Lösung 5

1.
```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLConnection;
import java.util.Date;

public class WebsideTester extends Thread {
String log="log.txt";

public void run() {
      while (!this.isInterrupted())
      {
            String path="http://tu-freiberg.de";
            URL url;
            try {  url = new URL(path);
                  URLConnection source = url.openConnection();
                  source.setUseCaches(false);
                  source.connect();
                  Object o=source.getContent();
                  int len =o.toString().length();
                  String s=new Date().toString()+" "+path+ "
"+String.valueOf(len);
                  System.out.println(s);
                  BufferedWriter bw=new BufferedWriter(new FileWriter(log,true));
                  bw.write(s);
                  bw.newLine();
                  bw.close();

                  } catch (MalformedURLException e) { // url nicht parseable
                        e.printStackTrace();
                  } catch (IOException e) {
                        String s=path+ " "+"nicht erreichbar";
//System.out.println(s);
                        BufferedWriter bw;
                        try {
                              bw = new BufferedWriter(new FileWriter(log));
                              bw.write(s); bw.newLine();
                              bw.close();
                        } catch (IOException e1) {
                              e1.printStackTrace();
                        }
                  }
                  try {
                        Thread.sleep(5000);
                  } catch (InterruptedException e) {
                        interrupt();e.printStackTrace();
                  }
            }
      }

public static void main(String[] args) {
      WebsideTester w=new WebsideTester();
      w.start();}}
```

2.
Server:
```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.net.URL;
import java.net.URLConnection;
import java.net.UnknownHostException;

public class UDPServer {

        public static void main(String[] args) {
                while (!Thread.interrupted()){
                        try {
                                byte[] inhalt_r=new byte[1024];
                                DatagramSocket socket_r;
                                socket_r = new DatagramSocket(5555);
                                DatagramPacket packet_r=new
DatagramPacket(inhalt_r,inhalt_r.length);
                                socket_r.receive(packet_r);
                                socket_r.close();

                                String path= new String(inhalt_r, 0, packet_r.getLength() );
                                byte[] inhalt_s=new byte[1];
                                try {
                                        URL url = new URL(path);
                                        URLConnection source;
                                        source = url.openConnection();
                                        source.setUseCaches(false);
                                        source.connect();
                                        inhalt_s[0]=1;

                                } catch (IOException e) {
                                        e.printStackTrace();
                                        inhalt_s[0]=0;
                                }
                                DatagramSocket socket_s=new DatagramSocket();
                                InetAddress adr=InetAddress.getByName("localhost");
                                DatagramPacket packet_s=new
DatagramPacket(inhalt_s,inhalt_s.length,adr,5556);
                                socket_s.send(packet_s);
                                socket_s.close();
                        } catch (SocketException e) {
                                        e.printStackTrace();
                        } catch (UnknownHostException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                }
            }
        }
}
```

Client:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.util.Scanner;

public class UDPClient {

        public static void main(String[] args) {
                try {
                        Scanner eingabe=new Scanner(System.in);
                        String path=eingabe.nextLine();
                        while (!path.isEmpty())
                        {
                                System.out.println("Path"+path.length());
                                DatagramSocket socket_s;
                                //String path="http://tu-freiberg.de";

                                socket_s = new DatagramSocket();
                                InetAddress adr = InetAddress.getByName("localhost");
                                byte[] inhalt_s=path.getBytes();
                                DatagramPacket packet_s = new DatagramPacket(inhalt_s,
inhalt_s.length,adr,5555);
                                socket_s.send(packet_s);
                                socket_s.close();

                                byte[] inhalt_r=new byte[1];
                                DatagramSocket socket_r;
                                socket_r = new DatagramSocket(5556);
                                DatagramPacket packet_r=new
DatagramPacket(inhalt_r,inhalt_r.length);
                                socket_r.receive(packet_r);
                                socket_r.close();
                                System.out.println(inhalt_r[0]);
                                path=eingabe.nextLine();
                        }
                        eingabe.close();

                } catch (IOException e) {
                        //inkl. SocketException,UnknownHostException
                        e.printStackTrace();
                }


        }

}
```

3.
Client:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.Scanner;

public class TCPClient {
	public static void main(String[] args) {
		try {
			Scanner stdinput = new Scanner(System.in);
			Socket socket=new Socket("localhost",12345);
			BufferedReader reader=new BufferedReader(new 
InputStreamReader(socket.getInputStream()));
			PrintWriter writer=new 
PrintWriter(socket.getOutputStream(),true);
			String anfrage;
			do{
				System.out.println("Please give one of the valid 
commands: put, get, delete, quit");
				anfrage=stdinput.nextLine();
				writer.println(anfrage);
				String antwort=reader.readLine();
				System.out.println(antwort);
			}while (!anfrage.equals("quit"));
			reader.close();
			writer.close();
			socket.close();
			stdinput.close();
		} catch (IOException e) {
			e.printStackTrace();
	}
	}
}
```

Server:

```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.concurrent.ConcurrentHashMap;

public class TCPServer {
	public static void main(String[] args) {
		ConcurrentHashMap<String,String> map=new 
ConcurrentHashMap<String,String>();
		map.put("links","poS");
		map.put("rechts","nIH");
		map.put("wo", "NuqDaq");

		ServerSocket servers = null;
		try {
			servers=new ServerSocket(12345);
```

```
                    while(true){
                    Socket s=servers.accept();

                    TCPServerThread sth=new TCPServerThread(map,s);
                    sth.start();
                    }
            } catch (IOException e) {
                    e.printStackTrace();
            }
        }
}
```

ServerThread:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.concurrent.ConcurrentHashMap;

public class TCPServerThread extends Thread {
        ConcurrentHashMap<String,String> map;
        Socket socket;

        public TCPServerThread(ConcurrentHashMap<String, String> map, Socket socket)
{
                super();
                this.map = map;
                this.socket = socket;
        }

        @Override
        public void run() {
                super.run();
                try {
                        BufferedReader reader=new BufferedReader(new
InputStreamReader(socket.getInputStream()));
                        PrintWriter writer=new
PrintWriter(socket.getOutputStream(),true);

                        String anfrage="";
                        String antwort="";

                        do{
                                anfrage=reader.readLine();
                                System.out.println(anfrage);
                                antwort=getAnswer(anfrage);
                                //antwort=anfrage;
                                writer.println(antwort);

                        }while (!anfrage.equals("quit"));

                        reader.close();
                        writer.close();
                        socket.close();
                } catch (IOException e) {
                        e.printStackTrace();
        }

    }

    private String getAnswer(String request) {
        String answer=null;
        String[] teile=request.split(" ");
        if (teile[0].equals("put")) {
            map.put(teile[1], teile[2]);
            answer=teile[2]+" saved";
        }
        else
            if (teile[0].equals("get")) {
                String vokabel=map.get(teile[1]);
                if (vokabel!=null) answer=vokabel+ " found";
                else answer=teile[1]+ "not found";
            }
            else
                if (teile[0].equals("delete")) {
                    if (map.get(teile[1])!=null)
                        {
                            map.remove(teile[1]);
                            answer=teile[1]+" removed";
                        }
                    else answer=teile[1]+" not found";
                }
                else
                    if (teile[0].equals("quit")) answer="bye";
                    else answer="what?";
        return answer;
    }

}
```