**RingElection**

import java.util.Scanner; import java.util.ArrayList; import java.util.List;

class **RingElection** {

private int numProcesses; private int coordinator;

private boolean[] activeProcesses; public **RingElection**(int numProcesses) {

this.numProcesses = numProcesses; this.activeProcesses = new boolean[numProcesses];

*// Initially, all processes are active*

for (int i = 0; i < numProcesses; i++) { activeProcesses[i] = true;

}

*// Assume the highest-numbered process is the initial coordinator*

coordinator = numProcesses - 1; System.out.**println**("Initial Coordinator: Process " +

coordinator);

}

*// Simulate process crash*

public void **simulateCrash**(int processId) {

if (processId >= 0 && processId < numProcesses && processId != coordinator) {

activeProcesses[processId] = false; *// Process is considered crashed*

System.out.**println**("Process " + processId + " has

crashed!");

} else {

System.out.**println**("Invalid choice. This process cannot be

crashed.");

}

}

public void **startElection**(int initiator) { if (!activeProcesses[initiator]) {

System.out.**println**("Process " + initiator + " is crashed and cannot start an election.");

return;

}

System.out.**println**("\nProcess " + initiator + " is initiating an election...");

List<Integer> electionPath = new ArrayList<>(); electionPath.**add**(initiator); System.out.**println**("Election path: " + electionPath); int maxId = initiator;

int current = (initiator + 1) % numProcesses; while (current != initiator) {

if (activeProcesses[current]) { System.out.**println**("Process " + maxId + " -> Process "

+ current + " (ELECTION)");

electionPath.**add**(current); System.out.**println**("Election path: " + electionPath); if (current > maxId) {

maxId = current;

}

} else {

System.out.**println**("Process " + current + " is skipped

(CRASHED).");

}

current = (current + 1) % numProcesses;

}

*// The highest ID process becomes the coordinator*

coordinator = maxId;

System.out.**println**("\nProcess " + coordinator + " wins the election and becomes the new coordinator.");

**announceNewCoordinator**();

}

private void **announceNewCoordinator**() {

int current = (coordinator + 1) % numProcesses; while (current != coordinator) {

if (activeProcesses[current]) { System.out.**println**("Process " + coordinator + " ->

Process " + current + " (ELECTED)");

}

current = (current + 1) % numProcesses;

}

}

public static void **main**(String[] args) { Scanner scanner = new **Scanner**(System.in);

*// Read number of processes from the user* System.out.**print**("Enter the number of processes: "); int numProcesses = scanner.**nextInt**();

*// Initialize the RingElection with the number of processes*

RingElection ringElection = new **RingElection**(numProcesses);

*// Ask the user which process to crash* System.out.**print**("Enter a process to crash: "); int crashProcess = scanner.**nextInt**();

while (crashProcess < 0 || crashProcess >= numProcesses || crashProcess == (numProcesses - 1)) {

System.out.**print**("Invalid choice. Enter a valid process to

crash: ");

crashProcess = scanner.**nextInt**();

}

ringElection.**simulateCrash**(crashProcess);

*// Read the initiator process that starts the election* System.out.**print**("\nEnter the process to start the election: "); int initiator = scanner.**nextInt**();

while (initiator < 0 || initiator >= numProcesses

|| !ringElection.activeProcesses[initiator]) {

System.out.**print**("Invalid or crashed process. Enter a valid process to start the election: ");

initiator = scanner.**nextInt**();

}

ringElection.**startElection**(initiator);

}

}