.out.println(x:"Create processes first.");
127                     }
128                     break;
129                 case 6:
130                     System.exit(status:0);
131                     break;
132                 default:
133                     System.out.println(x:"Invalid choice. Try again.");
134                     break;
135             }
136         }
137     }
138 }
139
```

Ring:

Bully.java 1 | Ring.java 1

Ring.java > ...

```java
import java.util.*;

public class Ring {
    int max_processes;
    int coordinator;
    boolean processes[];
    ArrayList<Integer> pid;

    public Ring(int max) {
        coordinator = max;
        max_processes = max;
        pid = new ArrayList<>();
        processes = new boolean[max];

        for (int i = 0; i < max; i++) {
            processes[i] = true;  // All processes are initially alive
            System.out.println("P" + (i + 1) + " created.");
        }
        System.out.println("P" + coordinator + " is the initial coordinator.");
    }

    void displayProcesses() {
        for (int i = 0; i < max_processes; i++) {
            if (processes[i])
                System.out.println("P" + (i + 1) + " is up.");
            else
                System.out.println("P" + (i + 1) + " is down.");
        }
        System.out.println("Current Coordinator: P" + coordinator);
    }

    void upProcess(int process_id) {
        if (process_id < 1 || process_id > max_processes) {
            System.out.println(x:"Invalid process ID.");
            return;
        }
        if (!processes[process_id - 1]) {
            processes[process_id - 1] = true;
            System.out.println("Process P" + process_id + " is now up.");
        } else {
            System.out.println("Process P" + process_id + " is already up.");
        }
    }

    void downProcess(int process_id) {
        if (process_id < 1 || process_id > max_processes) {
            System.out.println(x:"Invalid process ID.");
            return;
        }
        if (!processes[process_id - 1]) {
            System.out.println("Process P" + process_id + " is already down.");
        } else {
            processes[process_id - 1] = false;
            System.out.println("Process P" + process_id + " is down.");
        }
    }

    void displayArrayList(ArrayList<Integer> pid) {
        System.out.print(s:"[ ");
        for (Integer x : pid) {
            System.out.print(x + " ");
        }
        System.out.print(s:"]\n");
    }

    void initElection(int process_id) {
        if (process_id < 1 || process_id > max_processes || !processes[process_id - 1]) {
            System.out.println(x:"Invalid process ID or process is down.");
            return;
        }

        System.out.println("Process P" + process_id + " started election.");
        pid.clear(); // Clear previous elections
        pid.add(process_id);

        int temp = (process_id) % max_processes; // Start election in the next process

        while (temp != process_id - 1) {
            if (processes[temp]) {
                pid.add(temp + 1);
                System.out.print("Process P" + (temp + 1) + " sending the following list: ");
                displayArrayList(pid);
            }
            temp = (temp + 1) % max_processes;
        }
```

```
        // The highest process ID becomes the new coordinator
        coordinator = Collections.max(pid);
        System.out.println("Process P" + process_id + " has declared P" + coordinator + " as the new coordinator.");
        pid.clear();  // Clear for next election
    }

    Run | Debug
    public static void main(String args[]) {
        Ring ring = null;
        int max_processes = 0, process_id = 0;
        int choice;
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println(x:"\nRing Algorithm");
            System.out.println(x:"1. Create processes");
            System.out.println(x:"2. Display processes");
            System.out.println(x:"3. Up a process");
            System.out.println(x:"4. Down a process");
            System.out.println(x:"5. Run election algorithm");
            System.out.println(x:"6. Exit Program");
            System.out.print(s:"Enter your choice: ");
            choice = sc.nextInt();

            switch (choice) {
                case 1:
                    System.out.print(s:"Enter the total number of processes: ");
                    max_processes = sc.nextInt();
                    ring = new Ring(max_processes);
                    break;
                case 2:
                    if (ring != null) ring.displayProcesses();
                    else System.out.println(x:"Create processes first.");
                    break;
                case 3:
                    System.out.print(s:"Enter the process to up: ");
                    process_id = sc.nextInt();
                    if (ring != null) ring.upProcess(process_id);
                    else System.out.println(x:"Create processes first.");
                    break;
                case 4:
                    System.out.print(s:"Enter the process to down: ");
                    process_id = sc.nextInt();
                    if (ring != null) ring.downProcess(process_id);
                    else System.out.println(x:"Create processes first.");
                    break;
                case 5:
                    System.out.print(s:"Enter the process which will initiate election: ");
                    process_id = sc.nextInt();
                    if (ring != null) {
                        ring.initElection(process_id);
                        ring.displayProcesses();
                    } else {
                        System.out.println(x:"Create processes first.");
                    }
                    break;
                case 6:
                    System.exit(status:0);
                    break;
                default:
                    System.out.println(x:"Invalid choice. Try again.");
                    break;
            }
        }
    }
}
```