/\*\*

\* InetServer.java

\*/

import java.io.\*;

import java.net.\*;

/\*\*

\* Worker thread class

\*/

class Worker extends Thread

{

Socket socket;

Worker (Socket s)

{

this.socket = s;

}

/\*\*

\* Run the inet server

\*/

public void run()

{

System.out.println("Client Connected");

PrintStream out = null;

BufferedReader in = null;

try

{

// create an input stream on the specified socket

in = new BufferedReader( new InputStreamReader(this.socket.getInputStream()));

// create an output stream on the specified socket

out = new PrintStream(this.socket.getOutputStream());

try

{

String name;

// get the name from the input pipe

name = in.readLine();

// reply to the client by writing to the sockets output stream

out.println("Looking up " + name);

printRemoteAddress(name, out);

}

catch(IOException e)

{

out.println("Server read error");

e.printStackTrace();

}

finally

{

this.socket.close();

}

}

catch( IOException e)

{

System.out.println("Error opening i/o pipe on the specified socket: " + e);

}

}

/\*\*

\* Print the host name and ip address

\* @param name

\* @param out

\*/

private void printRemoteAddress(String name, PrintStream out)

{

try

{

out.println("Looking up " + name + "...");

InetAddress machine = InetAddress.getByName(name);

out.println("Host name: " + machine.getHostName());

out.println("Host IP: " + this.toText(machine.getAddress()));

}

catch(UnknownHostException e)

{

out.println("Failed to lookup name: " + name);

}

}

/\*\*

\* Take the given byte array and turn it into a string

\* @param ip a byte array representation of an ip address

\* @return a string representation of the ip address

\*/

public String toText(byte ip[])

{

StringBuffer result = new StringBuffer();

for( int i = 0; i <ip.length; ++i)

{

if(i > 0)

{

result.append(".");

}

result.append(0xff & ip[i]);

}

return result.toString();

}

}

/\*\*

\* Inet Server class

\*/

public class InetServer

{

public static void main(String args[])

{

int q\_len = 6;

int port = 1565;

Socket socket;

if( args.length == 0)

{

System.out.println("No port specified so using default: 1565");

System.out.println("Usage: java InetServer -p [port to open]");

}

for(int i = 0; i < args.length; ++i)

{

if( args[i].contains("-p"))

{

port = Integer.parseInt(args[i + 1]);

}

}

try

{

// create a socket on the given port

ServerSocket serverSocket = new ServerSocket(port, q\_len);

System.out.println(String.format("Tommy Leedberg's Inet server 1.8 starting up, listening at port %s.\n", port));

while(true)

{

socket = serverSocket.accept(); // wait for the next client connection

new Worker(socket).start(); // Spawn worker to handle it

}

}

catch(IOException e)

{

System.out.println( "Failed to start server with exception: " + e);

}

}

}

/\*\*

\* InetClient.java

\*/

import java.io.\*;

import java.net.\*;

/\*\*

\* Inet Client class to connect to an inet server

\*/

public class InetClient

{

public static void main(String args[])

{

// get the server name

String serverName = "localhost";

int port = 1565;

if( args.length == 0)

{

System.out.println("No port specified so using default port: 1565");

System.out.println("No hostname specified so using default: localhost");

System.out.println("Usage: java InetServer.java -p [port to open] -h [host name]");

}

// look for the CL params for the port or hostname

for(int i = 0; i < args.length; ++i)

{

if( args[i].contains("-p"))

{

port = Integer.parseInt(args[i + 1]);

}

if( args[i].contains("-h"))

{

serverName = args[i + 1];

}

}

System.out.println("Tommy Leedberg's Inet Client, 1.8.\n");

System.out.println(String.format("Using server: " + serverName + ", Port: %s\n", port));

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

try

{

String name = "";

while( !name.contains("quit"))

{

System.out.print("Enter a hostname or an IP address to get from the server, (quit) to end: ");

System.out.flush();

name = in.readLine();

if (name.indexOf("quit") < 0)

{

getRemoteAddress(name, serverName, port);

}

}

System.out.println("Cancelled by user request.");

}

catch (IOException x)

{

x.printStackTrace();

}

}

/\*\*

\* Get the remote address

\* @param name

\* @param serverName

\*/

private static void getRemoteAddress(String name, String serverName, int port)

{

Socket socket;

BufferedReader fromServer;

PrintStream toServer;

String textFromServer;

try

{

// Open a connection to server

socket = new Socket(serverName, port);

// Open an I/O pipe with the socket

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

toServer = new PrintStream(socket.getOutputStream());

// Send machine name or IP address to server

toServer.println(name);

toServer.flush();

// read in and then print out the response from the server

while((textFromServer = fromServer.readLine()) != null && textFromServer.length() != 0)

{

System.out.println(textFromServer);

}

socket.close();

}

catch (IOException x)

{

System.out.println("Socket error.");

x.printStackTrace();

}

}

}