*Created client*

*/\*\*PingClient.java*

*\*****@author:*** *Breeda Gonzalez*

*\* group members: Dan and Ike*

*\* Add details about what this program does here*

*\*/*

import java.util.\*;

import java.net.\*;

import java.text.\*;

public class PingClient

{

private static final int *PING\_MESSAGES* = 10;

private static final int *TOKEN\_TIMESTAMP* = 2;

private static final int *MAX\_WAIT\_TIME* = 1000; //milliseconds

private static final String *CRLF* = "\r\n";

public static void main(String[] args) throws Exception

{

/\*

\* Check that the arguments are entered in the system. Order should be

\* 1. IP address

\* 2. port number

\*/

if (args.length != 2)

{

System.*out*.println("usage: java PingClient <Host> <Port>");

}

try

{

if(!args[0].isEmpty())

System.*out*.println("\nHost: "+args[1]+"\nIP address: "+args[0]+"\n");

}

catch(ArrayIndexOutOfBoundsException e)

{

System.*out*.println("Host name and ip address please");

System.*exit*(1);

}

//from argument provided in assignment

InetAddress host = InetAddress.*getByName*(args[0]);

//from second argument, provided in assignment

int portNumber = Integer.*parseInt*(args[1]);

//Create datagram socket used for sending and receiving packets

DatagramSocket socket = new DatagramSocket();

//Set up the socket's max wait time for responses

socket.setSoTimeout(*MAX\_WAIT\_TIME*);

//Construct a ping message to be sent to the server

for (int sequence\_num = 0; sequence\_num <= *PING\_MESSAGES*; sequence\_num++)

{

String message = *generatePing*(sequence\_num);

DatagramPacket ping\_request =

new DatagramPacket(message.getBytes(), message.length(), host, portNumber);

//Send a ping request

socket.send(ping\_request);

//Datagram packet to hold server response

DatagramPacket ping\_response =

new DatagramPacket(new byte[message.length()], message.length());

//Wait for ping response from server

try

{ socket.receive(ping\_response);

*printData*(ping\_response);

}

catch (SocketTimeoutException e)

{ System.*out*.println("No response was received from the server; request timed out");

}

catch (Exception e)

{

//Any other errors that may occur to be caught

e.printStackTrace();

return; } }

}

private static String generatePing(int sequence\_num)

{

// For getting current date and time

SimpleDateFormat sdfNow = new SimpleDateFormat("MM/dd/yyyy HH:mm:ss");

String strNow = sdfNow.format(new Date(System.*currentTimeMillis*()));

return "PING #" + sequence\_num + " " + System.*currentTimeMillis*() + " ("+strNow+")";

}

//Print ping page to standard output stream

private static void printData(DatagramPacket request) throws Exception

{

String response = new String(request.getData());

String[] tokens = response.split(" ");

//Create sent and received timestamps for RTT

long sent\_timestamp = new Long(tokens[*TOKEN\_TIMESTAMP*]);

long received\_timestamp = System.*currentTimeMillis*();

//RTT calculation

long rtt = received\_timestamp - sent\_timestamp;

//print results to console

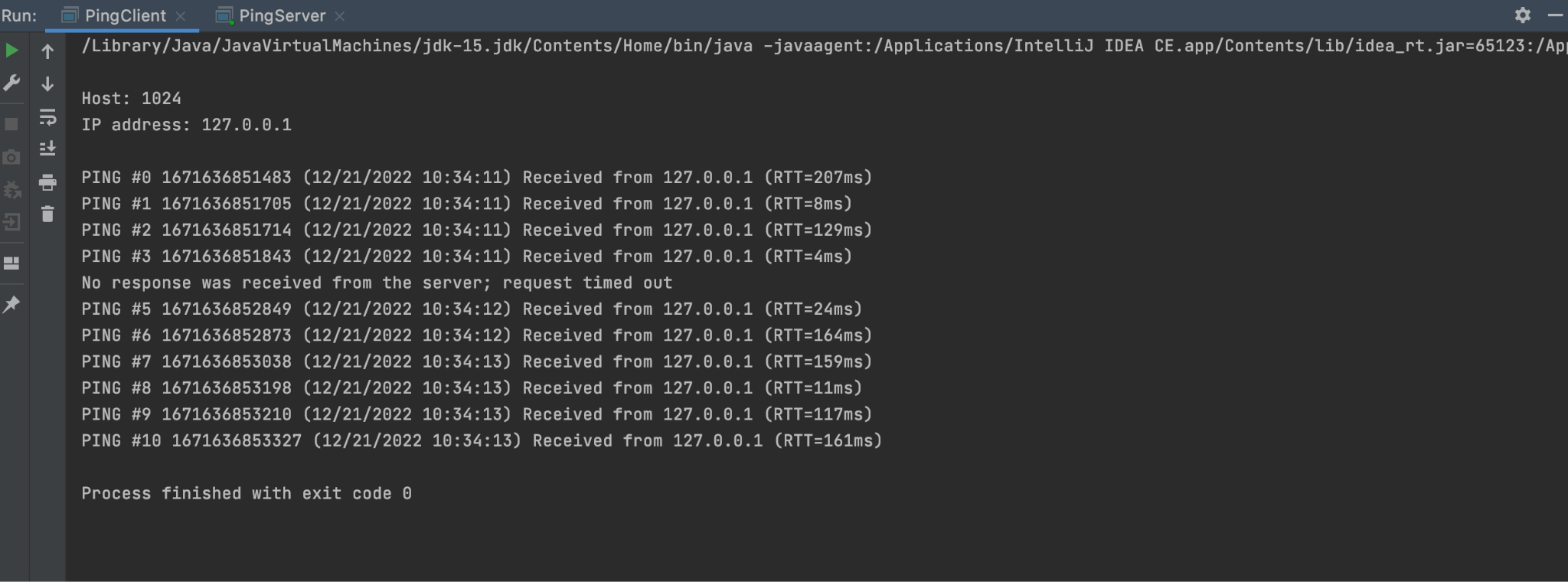
System.*out*.print(response+" Received from "+

request.getAddress().getHostAddress() + " "+"(RTT=" + rtt + "ms)"+*CRLF*);

}

}

Screenshot showing success:



Copy of Server in Java

*/\*\*UDP\_Pinger\_Server.java*

*\*Pre-supplied code of the server. Used for testing the client code that was written*

*\*/*

import java.io.\*;

import java.net.\*;

import java.util.\*;

// Server to process ping requests over UDP.

public class PingServer {

private static final double *LOSS\_RATE* = 0.3;

private static final int *AVERAGE\_DELAY* = 100; // milliseconds

public static void main(String[] args) throws Exception {

// Get command line argument.

if (args.length != 1) {

System.*out*.println("Required arguments: port");

return;

}

int port = Integer.*parseInt*(args[0]);

// Create random number generator for use in simulating

// packet loss and network delay.

Random random = new Random();

// Create a datagram socket for receiving and sending UDP packets

// through the port specified on the command line.

DatagramSocket serverSocket = new DatagramSocket(port);

// Processing loop.

while (true) {

// Create a datagram packet to hold incomming UDP packet.

DatagramPacket request = new DatagramPacket(new byte[1024], 1024);

// Block until the host receives a UDP packet.

serverSocket.receive(request);

// Print the recieved data.

*printData*(request);

// Decide whether to reply, or simulate packet loss.

if (random.nextDouble() < *LOSS\_RATE*) {

System.*out*.println(" Reply not sent.");

continue;

}

// Simulate network delay.

Thread.*sleep*((int) (random.nextDouble() \* 2 \* *AVERAGE\_DELAY*));

// Send reply.

InetAddress clientHost = request.getAddress(); // client

int clientPort = request.getPort(); // Client port

byte[] buf = request.getData(); // data being sent

DatagramPacket reply = new DatagramPacket(buf, buf.length, clientHost, clientPort);

serverSocket.send(reply);

System.*out*.println(" Reply sent.");

}

}

// Print ping data to the standard output stream.

private static void printData(DatagramPacket request) throws Exception {

// Obtain references to the packet's array of bytes.

byte[] buf = request.getData();

// Wrap the bytes in a byte array input stream,

// so that you can read the data as a stream of bytes.

ByteArrayInputStream bais = new ByteArrayInputStream(buf);

// Wrap the byte array output stream in an input stream reader,

// so you can read the data as a stream of characters.

InputStreamReader isr = new InputStreamReader(bais);

// Wrap the input stream reader in a bufferred reader,

// so you can read the character data a line at a time.

// (A line is a sequence of chars terminated by any combination of \r and \n.)

BufferedReader br = new BufferedReader(isr);

// The message data is contained in a single line, so read this line.

String line = br.readLine();

// Print host address and data received from it.

System.*out*.println("Received from " + request.getAddress().getHostAddress() + ": " + new String(line));

}

}