ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG VIỆT - HÀN
VIETNAM - KOREA UNIVERSITY OF INFORMATION AND COMMUNICATION TECHNOLOGY
한-베정보통신기술대학교

Editor: Huynh Cong Phap

Lab 1: TCP & UDP Socket

1.

Write a program that accepts the input string, uses it to construct a URL object and returns the properties.

Ans:

```
// program that accepts input string and uses it to construct a URL object

import java.net.*;
public class net1
{
    public static void main(String arg[ ])
    {
        try
        {
            URL u = new URL(arg[0]);
            System.out.println("Name of the file is: " + u.getFile( ));
            System.out.println("Host Name is: " + u.getHost( ));
            System.out.println("Port number is: " + u.getPort( ));
            System.out.println("Protocol type is: " + u.getProtocol( ));
        }
        catch(MalformedURLException e)
        {
            System.out.println(e);
        }
    }
}
```

2. Create two standalone applications. The first application will receive through the command line the following data: name, id and age. Then it will add the appropriate line to the 'students' table (which you should create in advance). The second application will print out to the screen all the information that the 'students' table has. The table 'students' should have the following columns: name, id and age.

The first application that enables adding a student:

```
import java.sql.*;


public class AddStudent
{
	public static void main(String args[])
	{
		Connection connection = null;
		try
		{
			Class.forName("org.gjt.mm.mysql.Driver").newInstance();
		}
		catch(Exception e)
		{
			System.out.println("Registering the Driver was failed");
		}

		try
		{
			connection =
```

```
DriverManager.getConnection("jdbc:mysql://localhost:3306/universityDB");
                }
                catch(Exception exception)
                {
                                        System.out.println("Connection with the
DataBase failed");
                                        exception.printStackTrace();
                                        System.exit(1);
                }

                Statement statement = null;

                try
                {
                        statement = connection.createStatement();

                        int temp = statement.execute("INSERT INTO students
VALUES('"+args[0]+"','"+args[1]+"','"+args[2]+"')");

                        if(temp==1)
                        {
                                System.out.println("a line was added to the table");
                        }
                }
                catch(SQLException e)
                {
                        e.printStackTrace();
                }
                finally
                {
```

```
                try
                {
                        statement.close();
                        connection.close();
                }
                catch(SQLException e)
                {
                        e.printStackTrace();
                }
        }
    }
}
```

The second application enables getting all the students' details:

```
import java.sql.*;

public class GetStudents
{
    public static void main(String args[])
    {
        Connection connection = null;
        try
        {
            Class.forName("org.gjt.mm.mysql.Driver").newInstance();
```

```
                }
                catch(Exception e)
                {
                        System.out.println("Registering the Driver was failed");
                }

                try
                {
                        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/universityDB");
                }
                catch(Exception exception)
                {
                                        System.out.println("Connection with the
DataBase failed");
                                        exception.printStackTrace();
                                        System.exit(1);
                }

                Statement statement = null;

                try
                {
                        statement = connection.createStatement();
                        ResultSet result = statement.executeQuery("SELECT * FROM
students");
                        while(result.next())
                        {
                                System.out.println(result.getString(1)+","+result.getString(2)+
","+result.getString(3));
                        }
```

```
                }
                catch(SQLException e)
                {
                        e.printStackTrace();
                }
                finally
                {
                        try
                        {
                                statement.close();
                                connection.close();
                        }
                        catch(SQLException e)
                        {
                                e.printStackTrace();
                        }
                }
        }
}
```

3. Create two stand-alone applications (a client and a server). The client should allow the user supply (in the command line) one of the following numbers: 1,2 or 3. The given number will be sent to the server. The server will respond in sending back one of the followings: “ONE”, “TWO” or “THREE” (according to the number it received). You should use TCP\IP protocol.

```
import java.io.*;

import java.net.*;

import java.util.*;
```

```
class NumbersTCPSimpleServer
{

  private ServerSocket sSoc;
  public static final int PORT = 1300;

  public static void main(String args[])
  {
        NumbersTCPSimpleServer server = new NumbersTCPSimpleServer();
              try
              {
                    server.go();
              }
              catch(IOException e)
              {
                    e.printStackTrace();
              }
              catch(ClassNotFoundException e)
              {
                    e.printStackTrace();
              }
  }


  public void go() throws IOException, ClassNotFoundException
  {
```

```
Socket soc = null;

sSoc = new ServerSocket(PORT);

while(true)

{

        //creating a Socket
                System.out.println("I am the great server. Now I am waiting ...");
                soc = sSoc.accept();
                System.out.println("I stopped waiting. A Socket was instantiated");

                //creating an OutputStream object
                OutputStream os = soc.getOutputStream();

        //creating an ObjectOutputStream object
        ObjectOutputStream oos = new ObjectOutputStream(os);

                //creating an InputStream object
                InputStream is = soc.getInputStream();

        //creating an ObjectOutputStream object
        ObjectInputStream ois = new ObjectInputStream(is);

                String received = (String)ois.readObject();
                String returned = null;
                if(received.equals("1"))
                {
                        returned = "ONE";
```

```
                }
                else
                {
                    if(received.equals("2"))
                    {
                        returned = "TWO";
                    }
                    else
                    {
                        if(received.equals("3"))
                        {
                            returned = "THREE";
                        }
                        else
                        {
                            returned = "UNKNOWN";
                        }
                    }
                }
                oos.writeObject(returned);

                ois.close();
                oos.close();
            soc.close();
        }
      }
    }
```

```
class NumbersTCPSimpleClient
{


    public static final int DT_PORT = 1300;

    String hostName;

    Socket soc;



    public static void main(String args[])
    {
        NumbersTCPSimpleClient client = new NumbersTCPSimpleClient(args[0]);
                        try
                        {
                client.go(args[1]);
                        }
                        catch(IOException e)
                        {
                                e.printStackTrace();
                        }
                        catch(ClassNotFoundException e)
                        {
                                e.printStackTrace();
                        }
          }
```

```
public NumbersTCPSimpleClient(String hostString)
{
    this.hostName = hostString;
}



public void go(String str) throws IOException, ClassNotFoundException
{

    soc = new Socket(hostName, DT_PORT);

    //creating an InputStream that will be connected to soc
    InputStream is = soc.getInputStream();

    //creating ObjectInputStream
    ObjectInputStream ois = new ObjectInputStream(is);

    //creating an OutputStream that will be connected to soc
    OutputStream os = soc.getOutputStream();

    //creating ObjectOutputStream
    ObjectOutputStream oos = new ObjectOutputStream(os);

                    //sending data to server
                    oos.writeObject(str);
                    System.out.println(str+" was sent");

                    //receiving data from server
```

```
                String temp = (String)ois.readObject();
                System.out.println(temp + " was received");

            oos.close();
            ois.close();
            soc.close();
        }

    }
```

Lab 2: URL

1. Write a program that lists the contents i.e., entire text of any web page on the net.

Ans:

```
import java.net.*;

public class net2
{
    public static void main(String arg[ ])
    {
        int i;
        BufferedInputStream bis;
        try
        {
            URL u = new URL(arg[0]);
            bis = (BufferedInputStream)u.getContent( );
            while((i = bis.read( )) > 0)
                System.out.print((char)i);
                System.out.println( );
        }
        catch(MalformedURLException e)
        {
            System.out.println(e);
        }
        catch (IOException e)
        {
            System.out.println(e);
        }
    }
}
```

2. Write a program that when given a URL string as input, opens a connection to the URL and displays the header information.

Ans:

```
public class net3
{
    public static void main(String arg[ ])
    {
        int i = 1;
        BufferedInputStream bis;
        try
        {
            URL ul = new URL(arg[0]);
            URLConnection u = ul.openConnection( );
            String s = u.getHeaderField(i);
            String sg = u.getHeaderFieldKey(i);
            while(s != null)
            {
                System.out.println("Header" + i + ":" + sg + "=" + s);
                i++;
                s = u.getHeaderField(i);
                sg = u.getHeaderFieldKey(i);
            }
        }
        catch(MalformedURLException e)
        {
            System.out.println(e);
        }
        catch(IOException e)
```

```
                {
                        System.out.println(e);
                }
        }
}
```

3. Write a program that retrieves the address of the local host, Null host and host of the site.

Ans:

```
public class net4
{
    public static void main(String arg[ ])
    {
            InetAddress i;
            BufferedInputStream bis;
            try
            {
                    i = InetAddress.getLocalHost( );
                    System.out.println("The local host is: " + i);
                    i = InetAddress.getByName(null);
                    System.out.println("The Null host is: " + i);
                    i = InetAddress.getByName("www.yahoo.com");
                    System.out.println("Yahoo Host address is: " + i);
            }
            catch(IOException e)
            {
                    System.out.println(e);
            }
```

```
    }
}
```

4. Write a program that opens a connection using the URL object and then examines the document's properties and content.

Ans:

```
public class net5
{
    public static void main(String arg[ ]) throws IOException
    {
            int i;
            URL ul = new URL ("http://www.yahoo.com");
            URLConnection url = ul.openConnection( );

            System.out.println("Date: " + new Date(url.getDate( )));
            System.out.println("Content-type:" + url.getContentType( ));
            System.out.println("Expires: " + url.getExpiration( ));
            System.out.println("Last Modified: " + url.getLastModified( ));
            int l = url.getContentLength( );
            System.out.println("Content-Length:" + l);
            if(l>0)
            {
                    System.out.println("Content");
                    InputStream is = url.getInputStream( );
                    int a = l;
                    while(((i = is.read( )) != -1) && (--a > 0))
                    {
                            System.out.print((char)i);
                    }
```

```
                    is.close();
                } else
                {
                    System.out.println("Content is not available");
                }
            }
        }
```

5. Write an application to show how to write a server and the client that goes with it.

Ans:

The example consists of two independently running Java programs: the client program and the server program. The client program is implemented by a single class, `KnockKnockClient`. The server program is implemented by two classes: `KnockKnockServer` and `KnockKnockProtocol`, `KnockKnockServer` contains the `main` method for the server program and performs the work of listening to the port, establishing connections, and reading from and writing to the socket. `KnockKnockProtocol` serves up the jokes. It keeps track of the current joke, the current state (sent knock knock, sent clue, and so on), and returns the various text pieces of the joke depending on the current state. This object implements the protocol-the language that the client and server have agreed to use to communicate.

Below are the codes for three programs.

```
//code for KnockKnock-server

import java.net.*;
import java.io.*;

public class KnockKnockServer {
   public static void main(String[] args) throws IOException {

      ServerSocket serverSocket = null;
      try {
         serverSocket = new ServerSocket(4444);
```

```
} catch (IOException e) {
    System.err.println("Could not listen on port: 4444.");
    System.exit(1);
}

Socket clientSocket = null;
try {
    clientSocket = serverSocket.accept();
} catch (IOException e) {
    System.err.println("Accept failed.");
    System.exit(1);
}

PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true);
BufferedReader in = new BufferedReader(
                                new InputStreamReader(
                                clientSocket.getInputStream()));
String inputLine, outputLine;
KnockKnockProtocol kkp = new KnockKnockProtocol();

outputLine = kkp.processInput(null);
out.println(outputLine);

while ((inputLine = in.readLine()) != null) {
    outputLine = kkp.processInput(inputLine);
    out.println(outputLine);
    if (outputLine.equals("Bye."))
      break;
}
out.close();
in.close();
```

```
        clientSocket.close();
        serverSocket.close();
    }
}
```

****

```
//code for KnockKnock-client

import java.io.*;
import java.net.*;

public class KnockKnockClient {
    public static void main(String[] args) throws IOException {

        Socket kkSocket = null;
        PrintWriter out = null;
        BufferedReader in = null;

        try {
            kkSocket = new Socket("taranis", 4444);
            out = new PrintWriter(kkSocket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(kkSocket.getInputStream()));
        } catch (UnknownHostException e) {
            System.err.println("Don't know about host: taranis.");
            System.exit(1);
        } catch (IOException e) {
            System.err.println("Couldn't get I/O for the connection to: taranis.");
            System.exit(1);
        }
```

```
        BufferedReader stdIn = new BufferedReader(new InputStreamReader(System.in));
        String fromServer;
        String fromUser;

        while ((fromServer = in.readLine()) != null) {
            System.out.println("Server: " + fromServer);
            if (fromServer.equals("Bye."))
                break;

            fromUser = stdIn.readLine();
                if (fromUser != null) {
                System.out.println("Client: " + fromUser);
                out.println(fromUser);
                }
        }

        out.close();
        in.close();
        stdIn.close();
        kkSocket.close();
    }
}
```

****

```
// code for KnockKnockProtocol

import java.net.*;
import java.io.*;

public class KnockKnockProtocol {
```

```
private static final int WAITING = 0;
private static final int SENTKNOCKKNOCK = 1;
private static final int SENTCLUE = 2;
private static final int ANOTHER = 3;

private static final int NUMJOKES = 5;

private int state = WAITING;
private int currentJoke = 0;

private String[] clues = { "Turnip", "Little Old Lady", "Atch", "Who", "Who" };
private String[] answers = { "Turnip the heat, it's cold in here!",
                             "I didn't know you could yodel!",
                             "Bless you!",
                             "Is there an owl in here?",
                             "Is there an echo in here?" };

public String processInput(String theInput) {
   String theOutput = null;

   if (state == WAITING) {
      theOutput = "Knock! Knock!";
      state = SENTKNOCKKNOCK;
   } else if (state == SENTKNOCKKNOCK) {
      if (theInput.equalsIgnoreCase("Who's there?")) {
         theOutput = clues[currentJoke];
         state = SENTCLUE;
      } else {
         theOutput = "You're supposed to say \"Who's there?\"! " +
                              "Try again. Knock! Knock!";
      }
```

```
        } else if (state == SENTCLUE) {
          if (theInput.equalsIgnoreCase(clues[currentJoke] + " who?")) {
            theOutput = answers[currentJoke] + " Want another? (y/n)";
            state = ANOTHER;
          } else {
            theOutput = "You're supposed to say \"" +
                        clues[currentJoke] +
                        " who?\"" +
                        "! Try again. Knock! Knock!";
            state = SENTKNOCKKNOCK;
          }
        } else if (state == ANOTHER) {
          if (theInput.equalsIgnoreCase("y")) {
            theOutput = "Knock! Knock!";
            if (currentJoke == (NUMJOKES - 1))
              currentJoke = 0;
            else
              currentJoke++;
            state = SENTKNOCKKNOCK;
          } else {
            theOutput = "Bye.";
            state = WAITING;
          }
        }
        return theOutput;
      }
    }
```

Running the Programs

You must start the server program first. To do this, run the server program using the Java interpreter, just as you would any other Java application. Remember to run the

server on the machine that the client program specifies when it creates the socket.

Next, run the client program. Note that you can run the client on any machine on your network; it does not have to run on the same machine as the server.

If you are too quick, you might start the client before the server has a chance to initialize itself and begin listening on the port. If this happens, you will see a stack trace from the client. If this happens, just restart the client.

If you forget to change the host name in the source code for the `KnockKnockClient` program, you will see the following error message:

```
Don't know about host: taranis
```

To fix this, modify the `KnockKnockClient` program and provide a valid host name(If you are running the application on the same machine provide the host name of your machine) for your network. Recompile the client program and try again.

If you try to start a second client while the first client is connected to the server, the second client just hangs.

When you successfully get a connection between the client and server, you will see the following text displayed on your screen:

Server: Knock! Knock!

Now, you must respond with:

Who's there?

The client echoes what you type and sends the text to the server. The server responds with the first line of one of the many Knock Knock jokes in its repertoire. Now your screen should contain this (the text you typed is in bold):

Server: Knock! Knock!
Who's there?
Client: Who's there?
Server: Turnip

Now, you respond with:

Turnip who?"

Again, the client echoes what you type and sends the text to the server. The server responds with the punch line. Now your screen should contain this:

Server: Knock! Knock!
Who's there?
Client: Who's there?

Server: Turnip
Turnip who?
Client: Turnip who?
Server: Turnip the heat, it's cold in here! Want another? (y/n)

If you want to hear another joke, type **y**; if not, type **n**. If you type **y**, the server begins again with "Knock! Knock!" If you type **n**, the server says "Bye." thus causing both the client and the server to exit.

If at any point you make a typing mistake, the `KnockKnockServer` object catches it and the server responds with a message similar to this:

Server: You're supposed to say "Who's there?"!

The server then starts the joke over again:

Server: Try again. Knock! Knock!

Note that the `KnockKnockProtocol` object is particular about spelling and punctuation but not about capitalization.

Lab 3: Multicast & Proadcast Programming

1. Build a multicast program as follows:

```
import java.net.*;
import java.io.*;

public class MulticastPeer {
  public static void main(String[] args) {

    MulticastSocket s = null;
    try {
      InetAddress group = InetAddress.getByName(args[1]);

      // The hard-coded port number is 6789 (the client must have a
      // MulticastSocket bound to this port).
      s = new MulticastSocket(6789);

      // the input InetAddress is the multicast group
      s.joinGroup(group);

      byte[] m = args[0].getBytes();


      DatagramPacket dp = new DatagramPacket(m, m.length, group, 6789);
      s.send(dp);

      // buffer used to receive
      byte[] buffer = new byte[1000];

      for(int i=0; i<3; i++) {
        DatagramPacket indp = new DatagramPacket(buffer, buffer.length);
        s.receive(indp);
        System.out.println("Received: " + new String(indp.getData(), 0,
```

```
                                        indp.getLength()));
      }
      s.leaveGroup(group);
    } catch (SocketException e) {
      System.out.println("Socket: " + e.getMessage());
    } catch (IOException e) {
      System.out.println("IO: " + e.getMessage());
    } catch (Exception e) {
      System.out.println("Misc: " + e.getMessage());
    } finally {
      if(s != null)
        s.close();
    }
  }
}
```

2. Build a broadcast program

MulticastServer.java

```
public class MulticastServer {
  public static void main(String[] args) throws java.io.IOException {
    new MulticastServerThread().start();
  }
}
```

MulticastServerThread.java

```
import java.io.*;
import java.net.*;
import java.util.*;
```

```
public class MulticastServerThread extends QuoteServerThread {

 private long FIVE_SECONDS = 5000;

 public MulticastServerThread() throws IOException {
  super("MulticastServerThread");
 }

 public void run() {
  while (moreQuotes) {
   try {
    byte[] buf = new byte[256];

    // construct quote
    String dString = null;


    if (in == null)
     dString = new Date().toString();
    else
     dString = getNextQuote();

    buf = dString.getBytes();

    // The hard-coded InetAddress of the DatagramPacket is "230.0.0.1"
    // and is a group identifier (rather than the Internet address of
    // the machine on which a single client is running). This particular
    // address was arbitrarily chosen from the reserved for this purpose.
    //
    // Created in this way, the DatagramPacket is destined for all
    // clients listening to port number 4446 who are member of the
    // "230.0.0.1" group.
```

```
        InetAddress group = InetAddress.getByName("230.0.0.1");
        DatagramPacket packet = new DatagramPacket(buf, buf.length,
                                        group, 4446);

          socket.send(packet);

        // sleep for a while
        try {
          sleep((long)(Math.random() * FIVE_SECONDS));
        } catch (InterruptedException e) { }

      } catch (IOException e) {
        e.printStackTrace();
        moreQuotes = false;
      }
    }
    socket.close();
  }
}
```

QuoteServerThread.java

```
import java.io.*;
import java.net.*;
import java.util.*;


public class QuoteServerThread extends Thread {
  // Notice that the server uses a DatagramSocket to broadcast packet received
  // by the client over a MulticastSocket. Alternatively, it could have used
  // a MulticastSocket. The socket used by the server to send the
  // DatagramPacket is not important. What's important when broadcasting
  // packets is the addressing information contained in the DatagramPacket,
  // and the socket used by the client to listen for it.
```

```
protected DatagramSocket socket = null;
protected BufferedReader in = null;
protected boolean moreQuotes = true;
private static int TTL = 128;

public QuoteServerThread() throws IOException {
  this("QuoteServerThread");
}

public QuoteServerThread(String name) throws IOException {
  super(name);

  // The port number doesn't actually matter in this example because
  // the client never send anything to the server.

  socket = new DatagramSocket(4445);

  try {
    in = new BufferedReader(new FileReader("one-liners.txt"));
  } catch (FileNotFoundException e) {
    System.err.println("Could not open quote file. Serving time instead.");
  }
}

protected String getNextQuote() {
  String returnValue = null;
  try {
    if ((returnValue = in.readLine()) == null) {
      in.close();
      moreQuotes = false;
      returnValue = "No more quotes. Goodbye.";
    }
```

```
    } catch (IOException e) {
      returnValue = "IOException occurred in server.";
    }
    return returnValue;
  }
}
```

MulticastClient.java

```
import java.io.*;
import java.net.*;
import java.util.*;

public class MulticastClient {
  public static void main(String[] args) throws IOException {
    MulticastSocket socket = new MulticastSocket(4446);
    InetAddress address = InetAddress.getByName("230.0.0.1");
    socket.joinGroup(address);

    DatagramPacket packet;

    // get a few quotes
    for (int i = 0; i < 5; i++) {
      byte[] buf = new byte[256];
      packet = new DatagramPacket(buf, buf.length);
      socket.receive(packet);

      String received = new String(packet.getData());
      System.out.println("Quote of the Moment: " + received);
    }

    socket.leaveGroup(address);
    socket.close();
```

```
    }
}
```