PERANCANGAN APLIKASI FILE TRANSFER PROTOCOL DENGAN MENGGUNAKAN BAHASA PEMROGRAMAN JAVA

SOFIANSYAH – L2F 301 494
Jurusan Teknik Elektro Universitas Diponegoro
Jl. Prof. Sudharto, Tembalang, Semarang
Email : Sofiansyahroni@Yahoo.com

ABSTRAK

Seiring dengan berkembanganya teknologi informasi, serta sangat mudahnya untuk mendapatkan suatu informasi di kalangan masyarakat. Internet merupakan salah satu cara untuk mendapatkan informasi sesuai dengan kebutuhan yang diinginkan, namun tak lepas dari itu semua, dalam mengelola suatu informasi, mereka dituntut untuk mengelola sistem jaringan informasinya agar dapat dijauhkan atau dihindarkan dari orang-orang yang tidak bertanggung jawab. Keamanan jaringan untuk komputer sangatlah diperlukan karena dapat mencegah terjadinya pencurian, penghapusan, pengerusakan dan pemanipulasian pada data. Salah satu cara untuk mengamankan jaringan komputer dilakukan dengan cara mengatur alamat jaringan atau ip address dan port pada komputer, sehingga alamat jaringan atau ip address dan port pada jaringan komputer hanya diketahui oleh orang yang berhak. Pengaturan kombinasi antara port dan ip address disebut dengan socket. Socket dapat dikatakan sebagai endpoint dari komunikasi dua arah antar aplikasi.

Salah satu aplikasi pada Application Layer adalah File Tranfer Protocol (FTP), karena aplikasi FTP berbasiskan paradigma Client/Server, maka untuk menggunakan FTP harus terdapat sebuah server yang bertugas sebagai FTP server dan yang lainnya adalah FTP client. FTP server bertugas untuk menangani permintaan (request) FTP dari client. FTP client berfungsi untuk melakukan permintaan (request) ke FTP server. Permintaan ini dapat berupa request untuk mengambil file dari server, atau menaruh file ke server.

I. PENDAHULUAN

1.1. Latar Belakang

Java merupakan bahasa pemrograman berorientasi objek yang serbaguna. Disamping itu Java juga menyediakan sejumlah perluasan yang mendukung dalam pengembangan aplikasi *client*/*server* terhadap suatu jaringan lokal maupun jaringan yang luas. Salah satu kelebihan pada Java terletak pada *multiplatform*-nya yang bisa berjalan di atas *platform* apa saja (Windows, Unix, Linux, Macintosh dan sebagainya). Java juga mendukung protokol TCP/IP maupun UDP

Dalam penggunaan jaringan komputer, khususnya yang berhubungan dengan dunia luar, sistem keamanan pada jaringan komputer sangat diperlukan. Keamanan pada sistem jaringan komputer dilakukan pada saat melakukan koneksi pada suatu jaringan dengan menggunakan kombinasi *ip address* dan *port* jaringan. Tujuan dari kombinasi antara *ip address* dan *port* jaringan adalah untuk menjaga keamanan komputer *server* dari gangguan-gangguan dari luar. Kombinasi antara *ip address* dan *port* jaringan disebut dengan *socket*, sebuah *socket* berfungsi sebagai *endpoint* dari komunikasi dua arah antar aplikasi yang digunakan.

Aplikasi pada FTP (*File Transfer Protocol*) adalah protokol sekaligus program yang dapat digunakan untuk melakukan operasi file dasar pada *host remote* dan untuk mentransfer file antar *host*. Sebagai sebuah program FTP dapat dioperasikan pengguna untuk melakukan perintah - perintah file secara manual. *File transfer protocol* juga dapat digunakan sebagai protokol oleh aplikasi yang membutuhkan pelayanan file.

1.2. Tujuan

Tujuan penyusunan Tugas Akhir ini adalah untuk membuat program *socket* untuk aplikasi *File Transfer Protocol* dengan menggunakan bahasa pemrograman Java. Diharapkan dengan adanya program aplikasi tersebut sistem keamanan pada komputer dapat terlaksana.

1.3. Pembatasan Masalah

Dalam pembuatan Tugas Akhir ini, Penulis memberikan batasan-batasan masalah yang akan dibahas adalah sebagai berikut :

1. Pembahasan difokuskan pada hasil dari konektivitas jaringan menuju *server file transfer protocol* mulai membuka koneksi ke *host* yang dituju, mendapatkan *stream* (input/output) dari koneksi, mengetahui apa yang dilakukan oleh input atau output *stream*, menutup *stream-stream* tersebut, menutup koneksi dengan *host*.
2. Aplikasi *client file transfer protocol* yang digunakan meliputi aplikasi memindah direktori dari, membuat direktori baru, menggandakan file, melihat isi direktori.
3. Pada perancangan aplikasi ftp yang akan dibuat tidak berbasiskan GUI (*Graphical User Interface*).

II. LANDASAN TEORI

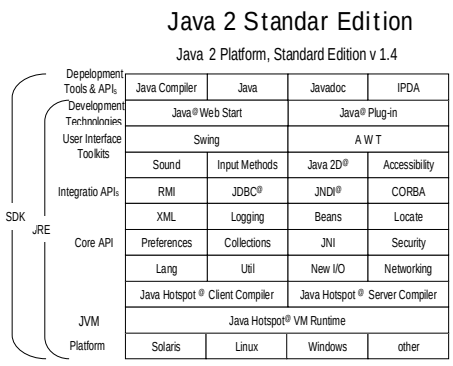
2.1 Perkembangan Java

Pada dasarnya, ada ada berbagai macam *platform* tempat aplikasi-aplikasi perangkat lunak (*software*) untuk dieksekusi seperti Microsoft Windows, Unix, Linux, Netware, Macintosh, dan OS/2, akan tetapi aplikasi-aplikasi yang berjalan pada suatu *platform* (misalnya Windows) tidak akan bisa

dijalankan di *platform* lain (misalnya Linux) tanpa usaha kompilasi ulang, bahkan dengan malakukan pengubahan kode program. Java *platform* merupakan perangkat lunak yang menjadi mesin *virtual* bagi aplikasi-aplikasi Java untuk dieksekusi, oleh sebab itu aplikasi Java tidak perlu dikompilasi ulang jika telah dikompilasi di suatu *platform* dan akan dijalankan di *platform* yang berbeda dengan platform saat dikompilasi, karena aplikasi Java dijalankan di atas *Java Virtual Machine* (JVM).

2.2 Java 2 Platform, Standard Edition (J2SE™)

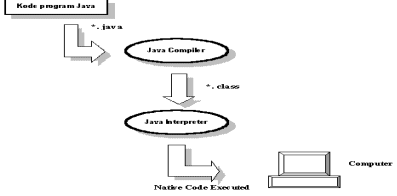
Platform ini digunakan untuk menjalankan dan mengembangkan aplikasi Java pada level *Personal Computer* (PC). Java juga mendukung protokol TCP/IP dan UDP. Bentuk Java 2 *Platform, Standar Edition* dapat dilihat pada Gambar 2.1 di bawah ini.



Gambar 2.1. Java 2 Platform Standar Edition

2.2.1 JVM (Java Virtual Machine)

Kode program Java dapat ditulis menggunakan berbagai macam teks editor seperti Notepad, Textpad maupun JCreator dan lain sebagainya yang bisa menghasilkan ekstensi Java (*.Java). Selain itu Java juga menyediakan alat *compiler* yang digunakan untuk mengkompilasi kode program Java, alat ini dirancang untuk menghasilkan kode program yang netral terhadap semua arsitektur perangkat keras (*hardware*) yang disebut sebagai *java bytecode* (*.class). *Java Virtual Machine* (JVM) menjadi bagian dari teknologi Java yang menyediakan media untuk menjalankan aplikasi Java (*java bytecode*). *Java bytecode* dapat dianggap sebagai kode-kode mesin dari JVM. Selanjutnya JVM akan menginterpretasikan kode-kode tersebut menjadi kode native atau kode mesin dari arsitektur yang bersangkutan. Berikut ini adalah gambar proses kompilasi program Java.



Gambar 2.2. Proses Kompilasi Program Java

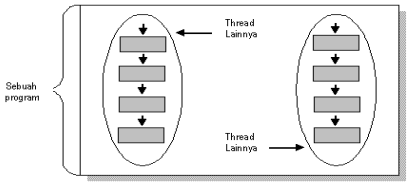
2.2.2 Java API (Application Programming Interface)

Java API adalah sekumpulan paket Java yang berisi koleksi banyak *class* dan *interface* dasar yang harus dipakai untuk pemrograman menggunakan bahasa Java. Aplikasi-aplikasi Java API adalah sebagai berikut :

a. *Applet*
b. *Java Networking*
c. *Java Database Connectivity* (JDBC)
d. *Java Security*
e. *Java Swing*
f. Java IDL + CORBA
g. *Java Server Pages* (JSP)
h. *Java Card*

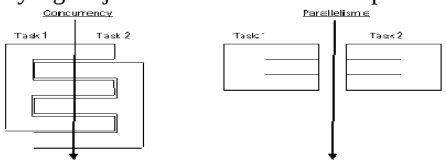
2.2.3 Multithreading

Thread (sering disebut juga *light wight process* atau *execution content*) adalah sebuah *single sequential flow of control* atau sebuah pengaturan alur program di dalam sebuah program, dan secara sederhana *thread* adalah sebuah sub program yang berjalan di dalam sebuah program [7]. Gambar di bawah ini menjelaskan bagaimana sebuah program terdiri dari bebarapa *thread*.



Gambar 2.3 Dua Thread Dalam Satu Program

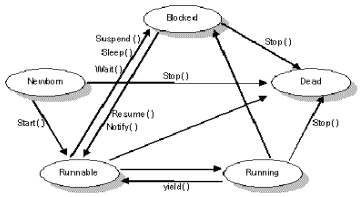
Sebuah program yang mempunyai beberapa *thread* yang berjalan secara konkuren atau paralel disebut dengan *multithreaded*. *Thread* akan berjalan secara konkuren adalah apabila program yang digunakan berjalan di atas mesin dengan *single processor* (dengan mengeksekusi secara bergantian dari satu *thread* ke *thread* lainnya), sedangkan *thread* yang berjalan secara paralel adalah apabila sebuah program yang digunakan berjalan di atas mesin dengan *multi processor* (masing-masing *thread* berjalan secara bersamaan di prosessor-prosessor yang terpisah). Gambar di bawah ini menjelaskan perbedaan antara *thread* yang berjalan secara konkuren dan paralel.



Gambar 2.4 Thread Berjalan Secara Konkurens Dan Paralel

Pada JVM terdapat *thread scheduler* atau penjadwalan *thread* yang berfungsi untuk menentukan *thread* mana yang akan beroperasi pada selang waktu tertentu. Pada proses *preemptive multithreading*, suatu *thread* dengan prioritas tinggi bisa segera masuk dan menginterupsi *thread* yang sedang beroperasi, hal ini sangat menguntungkan untuk membuat aplikasi *real*

time. Sedangkan pada proses *non-preemptive multithreading* (sering disebut dengan *cooperative multithreading*), *thread* yang sedang beroperasi tidak bisa diinterupsi sampai dia menyelesaikan tugasnya. Daur hidup sebuah *thread* dapat dilihat pada gambar dibawah ini.



Gambar 2.5 State-state Dari Thread

Dari gambar diatas bahwa sebuah *thead* memiliki lima buah *state* atau keadaan, ke lima *state* tersebut mempunyai fungsi dan tujuannya masing-masing. Tujuan dan fungsi ke lima *state* itu adalah sebagai berikut :

1. *Newborn*
2. *Runnable*
3. *Running*
4. *Dead*
5. Blocked

2.3 Java Networking

Paket ini juga mendukung protokol TCP/IP dan UDP. *Class – class* yang digunakan dalam protokol TCP adalah URL, *URLConnection*, *Socket*, *ServerSocket*, sedangkan untuk *class* yang menggunakan protokol UDP adalah *DatagramPacket*, *DatagramSocket*, *MulticastSocket*.

2.3.1 Socket

Socket merupakan suatu kombonasi antara alamat IP (Internet Protocol) dan *port*. *Socket* dapat dikatakan pula sebagai *endpoint* dari komunikasi dua arah antar aplikasi yang digunakan [7]. Ada dua jenis konstruktor yang dapat digunakan untuk membuat *socket*, kedua konstruktor itu antara lain :

- `Socket (String host, int port)`
- `Socket (InetAddress address, int port)`

Selain dipergunakan untuk melakukan hubungan dengan *server*, *Socket* dapat juga dipergunakan untuk memeriksa atau menginformasikan tentang suatu alamat dan nomor *port* yang dipergunakan, dengan menggunakan *method* berikut ini:

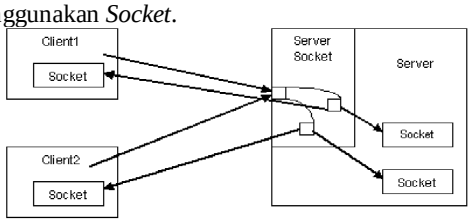
- `getInetAddress()`
- `getPort()`
- `getLocalPort()`

Class InetAddress berfungsi untuk mengenkapsulasi alamat IP dan mendukung konversi alamat IP dan nama *host*, *class* ini tidak memiliki konstruktor yang nampak. *Class InetAddress* memiliki tiga buah *method*, ketiga *class* itu antara lain :

- `getHostName()`
- `getAddress()`
- `toString()`

2.3.2 ServerSocket

Class ServerSocket biasanya dipergunakan untuk membuat aplikasi *server*. Berikut ini adalah cara melakukan koneksi antara *client* dengan *server* dengan menggunakan *Socket*.



Gambar 2.6 Koneksi Socket Menggunakan IP Address Dan Port

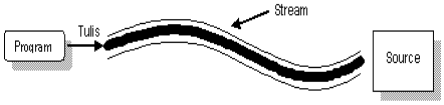
Class ServerSocket mempunyai dua buah konstruktor yang mengidentifikasikan jumlah *port* yang diinginkan untuk menerima hubungan, dan juga dapat dipilih menunggu seberapa lama waktu yang digunakan untuk menunggu suatu *port* tidak digunakan. Berikut ini adalah bentuk kedua konstruktor pada *socket server* :

- `ServerSocket (int port)`
- `ServerSocket (int port, int count)`

2.4 Java I/O (Input/Output)

Paket ini mendukung untuk proses pembacaan dan penulisan stream data. Untuk lebih jelasnya dapat dilihat pada Gambar 2.7 dan Gambar 2.8 di bawah ini.

Gambar 2.7 Skema Pembacaan Stream Data



Gambar 2.8. Skema Penulisan Stream Data.

2.4.1 Byte Stream

Kelompok *class* dan *interface* dalam *byte stream* dipergunakan untuk menangani proses baca atau tulis *byte* data, misalnya untuk membaca dan menulis data biner

2.4.2 Character Stream

Kelompok *class* dan *interface* dalam *character stream* digunakan untuk menangani proses baca atau tulis himpunan karakter Unicode.

2.4.3 Data Sink Stream

Kelompok *class* dan *interface* dalam *data sink stream* digunakan untuk menangani proses baca atau tulis dari suatu *stream* khusus, misalnya *file*, *piped* atau *string*..

2.4.4 Processing Stream

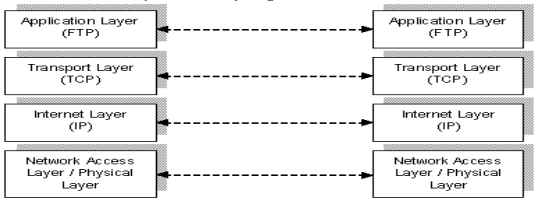
Kelompok *class* dan *interface* dalam *processing stream* digunakan untuk melakukan aksi tertentu pada sebuah *stream*, misalnya *buffering*, *encoding*, *printing*.

2.5 FTP (File Tranfer Protocol)

File transfer protocol merupakan salah satu aplikasi dari TCP/IP yang banyak digunakan untuk memindahkan atau mengcopy *file* dari komputer satu ke komputer lainnya. Aplikasi ini adalah aplikasi yang telah dikembangkan sejak awal perkembangan Internet. *File transfer protocol* adalah protokol sekaligus program yang dapat digunakan untuk melakukan operasi *file* dasar pada *remote host* dan untuk mentransfer *file* antar *host*. Sebagai sebuah program, *file transfer protocol* dapat dioperasikan pengguna untuk melakukan perintah *file* secara manual [3].

2.5.1 Prinsip Kerja FTP

File transfer protocol memakai TCP sebagai *transport protocol* untuk menyediakan hubungan *end to end* yang nyata dan pada *Internet Layer* terjadi komunikasi dengan menggunakan *Internet Protocol*. Pengguna yang memulai melakukan hubungan dengan dengan *server* disebut dengan *client*, sedangkan yang memberikan pelayanan pada *client* disebut dengan *server*. Berikut ini adalah gambar komunikasi yang dilakukan oleh *file transfer protocol*.



Gambar 2.15 Bentuk Komunikasi File Transfer Protocol.

2.5.2 Operasi-operasi dalam FTP

Seperti telah dijelaskan sebelumnya, *file transfer protocol* adalah protokol sekaligus program yang dapat digunakan untuk melakukan operasi *file* dasar pada *remote host* dan untuk mentransfer *file* antar *host*. Sebagai sebuah program, *file transfer protocol* dapat dioperasikan pengguna untuk melakukan perintah *file* secara manual. Operasi – operasi yang dipergunakan dalam *file transfer protocol* adalah sebagai berikut : Dir, Get, Put, Cd, Bye, Mkdir, Delete, Help, About.

III. PERANCANGAN DAN IMPLEMENTASI

3.1 Java 2 Platform, Standard Edition (J2SE™)

3.1.1 Java 2 Software Development Kit (JDK)

Java 2 Software Development Kit digunakan untuk mengembangkan dan mencoba program Java yang telah dibuat dengan menggunakan teks editor. Untuk meng-*install* atau memasang *Java Development Kit* (JDK) ini diperlukan spesifikasi sebagai berikut :

- Sistem operasi Windows 98, Windows NT 4.0/2000 dan Windows XP.
- Prossesor minimum Pentium 166 MHz
- RAM minimum 32 Mbytes.
- Ruang H*arddisk* minimum 65 Mbytes.

3.1.2 Mengatur PATH dan CLASSPATH

Cara melakukan pengaturan PATH dan CLASSPATH pada sistem operasi dengan menggunakan *MS Dos-prompt* maupun Windows dilakukan dengan mengubah isi dari file *C:\autoexec.bat*, berikut ini adalah cara penulisannya :

a. Pengaturan PATH
   PATH=%PATH%;c:\progra~1\java\bin;
b. Pengaturan CLASSPATH
   SET CLASSPATH = c:\progra~1 \java\
   classes.zip;

3.1.3 Program-program pada JDK

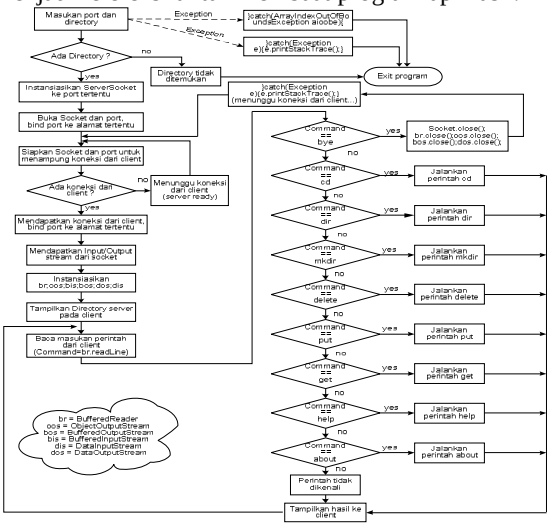
JDK (*Java Development Kit*) merupakan *file-file executable file* yang berada dalam satu paket JVM. Perintah-perintah di bawah ini merupakan beberapa perintah yang paling sering digunakan dalam Java. Daftar [*options*] bisa didapat jika hanya mengetikkan perintah tersebut tanpa menggunakan parameter apapun.

a. *javac*
   Sintaks : `javac [options] <nama file>.java`
b. *java*
   Sintaks : `java [options] <nama file yang berektensi .class> (tanpa akhiran .class)`
c. *appletviewer*
   Sintaks : `appletviewer [options] <nama file HTML>.html (atau htm)`

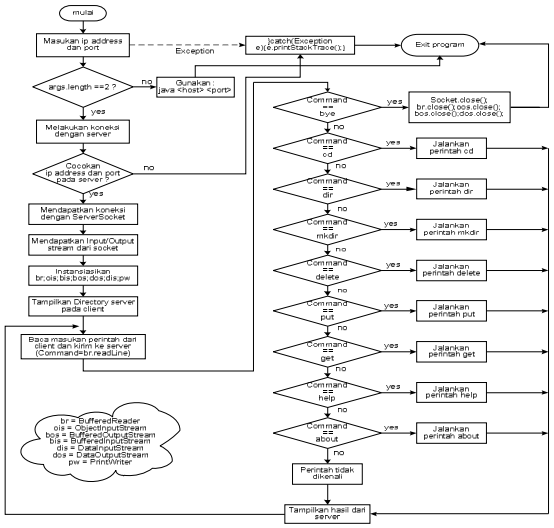
3.2 Perancangan Program

3.2.1 Flowchart Program

Berikut ini adalah *flowchart file* ftp yang menjadi referensi untuk membuat program aplikasi :



Gambar 3.3 Flowchart File Server



Gambar 3.4 Flowchart File Client

3.2.2 File Server

Tahapan-tahapan yang dilakukan dalam pembuatan *server file* ftp adalah dimulai dengan mengidentifikasikan *port* dan direktori mana yang dapat diakses oleh *client*, membuka koneksi dengan *client*, menerima koneksi dari *client*, mengetahui apa yang akan dilakukan oleh *client*, menutup *stream-stream* dan menutup koneksi dengan *client*.

3.2.3 File Client

Tahapan-tahapan dalam membuat *file client* hampir sama dengan tahapan pembuatan *file server*, akan tetapi dalam pembuatan *file client* saat membuat *Socket client*, ia diinstansiasikan dengan merujuk alamat *ServerSocket*, sedangkan untuk tahapan yang lainnya sama dengan pembuatan File Server.

IV. HASIL DAN PEMBAHASAN

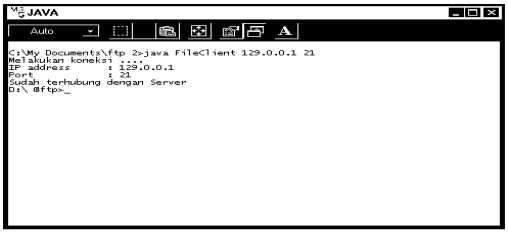
Seperti telah dijelaskan pada bab sebelumnya, bahwa program aplikasi ini dibuat tanpa menggunakan tampilan grafis. Alasan mengapa program tidak dibuat dalam bentuk grafis karena program yang akan diaplikasikan berupa transfer-transfer file antara file server dengan file client.

4.1 Hasil

Berikut ini adalah hasil koneksi antara file server dan file client dengan menggunakan alamat ip dan port. Pertama-tama file server dijalankan dengan menggunakan port dan direktori server yang akan diaktifkan untuk dihubungkan dengan client, seperti terlihat pada gambar di bawah ini.

Gambar 4.1 File Server Telah Dijalankan

Setelah file server telah diaktifkan, selanjutnya tiba saatnya bagi file client untuk melakukan koneksi dengan file server menggunakan alamat ip dan port yang telah diaktifkan oleh file server, untuk lebih jelasnya dapat dilihat pada gambar di bawah ini.



Gambar 4.2 File Client Melakukan Koneksi dengan File Server

Untuk penerimaan koneksi dari file client oleh file server dapat di lihat pada gambar di bawah ini.

Gambar 4.3 File Server Menerima Koneksi Dari File Client

4.2 Pembahasan

Sebagaimana telah dijelaskan sebelumnya dalam pembatasan masalah bahwa yang akan dibahas adalah bagaimana terjadi koneksi antara file server dengan file client serta aplikasi yang akan dipergunakan antara lain mengambil file dari server, menaruh file ke dalam server, menghapus file, membuat direktori baru, melihat isi direktori dan memindahkan direktori.

4.2.1 Menjalankan Direktori Server

Sebelum file server dijalankan, terlebih dahulu server harus mengetahui direktori mana yang akan diaktifkannya. Direktori ini nantinya yang akan dihubungkan langsung oleh server kepada client, dengan menggunakan direktori yang diaktifkan oleh server, sehingga client bisa melakukan operasi file antara lain mengambil file, menaruh file, menghapus file, membuat direktori baru, melihat isi direktori, memindahkan direktori pada server. Berikut ini adalah bentuk penulisan untuk menjalankan file server :

```
C:\My Documents\javac FileServer.java
C:\My Documents\java FileServer<port><directori>
```

Untuk menghubungkan file client dengan file server dapat ditulis dengan menggunakan perintah beikut ini :

```
C:\My Documents\javac FileClient.java
C:\My Documents\java FileClient<ip><port>
```

4.2.2 Konektivitas Client Dengan Server

Dari hasil gambar 4.1 diatas dapat dilihat bahwa pertama kali file server dijalankan, server akan menunggu datangnya koneksi dari client untuk melakukan hubungan dengan server. Class yang

digunakan oleh server untuk menunggu datangnya koneksi dari client menggunakan class ServerSocket. Untuk lebih jelasnya dapat dilihat pada perintah di bawah ini.

```
.....
ServerSocket ss = new ServerSocket(sPort);
.....
```

Dari perintah diatas dapat dilihat bahwa class "ServerSocket" ini berfungsi untuk membangun socket server yang mempunyai tujuan untuk menunggu datangnya koneksi dari client. Perintah "new" berfungsi untuk menciptakan instans baru dari suatu class "ServerSocket" yang dialokasikan pada "(sPort)", dalam pelaksanaannya penamaan instansiasi pada class "ServerSocket" adalah "ss".

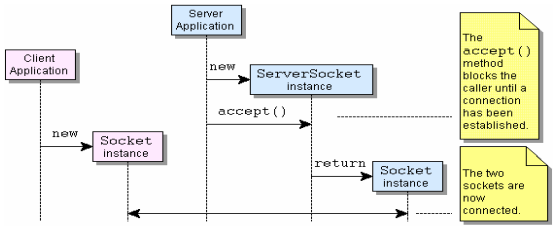
Dari pihak client, cara yang digunakan untuk melakukan hubungan dengan server menggunakan class Socket. Perintah "new" berfungsi untuk menciptakan instans baru dari suatu class Socket yang dialokasikan pada argumen "(args[])", "args[]" merupakan array instans dari suatu class String, dalam pelaksanaannya penamaan instansiasi pada class Socket adalah "s". Penggunaan class Socket pada client menggunakan dua buah argumen "(args[0], args[1])" penambahan character sesudah penulisan nama program pada aplikasi Java. Argumen pertama "(args[0])" menunjukkan alamat ip yang digunakan oleh server, sedangkan argumen kedua "(args[1])" menunjukkan nomor port yang diaktifkan pada server, dalam pelaksanaannya nomor port yang digunakan oleh file ftp menggunakan port 21. Untuk lebih lebih jelasnya dapat dilihat pada perintah dibawah ini

```
.....
Socket s = new Socket(args[0], Integer.
parseInt(args[1]));
.....
```

Perintah "Integer. parseInt(args[1])" berfungsi untuk menghasilkan nilai String yang berkaitan dengan "args[1]". Setelah kedua argumen pada client ketika dijalankan telah sesuai dengan alamat ip dan port yang sedang diaktifkan oleh server, maka server akan menerima datangnya koneksi dari client dengan menggunakan perintah dibawah ini.

```
.....
Socket s = ss.accept();
.....
```

Pada perintah diatas dapat dilihat bahwa class Socket yang telah diinstansiasikan dengan nama "s" mengacu kepada class ServerSocket yang telah diinstansiasikan dengan nama "ss" menggunakan perintah "accept()". Perintah "accept()" dalam Java berfungsi untuk menyetujui bahwa telah terjadi hubungan antara client dengan server dan jika tidak terdapat kesalahan dalam melakukan hubungan dengan server yang dilakukan oleh client dengan menggunakan alamat ip dan port. Untuk lebih jelasnya hubungan antara client dengan server dapat dilihat pada gambar di bawah ini.



Gambar 4.17 Prinsip Kerja Hubungan Antara Client Dengan Server

Dari gambar diatas dapat dilihat bahwa ketika kedua belah pihak yaitu client dan server setelah diinstansiasikan, kemudian server menerima datangnya koneksi dari client dengan menggunakan perintah "accept()". Setelah koneksi diterima oleh server, class ServerSocket akan diinstansiasikan secara otomatis dan akan membuat class baru yaitu class Socket, kemudian class ini yang akan melakukan hubungan dengan client secara terus menerus sampai adanya pemutusan hubungan oleh client.

4.2.3 Penggujian Aplikasi File FTP

4.2.3.1 Perintah Dir

Perintah "dir" berfungsi untuk melihat isi direktori yang berada di dalam server dan kemudian ditampilkan pada client. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>Dir
D:\@ftp>Dirc
```

4.2.3.2 Perintah CD

Perintah "cd" berfungsi untuk berpindah direktori dari direktori yang satu menuju ke direktori yang lain. Direktori yang dipergunakan pada file client merupakan direktori yang berada pada direktori file server. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>cd <nama direktori>
```

4.2.3.3 Perintah Put

Perintah "put" merupakan perintah yang berfungsi untuk menaruh file yang berada pada file client ke dalam direktori yang berada pada file server. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>put <nama file.ekstensi>
```

4.2.3.4 Perintah Get

Perintah "get" berfungsi untuk mengambil file yang berada pada file di direktori server, kemudian file tersebut ditaruh ke dalam direktori client. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>get <nama file.ekstensi>
```

4.2.3.5 Perintah MkDir

Perintah "mkdir" berfungsi untuk membuat direktori baru yang berada di dalam server. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>mkdir <nama direktori>
```

4.2.3.6 Perintah Delete

Perintah "delete" berfungsi untuk menghapus file. Dalam penggunaannya perintah "delete" dibagi menjadi dua buah fungsi, yang pertama untuk

menghapus file yang berada di server dan yang kedua untuk menghapus file yang berada di client. Perintah yang digunakan untuk menghapus file di server menggunakan perintah "deletes", sedangkan untuk menghapus file yang berada di dalam client menggunakan perintah "deletec". Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>deletes <nama file.ekstensi>
D:\@ftp>deletec <nama file.ekstensi>
```

4.2.3.7 Perintah Bye

Perintah "bye" merupakan perintah berfungsi untuk melakukan pemutusan hubungan antara client dengan server. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>bye
```

4.2.3.8 Perintah User

Perintah "user" berfungsi untuk mengetahui tentang alamat ip yang digunakan untuk melakukan hubungan dengan server dan berada pada port berapa server bekerja. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>user
```

4.2.3.9 Perintah About

Perintah "about" berfungsi untuk mengetahui tentang pembuatan dari program aplikasi tersebut. Berikut ini adalah bentuk penulisannya :

```
D:\@ftp>about
```

4.2.3.10 Perintah Help

Perintah "help" berfungsi untuk memberikan informasi tentang fasilitas yang digunakan untuk menjalankan program aplikasi. Berikut ini adalah bentuk penulisannya:

```
D:\@ftp>help
```

4.2.3.11 Penangganan Perintah Yang Tidak Dikenali

Untuk penangganan kesalahan penggunaan perintah pada program aplikasi menggunakan perintah "else". Perintah "else" berfungsi untuk menangani jika terjadinya kesalahan dalam penggunaan aplikasi-aplikasi file dasar.

4.2.4 Penanganan Error

Program ini mempunyai sistem penangganan error handling untuk menangani kesalahan-kesalahan yang terjadi. Penanganan error handling terjadi ketika pengguna tidak memenuhi persyaratan untuk menjalankan file server

V. PENUTUP

5.1 Kesimpulan

Setelah melakukan pembuatan program dan melakukan pengujian aplikasi yang digunakan, maka penulis dapat menarik beberapa kesimpulan sebagai berikut :

1. Class Socket berfungsi untuk membuat suatu *socket* yang menghubungkan *host* lokal ke *host* dan *port* yang diinginkan serta untuk melakukan hubungan dengan server.
2. Class ServerSocket berfungsi untuk membuat aplikasi server pada port yang ditentukan dan siap dihubungkan dengan aplikasi client.
3. Pada saat menjalankan program, terlebih dahulu file server dijalan dengan menggunakan sintak java <port><directory>. Setelah server sudah berjalan, kemudian tiba saatnya untuk menjalakan file client dengan menggunakan sintak java <alamat ip><port>. Port yang digunakan oleh client merupakan merupakan nomor port yang sudah diaktifkan oleh server dan alamat ip merupakan tempat saat server sedang berjalan di dalam komputer.
4. Pengujian aplikasi meliputi operasi mengambil file (get), menaruh file (put), membuat direktori baru (mkdir), menghapus file di server dan di client (deletes dan deletec), melihat isi direktori (dir) dan memindah direktori (cd) dan pemutusan hubungan dengan server (bye).
5. Penanganan kesalahan atau error handling pada program menggunakan metode Exception. Metode ini berfungsi menangani event yang terjadi ketika program menemui kesalahan saat instruksi program dijalankan dan ketika kesalahan terjadi, maka program secara otomatis akan dilempar ke sebuah objek yang disebut exception. Metode exception yang digunakan adalah IOException dan ArrayIndexOutOfBoundsException.
6. Penggunaan IOException berfungsi untuk menanggani kesalahan pada I/O yang telah terjadi. Sedangkan ArrayIndexOutOfBoundsException berfungsi untuk menangani kesalahan indeks array yang digunakan tidak valid atau sesuai dengan yang diinginkan.

5.2 Saran

Aplikasi ini hanya digunakan untuk mentransfer file antara client dengan server tanpa memperdulikan bentuk tampilan dari program yang dibuat. Untuk pengembangan lebih lanjut sebaiknya program dapat dibuat dengan menggunakan tampilan grafis.

DAFTAR PUSTAKA

1. Cornell, Gary and Horstmann, Cay S. Core Java Edisi Indonesia. Penerbit Andi Offset Yogyakarta. 1997.
2. Harold, Elliotte Rusty. Java Network Programming, 2nd Edition. O'Reilly & Associates, Inc. 2000.
3. Heywood, Drew. Konsep dan Penerapan Microsoft TCP/IP. Penerbit Andi Offset Yogyakarta. 1997.
4. Jackson, Jerry R. and McClellan, Alan L. Java By Example Edisi Indoseia. Penerbit Andi Offset Yogyakarta. 1996.
5. Naughton, Patrick. Java Handbook Konsep Dasar Pemrograman Java. Penerbit Andi Offset Yogyakarta. 1997.
6. Postel, Jon dan Reynolds, J, File Transfer Protocol, _____ ftp://ftp.isi.edu/innotes/rfc959.txt.

7. Purbo, W. Onno. dkk. Trik Pemrograman Java untuk Jaringan dan Internet. PT. Elexmedia Komputindo Kelompok Gramedia. Jakarta. 2000.
8. Tanembaum, Andrew S. Jaringan Komputer edisi Bahasa Indonesia jilid 1 dan 2. PT. Prenhallindo. Jakarta. 1996.
9. Wicaksono, Ady. Dasar-dasar Pemrograman Java 2. PT. Elexmedia Komputindo Kelompok Gramedia Jakarta. 2002.

SOFIANSYAH
Penulis dilahirkan di Jakarta, 03 Mei 1978. Saat ini penulis sedang menyelesaikan pendidikan S1 di Jurusan Teknik Elektro, Fakultas Teknik, Universitas Diponegoro. Kosentrasi yang diambil adalah Teknik Komputer & Informatika.

Semarang, September 2003
Disetujui
Dosen Pembimbing II

Agung Budi Prasetijo, ST. MIT.
NIP. 132 137 932