Beispiel Time Client/Server

```
/**
 *
 * Programmbeschreibung:
 * ---------------------
 * Dieses Programm ermittelt über eine TCP/IP-Verbindung die Uhrzeit eines
 * entfernten Rechners, wobei es sowohl die Rolle des Servers als auch
 * des Clients einnehmen kann.
 * Auf dem Rechner, von dem die Uhrzeit ermittelt werden soll, muß dieses
 * Programm im Modus Server (Parameter -s) gestartet werden. Dann wartet
 * es auf dem als Parameter übergebenen Port auf Anforderungen anderer
 * Rechner und sendet das aktuelle Datum und Uhrzeit in Form einer
 * Zeichenkette zurück.
 * Im Modus Client (Parameter -c) baut es eine Verbindung zum Port des
 * übergebenen Rechners auf und gibt die zurückgemeldete Uhrzeit aus.
 *
 * @author  Peter René Dietmüller
 * @version 1.0, 01/99
 *
 */

import java.io.*;
import java.net.*;
import java.text.DateFormat;
import java.util.Date;
```

```
public class Time {

  /* -- Time Server -- */
  public static void StartServer(int port) {

    Socket       client = null;  /* Socket des Clients      */
    String       cip;            /* IP-Adresse des Clients  */
    int          cport;          /* Port des Clients        */
    PrintWriter  cout;           /* Ausgabestrom zum Client */
    ServerSocket server = null;  /* Socket des Servers      */
    String       time;           /* versendete Uhrzeit      */

    /* -- Server starten -- */
    System.out.println("Starte Server und warte auf Port " + port +
                       " ... (Mit Strg-C beenden)");
    try {

      server = new ServerSocket(port);
      while(true) {

        /* -- Auf eingehende Verbindung warten -- */
        client = server.accept();
        cip   = client.getInetAddress().getHostAddress();
        cport = client.getPort();
        cout  = new PrintWriter(client.getOutputStream(), true);

        /* -- Uhrzeit ermitteln -- */
        time = DateFormat.getDateTimeInstance(DateFormat.FULL,
                                              DateFormat.FULL).format(new Date());
```

```
      /* -- Uhrzeit senden -- */
      cout.println(time);

      /* -- Trace ausgeben -- */
      System.out.println(cip + ":" + cport + " " + time);

      /* -- Verbindung schließen -- */
      client.close();

    }

  } catch(IOException e) {
    System.out.println("Server: " + e.toString());

  } finally {
    /* -- Socket schließen -- */
    try {
      if (server != null) server.close();
      System.out.println("Server gestoppt.");
    } catch (IOException e) {
      System.out.println("Server.close(): " + e.toString());
    }
  }

}
```

```
/* -- Time Client: Abfrage der Uhrzeit -- */
public static void StartClient(String server, int port) {

  String         line;
  BufferedReader in;
  Socket         s;

  System.out.println("Baue Verbindung zum Rechner " + server + ":" + port +
                     " auf.");

  try {

    /* -- Verbindungsaufbau -- */
    s = new Socket(server, port);
    s.setSoTimeout(1000);
    in = new BufferedReader(new InputStreamReader(s.getInputStream()));
    System.out.println("Verbindung zum Rechner aufgebaut.");

    /* -- Lese Uhrzeit des Servers -- */
    line = in.readLine();
    if (line != null) {
      System.out.println("Die Uhrzeit ist: " + line);
    }

    /* -- Schließe Verbindung -- */
    s.close();

  } catch(UnknownHostException e) {
    System.out.println("Der Server " + server + " ist unbekannt.");
```

```
  } catch(IOException e) {
    System.out.println("Timeout");
  }

}

/* -- Hilfetext -- */
public static void Usage() {
  System.out.println();
  System.out.println("Time ermittelt die Uhrzeit eines anderen Rechners.");
  System.out.println("(C)opryright by Peter René Dietmüller, 1998-99.");
  System.out.println();
  System.out.println("Damit die Uhrzeit eines entfernten Rechners abgefragt");
  System.out.println("werden kann, muß dieses Programm auf dem entfernten Rechner");
  System.out.println("in der Betriebsart Server laufen.");
  System.out.println();
  System.out.println("Aufruf: java Time -s <port> | -c <server> <port>");
  System.out.println();
  System.out.println("  -s ........ Programm wird in der Betriebsart Server");
  System.out.println("              gestartet. In diesem Modus wartet das");
  System.out.println("              Programm auf Anfragen und sendet die Uhrzeit");
  System.out.println("              des Rechners, auf dem es läuft.");
  System.out.println("  -c ........ Programm wird in der Betriebsart Client");
  System.out.println("              gestartet. In diesem Modus wird die Uhrzeit");
  System.out.println("              eines anderen Rechners abgefragt.");
  System.out.println("  <server> .. Name oder IP-Adresse des Servers.");
  System.out.println("  <port> .... Portnummer, auf der die Kommunikation");
  System.out.println("              stattfinden soll.");
  System.out.println();
}
```

```
/* -- Hauptprogramm -- */
public static void main(String[] args) {

  int         port;
  String      server;

  /* -- Parameter prüfen -- */
  if (args.length >= 1) {

    /* -- Server -- */
    if (args[0].equalsIgnoreCase("-s")) {
      if (args.length >= 2) {
        try {
          port = Integer.parseInt(args[1]);
          StartServer(port);
        } catch(NumberFormatException e) {
          System.out.println("Sie haben als Port keine Zahl angegeben.");
          Usage();
        }
      } else {
        System.out.println("Sie haben zu wenig Parameter angegeben.");
        Usage();
      }
```

```
      /* -- Client -- */
      } else if (args[0].equalsIgnoreCase("-c")) {
        if (args.length >= 3) {
          server = args[1];
          try {
            port = Integer.parseInt(args[2]);
            StartClient(server, port);
          } catch(NumberFormatException e) {
            System.out.println("Sie haben als Port keine Zahl angegeben.");
            Usage();
          }
        } else {
          System.out.println("Sie haben zu wenig Parameter angegeben.");
          Usage();
        }

      /* -- Falscher erster Parameter -- */
      } else {
        System.out.println("Der erste Parameter ist falsch.");
        Usage();
      }

    /* -- args.length < 1 --> Falsche Anzahl der Parameter -- */
    } else {
      System.out.println("Sie haben zu wenig Parameter angegeben.");
      Usage();
    }
  }

}
```

Test Server

```
L:\PDI\LVA\Ppk2\LEKTIO~4\Time>java Time -s 3000

Starte Server und warte auf Port 3000 ... (Mit Strg-C beenden)
127.0.0.1:1224 Montag, 11. Jänner 1999 16:52 Uhr GMT+01:00
127.0.0.1:1225 Montag, 11. Jänner 1999 16:53 Uhr GMT+01:00
127.0.0.1:1226 Montag, 11. Jänner 1999 16:53 Uhr GMT+01:00
140.78.131.99:1227 Montag, 11. Jänner 1999 16:53 Uhr GMT+01:00
127.0.0.1:1230 Montag, 11. Jänner 1999 16:54 Uhr GMT+01:00
127.0.0.1:1231 Montag, 11. Jänner 1999 16:54 Uhr GMT+01:00
140.78.131.99:1232 Montag, 11. Jänner 1999 16:54 Uhr GMT+01:00
^C
```

Test Client

```
L:\PDI\LVA\Ppk2\LEKTIO~4\Time>java Time -c localhost 3000
Baue Verbindung zum Rechner localhost:3000 auf.
Verbindung zum Rechner aufgebaut.
Die Uhrzeit ist: Montag, 11. Jänner 1999 16:54 Uhr GMT+01:00

L:\PDI\LVA\Ppk2\LEKTIO~4\Time>java Time -c 127.0.0.1 3000
Baue Verbindung zum Rechner 127.0.0.1:3000 auf.
Verbindung zum Rechner aufgebaut.
Die Uhrzeit ist: Montag, 11. Jänner 1999 16:54 Uhr GMT+01:00

L:\PDI\LVA\Ppk2\LEKTIO~4\Time>java Time -c pdi.fim.uni-linz.ac.at 3000
Baue Verbindung zum Rechner pdi.fim.uni-linz.ac.at:3000 auf.
Verbindung zum Rechner aufgebaut.
Die Uhrzeit ist: Montag, 11. Jänner 1999 16:54 Uhr GMT+01:00

L:\PDI\LVA\Ppk2\LEKTIO~4\Time>java Time -c localhost 2000
Baue Verbindung zum Rechner localhost:2000 auf.
Timeout

L:\PDI\LVA\Ppk2\LEKTIO~4\Time>java Time -c localhost 80
Baue Verbindung zum Rechner localhost:80 auf.
Verbindung zum Rechner aufgebaut.
Timeout
```

Beispiel WWW-Server

```
/**
 *
 * Programmbeschreibung:
 * ---------------------
 * Es handelt sich um einen vereinfachten WEB-Server, der nach dem dem Hypertext
 * Transfer Protocol Version 1.0 arbeitet und nur den Befehl GET versteht. Wird
 * ein anderer Befehl gesendet, so bleibt dieser unbeantwortet.
 * Dieser Server wertet nur die gesendete URL aus. Header-Zeilen, wie zum Beispiel
 * Accept, If-Modified-Since, werden nicht berücksichtigt.
 *
 * @author  Peter René Dietmüller
 * @version 1.0, 01/99
 *
 */

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

```
/**
 * Thread WWWVerbindung
 * --------------------
 * Für jede eingegangene Verbindung wird ein Exemplar der Klasse WWWVerbindung
 * angelegt, die sich um die Anfrage kümmert.
 */
class WWWVerbindung extends Thread {

  /* -- Datumsformat -- */
  static DateFormat df = DateFormat.getDateTimeInstance(DateFormat.MEDIUM,
                                                        DateFormat.MEDIUM, Locale.UK);

  Socket s;        /* Socket, auf dem die Anfrage kam   */
  String rootDir;  /* Wurzelverzeichnis des WWW-Servers */

  public WWWVerbindung(Socket s, String rootDir) {
    this.s = s;
    this.rootDir = rootDir;
  }

  private void PrintLog(String ip, int port, String line) {
    System.out.println(ip + ":" + port + ", " + line);
  }

  /* sendet den Header für ein HTML-Objekt */
  private void SendHeader(PrintWriter sout, String rsp, String ctt, long ctl, Date lm)
{
    sout.println("HTTP/1.0 " + rsp);
    sout.println("Server: PDI/0.1Beta");
```

```
  if (!ctt.equals(""))
    sout.println("Content-Type: " + ctt);
  sout.println("Content-Length: " + ctl);
  sout.println("Date: " + df.format(new Date()));
  if (lm != null)
    sout.println("Last-Modified: " + df.format(lm));
  sout.println();
  sout.flush();
}

/* sendet den Inhalt einer Datei */
private void SendFile(File f, OutputStream os) {

  int             len;
  byte[]          data;
  FileInputStream fin;

  /* -- Initialisieren -- */
  fin  = null;
  data = new byte[1024];

  try {
    /* -- InputStream anlegen -- */
    fin = new FileInputStream(f);
    /* -- Daten kopieren -- */
      while ((len = fin.read(data)) > 0) os.write(data, 0, len);
  } catch(FileNotFoundException e) {
  } catch(IOException e) {
  } finally {
```

```
    try {
      if (fin != null) fin.close();
    } catch(IOException ioe) {
    }
  }

}

/* sendet eine Fehlermeldung an den Client */
private void SendError(PrintWriter sout, int error) {

  String errMsg;

  /* -- Fehlermeldung festlegen -- */
  switch(error) {
    case 200: errMsg = "OK";                    break;
    case 404: errMsg = "Objekt nicht gefunden"; break;
    default:  errMsg = "unbekannter Fehler";    break;
  }

  /* -- Header -- */
  SendHeader(sout, error + " " + errMsg, "", 0, null);

  /* -- Fehlermeldung -- */
  sout.println("<HTML>");
  sout.println("<HEAD>");
  sout.println("<TITLE>Fehler " + error + ": " + errMsg + "</TITLE>");
  sout.println("</HEAD>");
  sout.println("<BODY>");
  sout.println("<H1>Fehler " + error + ": " + errMsg + "</H1>");
```

```
  sout.println("</BODY>");
  sout.println("</HTML>");
  sout.flush();

}

public void run() {

  /* -- HTTP-Befehle -- */
  final int CMD_GET  = 1;
  final int CMD_POST = 2;

  String ctt;  /* Content-Type         */
  int    cmd;  /* gesendeter Befehl    */
  String ext;  /* Extension d. lok. N. */
  String fn;   /* lokaler Dateiname    */
  File   f;    /* lokale Datei         */
  String ip;   /* IP Adresse d. Client */
  String line; /* gesendete Zeile      */
  Date   lm;   /* letzte Änderung (last modified) der lokalen Datei */
  int    port; /* Port d. Client       */
  String url;  /* gewünschte URL       */

  /* -- Variablen initialisieren -- */
  cmd  = 0;
  ip   = "";
  port = 0;
  url  = "";
```

```
try {

  /* -- Variablen setzen -- */
  ip   = this.s.getInetAddress().getHostAddress();
  port = this.s.getPort();

  /*****************************/
  /* Gesendete Daten einlesen */
  /*****************************/
  BufferedReader sin = new BufferedReader(
                       new InputStreamReader(this.s.getInputStream()));
  line = sin.readLine();
  while (line.length() > 0) {

    /* -- Trace ausgeben -- */
    //PrintLog(ip, port, line);

    /* -- Verarbeite GET-Befehl -- */
    if (line.startsWith("GET")) {
      /* -- Befehl merken -- */
      cmd = CMD_GET;
      /* -- URL ermitteln -- */
      url = "";
      int endOfUrl = line.indexOf(' ', 4);
      if (endOfUrl > 4) url = line.substring(4, endOfUrl);
    }

    line = sin.readLine();

  }
```

```
/**********************************/
/* Gesendeten Befehl verarbeiten */
/**********************************/
if (cmd == CMD_GET) {

  /* -- Variable url enthält den angeforderten URL -- */
  PrintLog(ip, port, "GET " + url);

  /* -- lokalen Dateinamen ermitteln -- */
  f = new File(this.rootDir + url);
  if (f.isDirectory()) f = new File(f + "default.htm");

  /* -- Extension ermitteln -- */
  ext = "";
  fn  = f.getAbsolutePath();
  int startOfExt = fn.lastIndexOf('.');
  if ((startOfExt > fn.lastIndexOf('/')) && (startOfExt > 0))
    ext = fn.substring(startOfExt + 1);
  /* Anmerkung: Wenn der Dateiname zum Beispiel ab.cd/xy lautet, dann bekommt
   * startOfExt den Wert 2 und als Extension würde cd/xy herauskommen, was
   * natürlich falsch ist. Die zusätzliche Bedingung
   * startOfExt > lastIndexOf('/') verhindert, daß eine Extension eines
   * Verzeichnisnamens herangezogen wird. */

  /* -- Content-Type ermitteln -- */
  ctt = "";
  if (ext.equalsIgnoreCase("htm") || ext.equalsIgnoreCase("html"))
    ctt = "text/html";
  else if (ext.equalsIgnoreCase("gif"))
    ctt = "image/gif";
```

```
        else if (ext.equalsIgnoreCase("jpg") || ext.equalsIgnoreCase("jpeg"))
          ctt = "image/jpeg";
        else if (ext.equalsIgnoreCase("txt") || ext.equalsIgnoreCase("text"))
          ctt = "text/plain";

        /* -- HTML-Seite zurücksenden -- */
        PrintWriter sout = new PrintWriter(this.s.getOutputStream());
        if (f.exists() && f.canRead()) {
          lm = new Date(f.lastModified());
          SendHeader(sout, "200 OK", ctt, f.length(), lm);
          SendFile(f, this.s.getOutputStream());
          PrintLog(ip, port, f.getAbsolutePath() + " gesendet.");
        } else { /* HTML-Objekt existiert nicht */
          SendError(sout, 404);
        }

      }

    } catch(IOException e) {
      System.out.println(ip + ":" + port + " IOException");

    } finally {
      /* -- Verbindung schließen -- */
      try {
        this.s.close();
      } catch(IOException ioe) {
      }
    }
  }
}
```

```
public class WWWServer {
  /* -- Konstanten -- */
  final static int PORT = 80;

  /* -- Hauptprogramm -- */
  public static void main(String[] args) {
    ServerSocket ss = null;     /* Server Socket                */
    Socket       cs = null;     /* Client Socket                */
    String       rootDir = "";  /* Wurzelverzeichnis des Servers */

    /* -- Startmeldung ausgeben -- */
    System.out.println();
    System.out.println("WWWServer - Einfacher WWW-Server");
    System.out.println("(C)opyright by Peter René Dietmüller, 1998-99");
    System.out.println();

    /* -- Parameter verarbeiten -- */
    if (args.length != 1) {
      System.out.println("Sie haben keinen oder zu viele Parameter angegeben.");
      System.out.println();
      System.out.println("Aufruf: java WWWServer <RootDirectory>");
      System.out.println();
      return;
    }
    rootDir = args[0];
    if (!(new File(rootDir)).exists()) {
      System.out.println("Das Verzeichnis " + rootDir + " existiert nicht.");
      System.out.println();
      return;
    }
```

```
    /* -- Server starten -- */
    System.out.println("Starte WWW-Server auf Port " + PORT +
                       " ... (mit Strg+C beenden)");

    try {
      /* -- Warte auf eingehende Verbindungen -- */
      ss = new ServerSocket(PORT);
      while (true) {  /* Endlosschleife */
        cs = ss.accept();
        (new WWWVerbindung(cs, rootDir)).start();
      }

    } catch(IOException e) {
      System.out.println("Port " + PORT + " ist bereits belegt.");

    } finally {
      if (ss != null) {
        try {
          ss.close();
        } catch(IOException ioe) {
        }
      }
      System.out.println("Beende WWW-Server ...");

    }
  }
}
```

Test WWW-Server

```
WWWServer - Einfacher WWW-Server
(C)opyright by Peter René Dietmüller, 1998-99

Starte WWW-Server auf Port 80 ... (mit Strg+C beenden)
127.0.0.1:1360, GET /
127.0.0.1:1360, y:\http/default.htm gesendet.
127.0.0.1:1361, GET /new.htm
127.0.0.1:1361, y:\http/new.htm gesendet.
127.0.0.1:1362, GET /pow/pow.gif
127.0.0.1:1363, GET /CDrag.jpg
127.0.0.1:1362, y:\http/pow/pow.gif gesendet.
127.0.0.1:1363, y:\http/CDrag.jpg gesendet.
127.0.0.1:1364, GET /aushaenge/981210_Diplomarbeitsthema_XSAT.html
127.0.0.1:1364, y:\http/aushaenge/981210_Diplomarbeitsthema_XSAT.html gesendet.
127.0.0.1:1365, GET /aushaenge/981210_Diplomarbeitsthema_VPN.html
127.0.0.1:1365, y:\http/aushaenge/981210_Diplomarbeitsthema_VPN.html gesendet.
127.0.0.1:1366, GET /aushaenge/981210_Diplomarbeitsthema_Einfuehrung_von_E-
Commerce.html
127.0.0.1:1366, y:\http/aushaenge/981210_Diplomarbeitsthema_Einfuehrung_von_E-
Commerce.html gesendet.
(... einige Zeilen gelöscht ...)
127.0.0.1:1378, y:\http/codeddrag/download_e.htm gesendet.
127.0.0.1:1379, GET /codeddrag/cd2102ee.exe
127.0.0.1:1379, y:\http/codeddrag/cd2102ee.exe gesendet.
```