DPA – Lab 4

import *java.io.\**;

import *java.net.\**;

*class* Client {

    Client() {

        try {

            InetAddress ia = InetAddress.getLocalHost();

            DatagramSocket ds = new DatagramSocket();

            DatagramSocket ds1 = new DatagramSocket(1300);

            System.out.println("\nRPC Client\n");

            System.out.println("Enter method and parameters (e.g., add 3 4 or gcd 12 18 24)\n");

            while (true) {

                BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

                String str = br.readLine();

                byte[] b = str.getBytes();

                DatagramPacket dp = new DatagramPacket(b, b.length, ia, 1200);

                ds.send(dp);

                byte[] buffer = new byte[4096];

                DatagramPacket dp1 = new DatagramPacket(buffer, buffer.length);

                ds1.receive(dp1);

                String s = new String(dp1.getData(), 0, dp1.getLength());

                System.out.println("\nResult = " + s + "\n");

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

*public* *static* void main(String[] args) {

        new Client();

    }

}

import *java.util.\**;

import *java.net.\**;

*class* RPCServer {

    DatagramSocket ds;

    DatagramPacket dp;

    String str, methodName, result;

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

    RPCServer() {

        try {

            ds = new DatagramSocket(1200);

            byte[] b = new byte[4096];

            while (true) {

                dp = new DatagramPacket(b, b.length);

                ds.receive(dp);

                str = new String(dp.getData(), 0, dp.getLength());

                if (str.equalsIgnoreCase("q")) {

                    System.exit(1);

                }

                StringTokenizer st = new StringTokenizer(str);

                methodName = st.nextToken();

                numbers.clear();

                while (st.hasMoreTokens()) {

                    numbers.add(Integer.parseInt(st.nextToken()));

                }

                switch (methodName.toLowerCase()) {

                    case "add":

                        result = "" + (numbers.get(0) + numbers.get(1));

                        break;

                    case "sub":

                        result = "" + (numbers.get(0) - numbers.get(1));

                        break;

                    case "mul":

                        result = "" + (numbers.get(0) \* numbers.get(1));

                        break;

                    case "div":

                        result = numbers.get(1) != 0 ? "" + (numbers.get(0) / numbers.get(1)) : "Division by zero";

                        break;

                    case "gcd":

                        result = "" + gcd(numbers);

                        break;

                    case "lcm":

                        result = "" + lcm(numbers);

                        break;

                    default:

                        result = "Unknown method";

                        break;

                }

                DatagramSocket ds1 = new DatagramSocket();

                byte[] b1 = result.getBytes();

                DatagramPacket dp1 = new DatagramPacket(b1, b1.length, InetAddress.getLocalHost(), 1300);

                ds1.send(dp1);

            }

        } catch (Exception e) {

            e.printStackTrace();

        }

    }

*private* int gcd(List<Integer> nums) {

        int result = nums.get(0);

        for (int i = 1; i < nums.size(); i++) {

            result = gcdTwo(result, nums.get(i));

        }

        return result;

    }

*private* int gcdTwo(int a, int b) {

        return b == 0 ? a : gcdTwo(b, a % b);

    }

*private* int lcm(List<Integer> nums) {

        int result = nums.get(0);

        for (int i = 1; i < nums.size(); i++) {

            result = lcmTwo(result, nums.get(i));

        }

        return result;

    }

*private* int lcmTwo(int a, int b) {

        return a \* (b / gcdTwo(a, b));

    }

*public* *static* void main(String[] args) {

        new RPCServer();

    }

}