```
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<Integer>();
      processes = new boolean[max];

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

```
void upProcess(int process_id) {
    if(!processes[process_id-1]) {
        processes[process_id-1] = true;
        System.out.println("Process P" + (process_id) + " is up.");
    } else {
        System.out.println("Process P" + (process_id) + " is already up.");
    }
}

void downProcess(int process_id) {
    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("[ ");
    for(Integer x : pid) {
        System.out.print(x + " ");
    }
    System.out.print(" ]\n");
}

void initElection(int process_id) {
    if(processes[process_id-1]) {
        pid.add(process_id);

        int temp = process_id;
```

```
        System.out.print("Process P" + process_id + " sending the following list:- ");
        displayArrayList(pid);

        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;
        }
        coordinator = Collections.max(pid);
        System.out.println("Process P" + process_id + " has declared P" + coordinator + " as the
coordinator");
        pid.clear();
      }
    }

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

      while(true) {
        System.out.println("Ring Algorithm");
        System.out.println("1. Create processes");
        System.out.println("2. Display processes");
        System.out.println("3. Up a process");
        System.out.println("4. Down a process");
```

```
System.out.println("5. Run election algorithm");
System.out.println("6. Exit Program");
System.out.print("Enter your choice:- ");
choice = sc.nextInt();

switch(choice) {
   case 1:
      System.out.print("Enter the total number of processes:- ");
      max_processes = sc.nextInt();
      ring = new Ring(max_processes);
      break;
   case 2:
      ring.displayProcesses();
      break;
   case 3:
      System.out.print("Enter the process to up:- ");
      process_id = sc.nextInt();
      ring.upProcess(process_id);
      break;
   case 4:
      System.out.print("Enter the process to down:- ");
      process_id = sc.nextInt();
      ring.downProcess(process_id);
      break;
   case 5:
      System.out.print("Enter the process which will initiate election:- ");
      process_id = sc.nextInt();
      ring.initElection(process_id);
      break;
   case 6:
      System.exit(0);
```

```
                break;
            default:
                System.out.println("Error in choice. Please try again.");
                break;
        }
      }
   }
}
```