**COMP 3410 -**

**Operating Systems (3, 1, 0)**

**Winter 2018**

**Lab/assignment 3: Sockets**

**(1 or 2 students per group on this lab)**

**Due: Thursday February 8th**

**Introduction**

Socket and RMI are two techniques used in Java for the purpose of IPC. These are used to establish two-way communication between two running programs, which are typically a client and a server, which are running on the network. Actually, they work in quite different ways principally.

* Socket is a way to send data (only data, not methods) on a port to a different host. The protocol can be defined by the user
* RMI is a technology in which the methods of remote Java objects can be invoked from other Java virtual machines running on different hosts.

In this lab, you will learn more about client-server communication programs based on both Java Socket.

**Using Sockets**

**Learning about sockets**

Read the following resource carefully to learn about sockets:

<http://java.sun.com/docs/books/tutorial/networking/sockets/index.html>

If you never did any network programming in Java, it may be a good idea to read some more:

<http://java.sun.com/docs/books/tutorial/networking/index.html>

You may want to have a look at other information:

<http://de.wikipedia.org/wiki/Socket>  
<http://www.javaworld.com/javaworld/jw-12-1996/jw-12-sockets.html> <https://stackoverflow.com/questions/10069059/is-it-possible-to-run-a-socket-server-and-socket-client-on-the-same-machine>

**Client / Server Communication**

You need to understand the methods and packages to write client program and a server program.

Try to design and implement java classes required to do socket communication, like:

* *SocketClient.java*: This class connects to the server and communicates with it according to the ClientProtocol
* *ClientProtocol.java*: This class describes how your client reacts to your servers messages
* *SocketServer.java*: This class sets up a socket listening for the client to connect - and communicates with the client
* *ServerProtocol.java*: This class describes how the server reacts to the messages of the client
* **Have the two classes pass some sample data back and forth between them.**

**Results:**

Hand in:.

* A print out of the programs;
* A printout of the output of running programs (screen capture)