4e année IR 2019/20

Object-oriented Programming

Socket Programming in Java

S. Yangui (INSA/LAAS)

Exercise 1:

In Java, a TCP client/server communication relies on stream sockets (`ServerSocket` and `Socket` classes). From the server side, such a communication consists of the detailed herein (below) steps:

1. Creating a `ServerSocket` instance
2. Waiting for prospective clients connections (blocking)
3. Getting the input/outputs flows
4. Communicating
5. Closing the connection

Write down the Java instruction(s) that implement(s) each of these steps.

Exercise 2:

From the client side, the communication steps are detailed below:

1. Creating a `Socket` instance
2. Getting the input/outputs flows
3. Communicating
4. Closing the connection

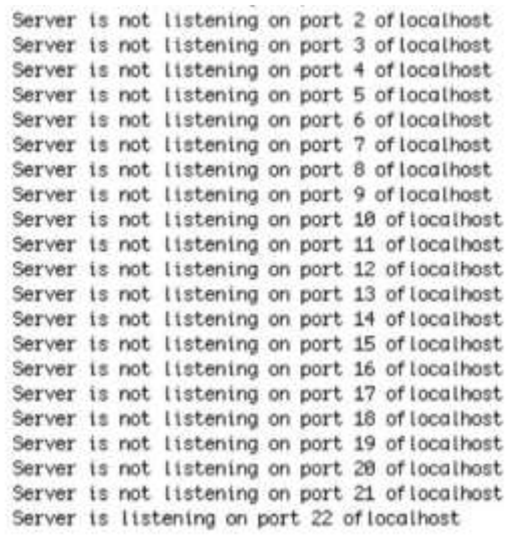
Write down the Java instruction(s) that implement(s) each of these steps.

Exercise 3:

Develop a DayTime Client/Server System where clients connect to a specific port to get the date and the time before local display.

Exercise 4:

Develop a Java program that acts as a port scanner. It checks a number of ports (for instance, from 1 to 1026) to see if they are open (a server is listening on that port number) or closed (a server is not listening on that port number).



```
Server is not listening on port 2 of localhost
Server is not listening on port 3 of localhost
Server is not listening on port 4 of localhost
Server is not listening on port 5 of localhost
Server is not listening on port 6 of localhost
Server is not listening on port 7 of localhost
Server is not listening on port 8 of localhost
Server is not listening on port 9 of localhost
Server is not listening on port 10 of localhost
Server is not listening on port 11 of localhost
Server is not listening on port 12 of localhost
Server is not listening on port 13 of localhost
Server is not listening on port 14 of localhost
Server is not listening on port 15 of localhost
Server is not listening on port 16 of localhost
Server is not listening on port 17 of localhost
Server is not listening on port 18 of localhost
Server is not listening on port 19 of localhost
Server is not listening on port 20 of localhost
Server is not listening on port 21 of localhost
Server is listening on port 22 of localhost
```

Modify your server code to make it allocating one or several ports to validate your port scanner.

Exercice 5:

Develop a concurrent TCP server. You need to code your server behavior with two different alternatives:

1. Multi-process management: The program should enable handing off processing to another process. The parent process creates the "door bell" (welcome) socket on well-known port and waits for clients to request connections. When a client does connect, the program fork off a child process to handle that connection so that parent process can return to waiting for connections as soon as possible.
2. Multithread management: same principle. You might just consider manually spawn off the threads (rather than the full processes). As a second alternative/step you may just consider the use of a thread pool where tasks are added at runtime to a pool that is managed by an orchestrator (executor service).

Exercice 6:

Develop an IP box program that opens four sockets, two TCP and two UDP.

- 2 TCP sockets
  - A receive-config socket : IP box acts as a Server (must be bound to a port you have to find, and the interface IP address)
  - A send-config socket : IP BOX acts a receiver
- 2 UDP sockets
  - App -- acts as the interface between the IP layer and the application
  - Iface – represents the network interface

Note that both UDP sockets must be bound to an used port and the interface address

The operation of IP box should be as follows:

- Send-config sockets connects to the Test Box and sends a "ready-to-test" command
- The Test box then connects to recv-config socket and send a '\n' terminated command which must be echoed
- The Test box then sends UDP packets to app and iface sockets which must be echoed (Note : If the Test box does not receive your echo, it retransmits the packet)
- On receiving both the echoes, the Test box sends a "send-stat" command to the send-config socket
- The IP box sends a "list-of-stats"
- The Test box then sends an exit message (during final test, this will have a 40 character hex string representing a hashed timestamp, which your program must RECORD!)