BullyAlgorith

import java.util.Scanner;

class **BullyAlgorithm** { private int numProcesses; private int coordinator;

private boolean[] activeProcesses; *// Tracks active processes*

private static int crashedProcess = -1; *// Initialize with an invalid process*

*ID*

*// Constructor to initialize processes and set the coordinator*

public **BullyAlgorithm**(int numProcesses, int initialCoordinator) { this.numProcesses = numProcesses;

this.activeProcesses = new boolean[numProcesses]; for (int i = 0; i < numProcesses; i++) {

activeProcesses[i] = true;

} else {

}

if (initialCoordinator >= 0 && initialCoordinator < numProcesses) { coordinator = initialCoordinator;

System.out.**println**("Initial Coordinator: Process " + coordinator);

System.out.**println**("Invalid process ID for coordinator. Defaulting to

last process as coordinator.");

coordinator = numProcesses - 1; *// Default to last process if invalid*

*ID*

}

}

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

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

return; *// Exit if the process is crashed*

}

System.out.**println**("\nProcess " + initiator + " is starting an

election...");

*// Send ELECTION message to all higher ID processes*

for (int i = initiator + 1; i < numProcesses; i++) { if (activeProcesses[i]) {

System.out.**println**("Process " + initiator + " -> Process " + i + "

(ELECTION)");

respond.");

+ " (OK)");

if (i == crashedProcess) {

System.out.**println**("Process " + i + " has crashed and does not

} else {

System.out.**println**("Process " + i + " -> Process " + initiator

}

}

}

**simulateElectionResponses**(initiator); **determineCoordinator**(initiator);

}

private void **simulateElectionResponses**(int initiator) { System.out.**println**("\nWaiting for OK responses..."); for (int i = initiator + 1; i < numProcesses; i++) {

if (activeProcesses[i]) {

if (i == crashedProcess) {

System.out.**println**("Process " + i + " has crashed and does not

respond.");

+ " (OK)");

} else {

System.out.**println**("Process " + i + " -> Process " + initiator

}

}

}

}

private void **determineCoordinator**(int initiator) { boolean newCoordinatorFound = false;

for (int i = initiator + 1; i < numProcesses; i++) { if (activeProcesses[i] && i != crashedProcess) {

if (i > initiator) {

System.out.**println**("Process " + i + " takes over the

election.");

}

}

}

**startElection**(i); *// Higher process takes over*

newCoordinatorFound = true; return;

if (!newCoordinatorFound) { coordinator = initiator;

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

**announceNewCoordinator**();

}

}

*// Announce the new coordinator to all other active processes*

private void **announceNewCoordinator**() {

for (int i = 0; i < numProcesses; i++) {

if (i != coordinator && activeProcesses[i]) { System.out.**println**("Process " + coordinator + " -> Process " + i +

" (COORDINATOR)");

}

}

}

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

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

System.out.**print**("Enter the initial coordinator process ID: "); int initialCoordinator = scanner.**nextInt**();

BullyAlgorithm bully = new **BullyAlgorithm**(numProcesses, initialCoordinator); System.out.**print**("Enter the process ID to crash (enter -1 for no crash): "); crashedProcess = scanner.**nextInt**();

if (crashedProcess >= numProcesses || crashedProcess < -1) { System.out.**println**("Invalid process ID for crash."); crashedProcess = -1; *// Reset to no crash if invalid input*

} else if (crashedProcess != -1) { bully.activeProcesses[crashedProcess] = false;

System.out.**println**("Process " + crashedProcess + " has been crashed.");

}

System.out.**print**("Enter the process to start the election: "); int initiator = scanner.**nextInt**();

if (initiator >= 0 && initiator < numProcesses) { bully.**startElection**(initiator);

} else {

System.out.**println**("Invalid process ID.");

}

scanner.**close**();

}

}