Laboratory 1: Chatterbox

(http://people.f4.htw-berlin.de/~weberwu/info2/labs/Exer1.shtml)

1. Start your chatterbox by writing a method that listens on a port. This is your chatterbox server.

First we create a new class Server. The main method gets called, creates an anonymous object from class Server.

The main method:

```
public static void main(String[] args) {
    new Server(1050);
    //new Client("127.0.0.1", 1049);
}
```

("//" = If the used computer is the Server, new Client gets commented out. If the computer is the Client, the Server gets commented out)

Server constructor:

```
public Server(int port) {
    try {
        ss = new ServerSocket(port);
    } catch (IOException io) {}
    listen();
}
```

The constructor takes a parameter port, which creates a server socket. If something goes wrong, the IOException gets caught.

At last a listen() method gets invoked.

The listen() method:

```
public void listen() {
    try {
        socket = ss.accept();
        br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

        String listen = br.readLine();
        System.out.println(listen);
        socket.close();
        ss.close();
    } catch (IOException io) {}
}
```

A socket and a buffered reader get initialized, the buffered reader reads the client's output and prints it to the console. Then the socket and server socket ss get closed. Again, if something goes wrong, the IOException gets caught.

2. Now write a client that writes to a port.

Now we create a new class Client.

Then we comment out the Server (s. picture 1) in the main method in class Server, so that we call the constructor of class Client instead of class Server.

Client constructor:

```
public Client(String host, int port) {
    try {
        socket = new Socket(host, port);
        pw = new PrintWriter(socket.getOutputStream(), true);

        write();
    }catch(IOException io) {
        System.out.println("Connection failed..");
    }
}
```

The constructor takes two parameters: String host for IP address, Int port for used port. A socket (with IP address and port) and print writer get created. The print writer writes into the output-stream of the used socket. If the connection succeeds the method write() gets called, if not IOException gets caught and "Connection failed.." gets printed to the console.

Write() method:

```
public void write() {
    sc = new Scanner(System.in);
    String text = sc.nextLine();
    pw.println(text);
}
```

A scanner sc gets initialized and reads the user input. This gets saved in a String text and then the PrintWriter prints it to the output stream.

3. Test your methods on your own machine. For now, just echo what you have read to the console to see it working. Now publish your computer name and port number on the board in the lab.

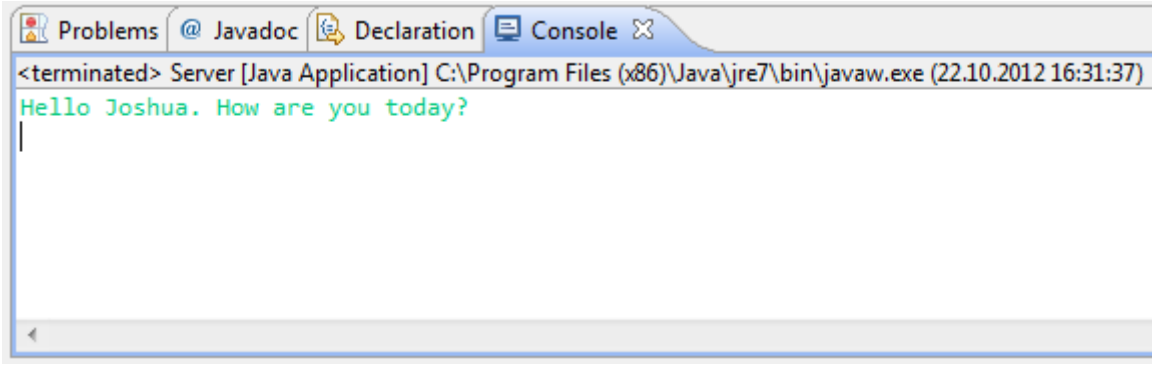
Joshua runs the program and creates a server, which listens to the port 1050:

```
public static void main(String[] args) {
    new Server(1050);
    //new Client("127.0.0.1", 1049);
}
```

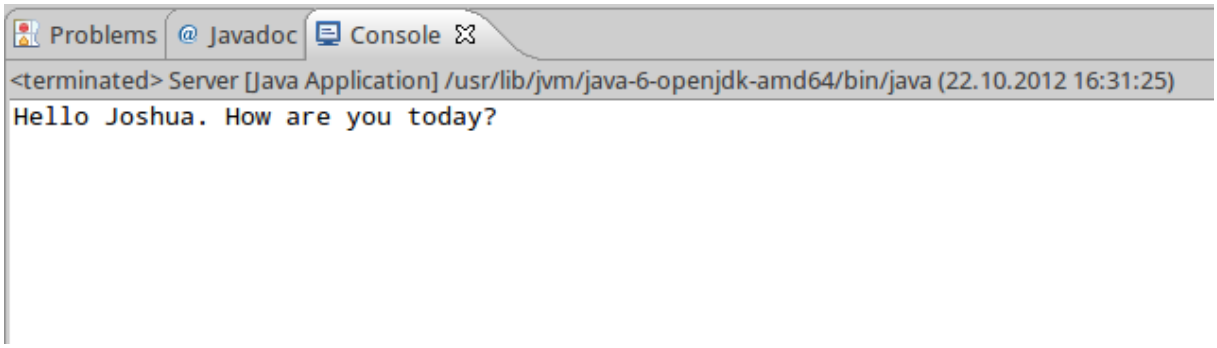
Markus also runs the program and as a client and tries to establish a connection to Joshua's IP address and port 1050.

```
public static void main(String[] args) {
    //new Server(1049);
    new Client("141.45.199.210", 1050);
}
```

Markus types in a message to Joshua:



Joshua receives this message:



It works!

4. Start chatting with a few of your neighbors! Describe what works and does not work.

At the moment it is not possible to send and receive messages simultaneously.

Also the program gets terminated after every single sent or received message.

To realize simultaneous sending and receiving we could have used threads in the code.

We worked with networking in Java for the first time, which was a complete new experience for both of us. So it took us some time to understand how the two classes communicate with each other.

All in all the lab took us round about 6 hours of work.

Source code:

Server class:

```
import java.io.*;
import java.net.*;

public class Server {
	BufferedReader br;
	ServerSocket ss;
	Socket socket;

	public Server(int port) {
		try {
			ss = new ServerSocket(port);
		} catch (IOException io) {}
		listen();
	}

	public static void main(String[] args) {
		//new Server(1049);
		new Client("141.45.199.210", 1050);
	}

	public void listen() {
		try {
			socket = ss.accept();
```

```
                        br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));


                        String listen = br.readLine();
                        System.out.println(listen);
                        socket.close();
                        ss.close();
                } catch (IOException io) {}


        }
}
```

Client class:

```
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class Client {
	Scanner sc;
	Socket socket;
	PrintWriter pw;

	public Client(String host, int port) {
		try {
			socket = new Socket(host, port);
			pw = new PrintWriter(socket.getOutputStream(), true);

			write();
		}catch(IOException io) {
			System.out.println("Connection failed..");
		}
	}

	public void write() {
		sc = new Scanner(System.in);
		String text = sc.nextLine();
		pw.println(text);
	}
}
```