**1. BullyAlgorithmServer.java**

This is the server that simulates the coordinator of the election process.

java

CopyEdit

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class BullyAlgorithmServer {

private static final int PORT = 12345;

private static List<Socket> clientSockets = new ArrayList<>();

private static List<Integer> processIDs = new ArrayList<>();

private static int currentLeader = -1;

public static void main(String[] args) {

try {

ServerSocket serverSocket = new ServerSocket(PORT);

System.out.println("Bully Algorithm Server started...");

// Accept connections from clients

while (true) {

Socket clientSocket = serverSocket.accept();

clientSockets.add(clientSocket);

DataInputStream inputStream = new DataInputStream(clientSocket.getInputStream());

int processID = inputStream.readInt();

processIDs.add(processID);

System.out.println("Process " + processID + " connected.");

}

} catch (IOException e) {

e.printStackTrace();

}

}

public static void setLeader(int leaderID) {

currentLeader = leaderID;

System.out.println("Leader elected: Process " + leaderID);

}

public static int getCurrentLeader() {

return currentLeader;

}

}

**2. BullyAlgorithmClient.java**

This is the client representing a process that can initiate or participate in the election process.

java

CopyEdit

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class BullyAlgorithmClient {

private static final String SERVER\_ADDRESS = "127.0.0.1"; // Change to the server's IP

private static final int PORT = 12345;

private static int processID;

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter process ID: ");

processID = scanner.nextInt();

try {

// Connect to the server

Socket socket = new Socket(SERVER\_ADDRESS, PORT);

DataOutputStream outputStream = new DataOutputStream(socket.getOutputStream());

outputStream.writeInt(processID);

System.out.println("Process " + processID + " connected to the server.");

// Start the election process

if (processID == 1) { // Assume process 1 initiates the election

System.out.println("Process " + processID + " is initiating the election.");

initiateElection(socket);

}

// Listen for messages from server about the elected leader

BufferedReader reader = new BufferedReader(new InputStreamReader(socket.getInputStream()));

String message;

while ((message = reader.readLine()) != null) {

System.out.println("Received: " + message);

}

socket.close();

} catch (IOException e) {

e.printStackTrace();

}

}

private static void initiateElection(Socket socket) {

try {

// Simulate sending election message to all processes with higher IDs

System.out.println("Process " + processID + " is sending election message to higher ID processes.");

// Send message to all clients (simulating election)

for (int i = processID + 1; i <= 5; i++) {

// Connect to each higher ID process and send election message

Socket targetSocket = new Socket(SERVER\_ADDRESS, PORT);

PrintWriter out = new PrintWriter(targetSocket.getOutputStream(), true);

out.println("Election message from Process " + processID);

System.out.println("Election message sent from Process " + processID + " to Process " + i);

targetSocket.close();

}

// Wait for acknowledgment and proceed with leader election

System.out.println("Election in progress...");

// Send leader election announcement to the server

PrintWriter serverOut = new PrintWriter(socket.getOutputStream(), true);

serverOut.println("Process " + processID + " is the leader!");

} catch (IOException e) {

e.printStackTrace();

}

}

}