/\*

CL-9

BE-9 R9

Roll No. 43147

**Assignment 1 B**

Power Calculation: Design a distributed application which consist of a client server communication using TCP, UDP & RMI techniques in Java. Multiple clients can simultaneously connect to the server and send messages of the format -> (a, b) where a and bare integers and server returns the value a^b (a raised to b).

\*/

**// Power Calculation: using UDP**

**//Client.java**

// Importing required classes

import java.io.IOException;

// Importing classes fromjava.nio package as

// this package is responsible for networking

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.Scanner;

public class Client {

// Main driver method

public static void main(String args[])

throws IOException

{

// Creating an object of Scanner class to read user input

Scanner sc = new Scanner(System.in);

// Step 1

// Create the socket object for carrying data

DatagramSocket ds = new DatagramSocket();

InetAddress ip = InetAddress.getLocalHost();

byte buf[] = null;

// loop while user not enters "Exit"

while (true) {

System.out.print(

"Enter the two numbers in given format: ");

System.out.println(

"'number1 , number2'");

System.out.println(

"type 'Exit' to terminate \n");

// Awaiting from entered input

String inp = sc.nextLine();

buf = new byte[65535];

// Converting the String input into the byte array

buf = inp.getBytes();

// Step 2

// Creating the datagramPacket for sending the data.

DatagramPacket DpSend = new DatagramPacket(

buf, buf.length, ip, 1234);

// Invoking the send call to actually send the data.

ds.send(DpSend);

// Break the loop if user enters "Exit" using the break keyword

if (inp.equals("Exit"))

break;

buf = new byte[65535];

// Creating an object of DatagramPacket class

DatagramPacket DpReceive

= new DatagramPacket(buf, buf.length);

ds.receive(DpReceive);

// Print and display command

System.out.println(

"Answer = "

+ new String(buf, 0, buf.length));

}

}

}

**// Server.java**

// Importing required classes

import java.io.IOException;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.util.StringTokenizer;

import java.lang.Math;

class Server {

// Main driver method

public static void main(String[] args)

throws IOException

{

// Creating a socket to listen at port 1234

DatagramSocket ds = new DatagramSocket(1234);

byte[] buf = null;

// Initializing them initially with null

DatagramPacket DpReceive = null;

DatagramPacket DpSend = null;

while (true) {

buf = new byte[65535];

// Creating a DatagramPacket to receive the data.

DpReceive = new DatagramPacket(buf, buf.length);

// Receiving the data in byte buffer.

ds.receive(DpReceive);

String inp = new String(buf, 0, buf.length);

// Using trim() method to

// remove extra spaces.

inp = inp.trim();

System.out.println("\nEquation Received:- "

+ inp);

// Exit the server if the client sends "bye"

if (inp.equals("Exit")) {

System.out.println(

"\nClient sent Exit.....Terminating");

// Exit from program here itself without checking further

break;

}

// Use StringTokenizer to break the input string

StringTokenizer st = new StringTokenizer(inp);

Double a = Double.parseDouble(st.nextToken());

String operation = st.nextToken();

Double b = Double.parseDouble(st.nextToken());

Double result = Math.pow(a, b);

System.out.println("Sending the result...");

String res = Double.toString(result);

// Clearing the buffer after every message

buf = res.getBytes();

// Getting the port of client

int port = DpReceive.getPort();

DpSend = new DatagramPacket(

buf, buf.length, InetAddress.getLocalHost(),

port);

ds.send(DpSend);

}

}

}

// OUTPUT

Graphical user interface

Description automatically generated