Tema 4

package org.example;  
  
import java.util.Scanner;  
  
// create process class for creating a process having id and status  
class Process {  
 public int id;  
 public String status;  
  
 public Process(int id){  
 this.id = id;  
 this.status = "active";  
 }  
}  
public class BullyAlg {  
  
 Scanner sc;  
 Process[] processes;  
 int n;  
  
 public BullyAlg(){  
 sc= new Scanner(System.*in*);  
 }  
  
 // method for initializing the processes  
 public void initialize(){  
  
 // get input from the user for processes  
 System.*out*.println("Enter total number of processes of Processes");  
 n = sc.nextInt();  
  
 // initialize processes array  
 processes = new Process[n];  
 for(int i = 0; i<n; i++){  
 processes[i]= new Process(i);  
 }  
 }  
  
 // create election() method for electing process  
 public void performElection(){  
  
 // we use the sleep() method to stop the execution of the current thread  
 try {  
 Thread.*sleep*(1000);  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }  
  
 // show failed process  
 System.*out*.println("Process with id " + processes[getMaxValue()].id + " fails");  
  
 // change status to Inactive of the failed process  
 processes[getMaxValue()].status = "Inactive";  
  
 // declare and initialize variables  
 int idOfInitiator = 0;  
 boolean overStatus = true;  
  
 // use while loop to repeat steps  
 while(overStatus){  
 boolean higherProcesses = false;  
  
 // iterate all the processes  
 for(int i = idOfInitiator + 1; i< n; i++){  
 if(processes[i].status.equals("active")){  
 System.*out*.println("Process " + idOfInitiator  
 + " Passes Election("  
 + idOfInitiator+") message to process" +i);  
 higherProcesses = true;  
 }  
 }  
  
 // check for higher process  
 if(higherProcesses){  
  
 // use for loop to again iterate processes  
 for(int i = idOfInitiator + 1; i< n; i++){  
 if(processes[i].status == "active"){  
 System.*out*.println("Process " + i + " Passes Ok("+i+") message to process" + idOfInitiator);  
 }  
  
 }  
 // increment initiator id  
 idOfInitiator++;  
 }  
  
 else{  
 // get the last process from the processes that will become coordinator  
 int coord = processes[getMaxValue()].id;  
  
 // show process that becomes the coordinator  
 System.*out*.println("Finally Process " + coord + " Becomes Coordinator");  
  
 for(int i = coord - 1; i>= 0; i--){  
 if(processes[i].status == "active"){  
 System.*out*.println("Process " + coord + " send Coordinator(" + coord + ") message to process " +i);  
 }  
 }  
  
 System.*out*.println("End of Election");  
 overStatus = false;  
 break;  
 }  
 }  
  
 }  
  
 // create getMaxValue() method that returns index of max process  
 public int getMaxValue(){  
 int mxId = -99;  
 int mxIdIndex = 0;  
 for(int i = 0; i<processes.length; i++){  
 if(processes[i].status == "active" && processes[i].id >mxId){  
 mxId = processes[i].id;  
 mxIdIndex = i;  
 }  
 }  
 return mxIdIndex;  
 }  
  
 // main() method start  
 public static void main(String[] args) {  
  
 BullyAlg bully = new BullyAlg();  
  
 bully.initialize();  
 bully.performElection();  
 }  
  
}