```
+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP                      |
| P4:   UP                      |
| P5:   UP      <-- COORDINATOR |
+-------------------------------+
+........MENU........+
1. Activate a process.
2. Deactivate a process.
3. Send a message.
4. Exit.
+....................+
2
Deactivate process:
4

+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP                      |
| P4:   DOWN                    |
| P5:   UP      <-- COORDINATOR |
+-------------------------------+
+........MENU........+
1. Activate a process.
2. Deactivate a process.
3. Send a message.
4. Exit.
+....................+
2
Deactivate process:
5
```

```
+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP      <-- COORDINATOR |
| P4:   DOWN                    |
| P5:   DOWN                    |
+-------------------------------+
+........MENU........+
1. Activate a process.
2. Deactivate a process.
3. Send a message.
4. Exit.
+....................+
1
Activate process:
4
--------Process 4 held election-------
Election message sent from process 4 to process 5

+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP                      |
| P4:   UP      <-- COORDINATOR |
| P5:   DOWN                    |
+-------------------------------+
+........MENU........+
1. Activate a process.
2. Deactivate a process.
3. Send a message.
4. Exit.
+....................+
3
Send message from process:
5
Process 5 is down
```

```
+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP                      |
| P4:   UP      <-- COORDINATOR |
| P5:   DOWN                    |
+-------------------------------+
+........MENU........+
1. Activate a process.
2. Deactivate a process.
3. Send a message.
4. Exit.
+....................+
3
Send message from process:
3
Message Sent: Coordinator is alive

+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP                      |
| P4:   UP      <-- COORDINATOR |
| P5:   DOWN                    |
+-------------------------------+
+........MENU........+
1. Activate a process.
2. Deactivate a process.
3. Send a message.
4. Exit.
+....................+
4
```

```
+------Current System State-----+
| P1:   UP                      |
| P2:   UP                      |
| P3:   UP                      |
| P4:   UP      <-- COORDINATOR |
| P5:   DOWN                    |
+-------------------------------+
```

Bully.java > Bully > coordinator

```java
import java.util.Scanner;

public class Bully {
    static boolean[] state = new boolean[5];
    public static int coordinator = 4;

    public static void getStatus() {
        System.out.println(x:"\n+------Current System State-----+");
        for (int i = 0; i < state.length; i++) {
            System.out.println("| P" + (i + 1) + ":\t" + (state[i] ? "UP" : "DOWN") +
                    (coordinator == i ? "\t<-- COORDINATOR\t|" : "\t\t\t|"));
        }
        System.out.println(x:"+-----------------------------+");
    }

   /**
    * The function checks if a process is already up and if not, holds an election and sets the
    * coordinator.
    *
    * @param up The parameter "up" is an integer representing the process number that needs to be
    * brought up or activated.
    */
    public static void up(int up) {
        if (state[up - 1]) {
            System.out.println("Process " + up + " is already up");
        } else {
            state[up - 1] = true;
            System.out.println("--------Process " + up + " held election-------");
            for (int i = up; i < state.length; ++i) {
                System.out.println("Election message sent from process " + up + " to process " + (i + 1));
            }
            for (int i = state.length - 1; i >= 0; --i) {
                if (state[i]) {
                    coordinator = i;
                    break;
                }
            }
```

Bully.java > Bully > coordinator

```java
/**
 * The "down" function sets a process state to "down" and updates the coordinator if necessary.
 *
 * @param down The parameter "down" is an integer representing the process number that needs to be
 * brought down.
 */
    public static void down(int down) {
        if (!state[down - 1]) {
            System.out.println("Process " + down + " is already down.");
        } else {
            state[down - 1] = false;
            if (coordinator == down - 1) {
                setCoordinator();
            }
        }
    }

/**
 * The function checks if the coordinator is alive and initiates an election if it is down.
 *
 * @param mess The parameter "mess" is an integer representing the process number that is sending a
 * message.
 */
    public static void mess(int mess) {
        if (state[mess - 1]) {
            if (state[coordinator]) {
                System.out.println(x:"Message Sent: Coordinator is alive");
            } else {
                System.out.println(x:"Coordinator is down");
                System.out.println("Process " + mess + " initiated election");
                for (int i = mess; i < state.length; ++i) {
                    System.out.println("Election sent from process " + mess + " to process " + (i + 1));
                }
                setCoordinator();
            }
        } else {
            System.out.println("Process " + mess + " is down");
```

Bully.java > Bully > coordinator

```
78          }
79      }
80
81  /**
82   * This function sets the coordinator variable to the index of the last true value in the state array.
83   */
84      public static void setCoordinator() {
85          for (int i = state.length - 1; i >= 0; i--) {
86              if (state[i]) {
87                  coordinator = i;
88                  break;
89              }
90          }
91      }
92
93      /**
94       * The function presents a menu to activate, deactivate, or send messages between processes in a
95       * distributed system.
96       */
        Run | Debug
97      public static void main(String[] args) {
98          int choice;
99          Scanner sc = new Scanner(System.in);
100         for (int i = 0; i < state.length; ++i) {
101             state[i] = true;
102         }
103         getStatus();
104         do {
105             System.out.println(x:"+........MENU........+");
106             System.out.println(x:"1. Activate a process.");
107             System.out.println(x:"2. Deactivate a process.");
108             System.out.println(x:"3. Send a message.");
109             System.out.println(x:"4. Exit.");
110             System.out.println(x:"+...................+");
111             choice = sc.nextInt();
112             switch (choice) {
113                 case 1: {
```

Bully.java > Bully > coordinator

```
108             System.out.println(x:"3. Send a message.");
109             System.out.println(x:"4. Exit.");
110             System.out.println(x:"+..................+");
111             choice = sc.nextInt();
112             switch (choice) {
113                 case 1: {
114                     System.out.println(x:"Activate process:");
115                     int up = sc.nextInt();
116                     if (up == 5) {
117                         System.out.println(x:"Process 5 is the coordinator");
118                         state[4] = true;
119                         coordinator = 4;
120                         break;
121                     }
122                     up(up);
123                     break;
124                 }
125                 case 2: {
126                     System.out.println(x:"Deactivate process:");
127                     int down = sc.nextInt();
128                     down(down);
129                     break;
130                 }
131                 case 3: {
132                     System.out.println(x:"Send message from process:");
133                     int mess = sc.nextInt();
134                     mess(mess);
135                     break;
136                 }
137             }
138             getStatus();
139         } while (choice != 4);
140         sc.close();
141     }
142 }
143
144
```