**Bully Algorithm**

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;

        }

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

            coordinator = initialCoordinator;

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

} else {

            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)");

                if (i == c        rashedProcess) {

                    System.out.println("Process " + i + " has crashed and does not respond.");

                } else {

                    System.out.println("Process " + i + " -> Process " + initiator + " (OK)");

                }

            }

        }

        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.");

                } else {

                    System.out.println("Process " + i + " -> Process " + initiator + " (OK)");

                }

            }

        }

    }

    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 {

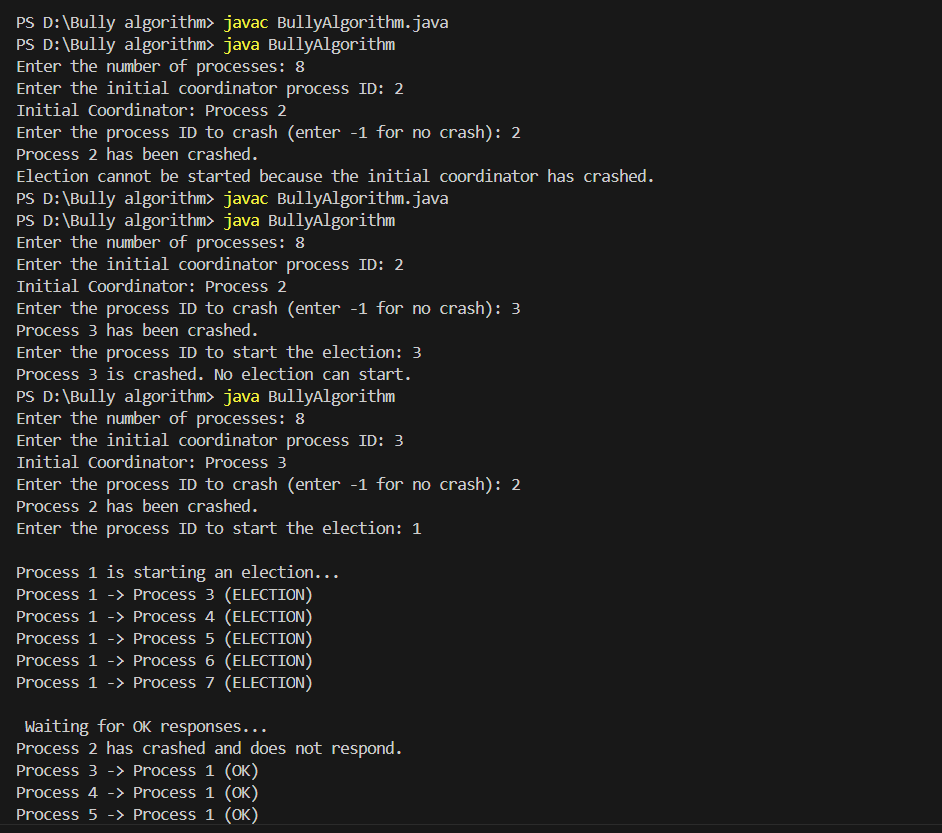
            System.out.println("Invalid process ID.");

        }

        scanner.close();

    }

}



**Ring Algorithm**

import java.util.Scanner;

import java.util.ArrayList;

import java.util.List;

class RingElection {

    private int numProcesses;          // Total number of processes

    private int coordinator;           // The current coordinator

    private boolean[] activeProcesses; // Tracks active processes

    public RingElection(int numProcesses, int initialCoordinator) {

        this.numProcesses = numProcesses;

        this.activeProcesses = new boolean[numProcesses];

        // Initially mark all processes as active

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

            activeProcesses[i] = true;

        }

        // Set the initial coordinator as provided by the user

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

            coordinator = initialCoordinator;

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

        } else {

            System.out.println("Invalid coordinator ID, defaulting to Process " + (numProcesses - 1));

            coordinator = numProcesses - 1;  // Default to the highest process if invalid input

        }

    }

    public void simulateCrash(int processId) {

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

            if (processId != coordinator) {

                activeProcesses[processId] = false;  // Mark process as crashed

                System.out.println("Process " + processId + " has crashed!");

            } else {

                System.out.println("Cannot crash the current coordinator (Process " + coordinator + ").");

            }

        } else {

            System.out.println("Invalid process ID for crash simulation.");

        }

    }

    public void startElection(int initiator) {

        // Check if the initiator is active before starting the election

        if (!activeProcesses[initiator]) {

            System.out.println("Election cannot start because Process " + initiator + " is crashed.");

            System.exit(0);  // Exit the program immediately

        }

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

        List<Integer> electionPath = new ArrayList<>();

        electionPath.add(initiator); // Add initiator to path

        System.out.println("Election path: " + electionPath);

        int maxId = initiator;  // Start with the initiator's ID as max

        int current = (initiator + 1) % numProcesses; // Move to the next process

        while (current != initiator) {

            if (activeProcesses[current]) {

                System.out.println("Process " + maxId + " -> Process " + current + " (ELECTION)");

                electionPath.add(current); // Add process to the election path

                System.out.println("Election path: " + electionPath);

                if (current > maxId) {

                    maxId = current; // Update maxId to the highest ID in the election path

                }

            } else {

                System.out.println("Process " + current + " is skipped (CRASHED).");

            }

            current = (current + 1) % numProcesses; // Move to the next process in the ring

        }

        coordinator = maxId; // The process with the highest ID becomes the coordinator

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

        announceNewCoordinator();

    }

    private void announceNewCoordinator() {

        int current = (coordinator + 1) % numProcesses; // Start from the next process

        while (current != coordinator) {

            if (activeProcesses[current]) {

                System.out.println("Process " + coordinator + " -> Process " + current + " (ELECTED)");

            }

            current = (current + 1) % numProcesses; // Move to the next process in the ring

        }

    }

    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 (0 to " + (numProcesses - 1) + "): ");

        int initialCoordinator = scanner.nextInt();

        RingElection ringElection = new RingElection(numProcesses, initialCoordinator);

        // Ask the user for the process to crash

        System.out.print("Enter the process ID to simulate crash (0 to " + (numProcesses - 1) + "): ");

        int crashProcess = scanner.nextInt();

        ringElection.simulateCrash(crashProcess); // Simulate the crash for the chosen process

        // Allow the user to start the election

        System.out.print("Enter the process to start the election: ");

        int initiator = scanner.nextInt();

        // Check if the chosen initiator is active

        if (initiator >= 0 && initiator < numProcesses) {

            if (ringElection.activeProcesses[initiator]) {

                ringElection.startElection(initiator); // Start the election if the initiator is active

            } else {

                System.out.println("Election cannot start because Process " + initiator + " is crashed.");

                System.exit(0);  // Exit the program immediately if the initiator is crashed

            }

        } else {

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

            System.exit(0);  // Exit the program if invalid process ID is entered

        }

    }

}

