Overview

In this task, you will take the HTTPAsk server you did in Task 3, and turn it into a *concurrent* server. The server you did for Task 3 deals with one client at a time (most likely). A concurrent server can handle multiple clients in parallel.

What you will learn here is to:

- Use Java threading to implement a concurrent server that can handle many clients at the same time.

Task

The description here is almost the same as for Task 3: You should implement a class called ConcHTTPAsk ("Conc" because it is concurrent). It's "main" method implements the server. It takes one argument: the port number. So if you want your server to run on port 8888, you would start it in the following way:

```
$ java ConcHTTPAsk 8888
```

The difference between ConcHTTPAsk and HTTPAsk from Task 3 is that as soon as a client contacts your ConcHTTPAsk server, the client will be served immediately. It does not have to wait for ConcHTTPAsk to finish serving the current client.

Instructions and Tips

There are many different ways of implementing servers that can handle multiple clients in parallel. The approach we will use here is to use Java's support for *multithreading*. In Java, a thread represents a line of execution. A Java program can consist of many threads executing at the same time. What you should do here is to let the ConcHTTPAsk server *create one thread for each client*. In other words, when a new client contacts the server (the server returns from calling the *accept* method on its welcoming socket), the server creates a new thread, and this thread will serve the client.

So how do we create threads in Java? There are basically two ways: See Defining and Starting a Thread (Länkar till en externa sida.) in the Java Tutorials. We recommend that you use the first method (Provide a Runnable object), by creating a class that implements the Runnable interface (Länkar till en externa sida.). Creating a thread involves the following steps:

1. Define a class that implements the Runnable interface. (Let's call this class MyRunnable.)
2. Create a MyRunnable object
3. Create a Thread object, using the MyRunnable object from step 2 as parameter
4. Take the Thread object from step 3, and call its *start* method

5. This will, in turn, call the *run* method in the class that you defined in step 1.
    - The run method is where there thread will do its work.
    - This is where you should implement the code that corresponds to the HTTPAsk server from Task 3

This may seem complicated at first, but this simple example on StackOverflow is easy to follow (Länkar till en externa sida.). The example also shows how you can pass a parameter to a thread. That is something that you need to do: each thread should serve a specific client, on a separate connection socket, so the connection socket needs to be an argument to the thread.

Resources

For this task, you will not be provided with any additional resources. You should be able to take your code from Task 3, and continue working on this.