Overview

The first programming assignment is to implement a TCP client, called TCPAsk. TCPAsk operates in a straight-forward manner:

1. Open a TCP connection to a server at a given host address and port number.
2. Take any data that the server sends, and and print the data.
3. TCPAsk takes an optional string as parameter. This string is sent as data to the server when the TCP connection is opened, followed by a newline character (linefeed '\n').

For example, assume we want to contact the "daytime" server at "time.nist.gov". The Daytime protocol is a standardised protocol for asking a Daytime server about the current date and time. Daytime runs at TCP port 13 by default. So we could use our program in the following way:

```
$ java TCPAsk time.nist.gov 13
time.nist.gov:13 says:

58128 18-01-10 23:18:34 00 0 0  40.2 UTC(NIST) *
```

The following example uses another Internet protocol, the "whois" protocol, which is used to make queries about resources on the Internet. Here we use the third optional argument to TCPAsk to pass a string to the whois server to ask for information about the domain name "kth.se".

```
$ java TCPAsk whois.iis.se 43 kth.se
whois.iis.se:43 says:
...
```

(The output is quite lengthy, so try this for yourself and check the output!)

In the task, you will learn how to:

- create TCP sockets, and use them to send and receive data;
- design the client side of client-server communication; and
- deal with errors that can happen during the communication.

Task

You will be provided with the source code for TCPAsk. But the program does not do much. It reads and parses its arguments, calls the method TCPClient.askServer, and prints out the result. The specification of TCPClient.askServer is as follows:

```
public static String askServer(String hostname, int port, String ToServer)
throws  IOException
```

```
public static String askServer(String hostname, int port) throws  IOExcepti
on
```

Your job is to implement the TCPClient class.

Instructions and Tips

You should use the Socket class (Länkar till en externa sida.) to create the client's socket. In Java, InputStream (Länkar till en externa sida.) and OutputStream (Länkar till en externa sida.) are the basic classes for I/O, which read and write "raw" data in

the form of byte arrays. The slides from Kurose-Ross_5e ladda nersuggest to use additional "wrappers" around the socket's InputStream and OutputStream. Those wrappers add data processing and buffering to the basic InputStream and OutputStream. However, in this assigned we want to transmit data transparently without additional data processing, so there is little use of the additional wrappers. Furthermore, it can be discussed if the examples in the textbook represent the most suitable ways of doing socket I/O in Java.
We recommend that you instead of using wrappers just use InputStream and OutputStream classes for "raw" I/O with byte arrays and explicitly make the conversion between byte arrays and strings yourself. See the slides from the project

introduction about IO_Encoding.pdf ladda ner for examples of to use the Socket class (Länkar till en externa sida.) for byte I/O.

Testing

You will need servers to test against. Here are some suggestions.

Protocol	Server name	Port	Arguments	Comment
Daytime	time.nist.gov	13	–	Public server. Note that this server has limitations for how frequently you can query it (at most once every four seconds). See https://tf.nist.gov/tf-cgi/servers.cgi (Länkar till en externa sida.).
Daytime	java.lab.ssvl.kth.se	13	–	KTH server.
Whois	whois.iis.se	43	String (domain name, IP address or AS number)	Public server.
Whois	whois.internic.net	43	String (domain name, IP address or AS number)	Public server.

Echo	java.lab.ssvl.kth.se	7	String	KTH Server. Returns whatever data it receives. *Note: server does not close connection.*
Discard	java.lab.ssvl.kth.se	9	String	KTH Server. Drops whatever data it receives. *Note: server does not close connection.*
Chargen	java.lab.ssvl.kth.se	19	–	KTH Server. Generates stream of characters. *Note: server does not close connection.*

Use public servers with care. Abuse may be detected and reported.

Resources

For this task, you will be provided with the following files:

- TCPAsk.java – the TCPAsk Java program
- tcpclient/TCPClient.java – Skeleton declaration of the tcpclient.TCPClient class.

The files are available in a zip archive called "task1.zip". In this zip archive, the files are stored in a directory called "task1". So, when you unzip the archive, the "task1" directory will be created, and in this directory you will find the two Java files. *This is important, because you are expected to submit your files in a zip archive with exactly the same structure!*

You can find the zip archive here.