Lab 1: TCP File Exchange

Write a client-server application that allows to exchange files between two hosts.

The client, launched with a host name, a port number and a file name as arguments:

- opens a TCP connexion with the server waiting at the given host and port,
- reads binary data from the given file,
- writes this data to the TCP socket,
- closes all opened resources,
- writes to the standard output the number of bytes transfered to the server.

The server, launched with a port number and a directory name as arguments:

- waits on the given port for a client TCP connexion,
- reads binary data from the TCP socket as soon as a session is opened with a client,
- writes this data to a file named `addr-time` in the given directory, where `addr` is the socket address of the client and `time` is a long integer representing the current time,
- change the file name to `messageId` if it is a mail (using the provided `MailFile`),
- closes all opened resources related to the client and the output file,
- writes to the standard output the number of bytes received from the client,
- waits on the given port for another client TCP connexion,
- and so on…

This server should be able to manage several clients concurrently.

Technical instructions: choose between the two different versions:

- the "classical" version (if you are a beginner in Java programming), with the `java.net` and `java.io` packages, using the `ServerSocket`, `Socket`, `FileInputStream` and `FileOutputStream` classes
- the "optimized" version, using Java NIO (preferable, for efficiency reasons), with:
  - the `AsynchronousServerSocketChannel`, `AsynchronousSocketChannel` and `FileChannel` classes for the server
  - the `FileChannel#transferTo` method for the client

Coding good practice: A Java program should be composed of:

- at least one class that provides public methods implementing the program main functionalities (for example, for the client, a method to send a file to a given host and port)
- a main class that contains the `public static void main(String[] args)` method. This method should:
  - check the command line arguments and display an error/help message if they are not correct (number and type of arguments). This help message should also be displayed when the program is launched with the `-h` argument
  - create an instance of the previous class and invoke the appropriate method

Assignment: give two executable jar files with the Java sources (for the client and the server). The server jar file should not contain the provided `MailFile` class, but should run without error when the `io-utils.jar` is in the same directory as the server executable jar file.