Nicolas Delerue
McGill University
308-573B
(Professor Ratzer)

A Java-Based FTP software

Winter 1998

Table of Content:

1FOREWORD..4

2A BRIEF OVERVIEW OF JAVA..5

2.1The Java Idea..5
2.2The Java Virtual Machine..6
2.3Some adaptations I did to the JDK..7

3A BRIEF OVERVIEW OF THE FTP STANDARD.....................7

3.1The Telnet standard and the Network Virtual Terminal............8
3.2The FTP standard...8
3.3Comparison between the NVT and the JVM...............................10

4THE SOFTWARE...10

5THIS IS STILL A WORK IN PROGRESS.................................15

6PORTABILITY...16

7CONCLUSION...17

8APPENDIX...18

8.1Some useful URLs and bibliography...18
8.2The source code..19
8.3Acknowledgements...71

1 Foreword

I often transfer data on the Internet and thus I often use FTP software. On my own computer (a PC running under Windows 95), I have found a few useful software, but none of them were freeware. Under UNIX the situation is worse, because the only FTP software installed on most of the UNIX systems has no graphical interface and is not really user friendly (you can not even delete a character you just typed). As in High Energy Physics (my main field of study), we sometimes have to transfer up to 10Go in a night, I thought it would be great to write a user-friendly FTP software.

I have learned a few computer languages, mainly physics oriented such as Fortran 77 and 90, more recently, I learned C here at McGill, but I wanted to learn a graphical language. Even if I had read about it, I had never had the opportunity of writing a program in Java. As Java is said to be portable on all kinds of computers, it meets the universality I want for my software. I also often writes Web sites, and therefore I wanted to learn Java. That is why I wrote my software in Java.

2 A Brief overview of Java

2.1 The Java Idea

Within the last few years, Java has become one of the most famous programming languages. This is due to some of the key ideas of the Java Language. One of its most important features is portability: one of the Java mottos is "*Write once, run everywhere*". Once you have written an application, you should be able to run it on any kind of computers and even on some other electronic devices powered by Java, such as cellular telephones, microwaves oven or VCR. This feature may seem not so important for people who use to write code only for PC's, but when you want to write code for UNIX systems, it is important to be sure that your application will run anywhere. Java provides you this, without even the need of testing your software on all these platforms!

Another important feature of Java is network-oriented Java is the first language to provide in its core many useful network tools such has direct addressing of a network socket through the URL of the remote host, detection of malformed URLs…

Java is also adapted to the multitask technologies and within the same application, some components can run independently of the others. These are called threads and provide more flexibility to the final product.

A last important feature of Java is that it is a graphical language that provides easy to use tools for both MS-Windows and X-Windows programming.

But as Java is a very recent language, it is still evolving, some adaptation are done every weeks an it is very important to often connect to Java's web site (see the Bibliography) to known if new bugs have been detected or fixed and to download new releases of the JDK when necessary (but this means a 10 Megs download!) or also to submit new bugs when I thought to have detected some...

I started with JDK 1.0 that was on a CD-ROM I got from a book, but soon I decided to move to JDK 1.1.0.

Later I decided to use powerful new graphical tools released by sun in February, and know as the JFC (Java Foundation Class) or the swing components but these required JDK 1.1.5, some I changed once more of compiler.

As I would like to implement a help to my software, I will need to use the JavaHelp system, but this will only be available with the release 1.2 (actually available for testing), I will have to change once again of compiler…

2.2 The Java Virtual Machine

To ensure the universal portability between on a huge diversity of platforms, the programmer must know at least how to access the different components of the machine, and where he/she can expect an input to come from and where he/she can send outputs. Therefore, the Java standard defines a **Java Virtual Machine** (**JVT**) that is the minimum that must have all platforms on which exists. The JVT includes only two things: an input and an output.

But, it is important to know that these two things are not the standard input (a keyboard) and output (a printer) used by most of the programming languages and by the *Network Virtual Terminal* (see below).

In the JVT the input are provided by a pointing device such as a mouse and the output by a graphical display area. This choice has been made by sun because these input/output are more likely to be on any kind of system rather than a keyboard and a printer. For example, a touch-screen does not have a keyboard and a Java-applet is not able to write on a printer.

The use of some instruction like `System.out.println()` (a basic instruction that sends a string on the output) for example, is not “100% pure Java”, because on some automated systems this instruction will fail due to the lack of device where to display the string without formatting it.

2.3 *Some adaptations I did to the JDK*

After my first month of Java-Programming, I found this language was great, except one thing: the lack of pre-processor. Till this year I had never used languages with pre-processor, but when I learned C in the Fall session, I really enjoyed the features of the pre-processor and later when I wrote the web server of the McGill Daily Français, I made an intensive use of the C pre-processor.

The lack of pre-processor was a problem for me as I expected to translate this software at least in German and French and I know that a translation is really easier when all the texts to translate are gathered at the same place. The translation may also require some changes in the size of some dialog boxes (a text with the same meaning do not have the same length in all languages, you can check that just by having a look at the bilingual Canadian constitution) so these size must also be all determined at the same place. Thus I choose to re-use the pre-processor available with `gcc` (gnu C compiler) and made a little script (`jav.bat`) that automatically pre-processes my files and compile them:

```
@echo off
cpp -P %1.java >
c:\disk_f\programmation\java\compilation\%1.java
javac -g c:\disk_f\programmation\java\compilation\%
1.java
copy c:\disk_f\programmation\java\compilation\%1.class
```

(Where `c:\disk_f\programmation\java\compilation\` is a directory where the pre-processed files are stored; `gcc` and `javac` must both be in the path)

3 A brief overview of the FTP standard

Opposed to the brand new Java language, the Internet protocols are more than 10 years old. This means that they are based on the technology that was available and commonly spread at that time…

3.1 The Telnet standard and the Network Virtual Terminal

Nearly all Internet protocols rely on the Telnet Protocol Specification, as described by J.Postel and J.Reynolds in the RFC 854.

The most important thing that the Telnet protocol does is the specification of the minimum requirements of any machine that implements these protocols. These requirements define what is called the **Network Virtual Terminal** (**NVT**). An NVT is the minimum unit required communicating over the Internet. It has:

-A keyboard or any such input device able to enter characters

-A printer or any such output device able to display/print characters.

The definition given in the FRC 854 is:

```
    "An NVT is an imaginary device which provides a
standard, network-wide, intermediate representation of
a canonical terminal.
    […]
    The NVT has a printer and a keyboard.  The printer
responds to incoming data and the keyboard produces
outgoing data which is sent over the TELNET connection"
```

It is worth to note that for example any UNIX process may be considered as an NVT, so communications using the Internet protocols can be done between humans, but also between a human and a process or between two processes.

The telnet specification also defines how Telnet commands must be embedded within the data (here data may refer to both real data or simply to command transmitted by a "higher level" protocol such as the FTP protocol).

3.2 The FTP standard

The FTP standard defines the set of command that must be used between two NVT to initiate transfers of data such as files. It is worth to know (I spent a long time before finding that), that the data are not transferred using the communication channel established between the two NVT, but for each data transfer, a data transmission channel is opened, used to transmit raw and then closed to signify the End Of File. This channel is called the **Data Port** (**DTP**), whereas the normal NVT communication channel is the **Command Port**. It is also important to know that a new data port must be opened prior to each data transmission this is done with the command `PORT`.

In addition to the standard NVT requirement, any FTP host must have access to a file system. The standard has been built assuming that the file system has a tree structure as does UNIX, Multics or MS-DOS.

Thus, the commands are split between different groups:

- Access control commands that let the user login, enter his/her password…
- Transfer Parameter Commands, these commands let the user specify the DTP, the transmission mode (binary, ASCII or EBCDIC).
- Service commands that are used to manage the connection (Abort, Reset…) or to initiate file transfer.

To these commands corresponds a set of specified answer. These answers are made of a tree digit number followed by a human-readable message. The first digit specifies the completion of the command:

- **1yz** Positive preliminary reply: the command sent by the user was successfully understood and an action started. Often this answer means that the user has to check his DTP to receive data.
- **2yz** Positive Completion reply: The request as been successfully completed.
- **3yz** Positive Intermediate reply: the request has been understood, but the server needs more information (a password for example) before processing the command.
- **4yz** Transient Negative Completion Reply: The command could not be processed at the moment, but the user is encouraged to try again… For example, if the server shuts down after 900 seconds it

sends a 421 code that means that the next command will not be understood, but if the user logs back in and tries again, the command may be understood.

- **5yz** Permanent Negative Completion Reply: The command was not accepted (eg: command not understood, file not found…)

3.3 Comparison between the NVT and the JVM

After the description of both the Java Virtual Machine and the Network Virtual Terminal it appears that the have different nature:

One is based on a character-oriented communication whereas the other is based on an event-oriented communication.

The conversion between these two different philosophy has sometimes been hard especially when the FTP answer where specified to be aimed "for human user only" and thus not for automated processing whereas their display on the JVM required a processing (this worst problem appeared with the `LIST` command that was the only way of having accurate information about the files, but which is not for automated processing).

The same problem appears with the Network Computers (NC) that are supposed to be computers of the future: as they are installed on the network, they require a character oriented communication, but end user more and more require nice graphical display, these NC have to internally run event oriented communication. Even if some problems may appear, I think that in this context, Java provides a good set of tools for these "translations" and it is a good language for such computers.

4 The software

After the description of the main tools on which my software relies, I will now describe it.

As the aim of an FTP software is to transfer files between computers it must provide a way to connect onto these computers. As I think that sometimes one may need to transfer files from one computer to several others, I have not limited the number of computers to which one can be connected at the same time.

Each computer will be handled the same without consideration of this computer being the local computer or a remote computer. For each of these computer a “connection window” is opened and each computer is handled independently.

This feature may be very useful for somebody who wants to transfer files between two remote systems: even if the local internet connection is very bad or slow (DAS connection for example), the user will be able to transfer data between the remote systems at the highest speed possible between these two host (with the FTP software I have tested under win 95 you always have to download the file on your hard disk and then upload it on the target host; if you have a DAS connection and want to transfer files of the scale of the megabyte, this may take a while…).

To let the user connect to a host, a list of recorded host is displayed in each “connection window” of the software’s main screen. This list of ‘predefined host’ will be read from the `.netrc` file, as the basic UNIX FTP software usually does it. To preserve compatibility and universality, the password will not be encrypted in this file (exactly as in the usual `.netrc` file) this is also required by the fact that as French citizen I am not allowed to write cryptographic procedures and them bring them back home!

Here is a sample of `.netrc` file (more information can be found in the UNIX man pages):

```
machine willy.cs.mcgill.ca
name Willy
login nicolas
password xxxxx

machine lisa.cs.mcgill.ca
name Lisa
login nicolas
password xxxxx

machine criens.u-psud.fr
login p99deler
password xxxxx

machine korsika.desy.de
login delerue
password xxxxx

machine abraham.ugrad.physics.mcgill.ca
```

```
login nicolas
password xxxxx
```

Two entries are added to this list. The first one is “connect to a local host” and the second one is “connect to another host”.

As I have chosen to deal with the local host exactly as I does with the remote ones, there will be no specified window for the local host and thus, it was necessary to allow the user define where he/she wants to connect to the local host.

As the user may want to connect to hosts he never connected to before, I added the functionality “connect to another host”. When double-clicked this option displays a dialog box where the user has to enter his/her login and password and the remote host name. After, the user may click on three different buttons. The first one will initiate the connection and add this host name to `.netrc` The second one will simply connect to the remote host and the third one will simply add the host’s name to the `.netrc` (but the “add to the `.netrc`” feature is not implemented at the moment).

The background of the non-connected “connection window” is green whereas it is blue for connected windows.

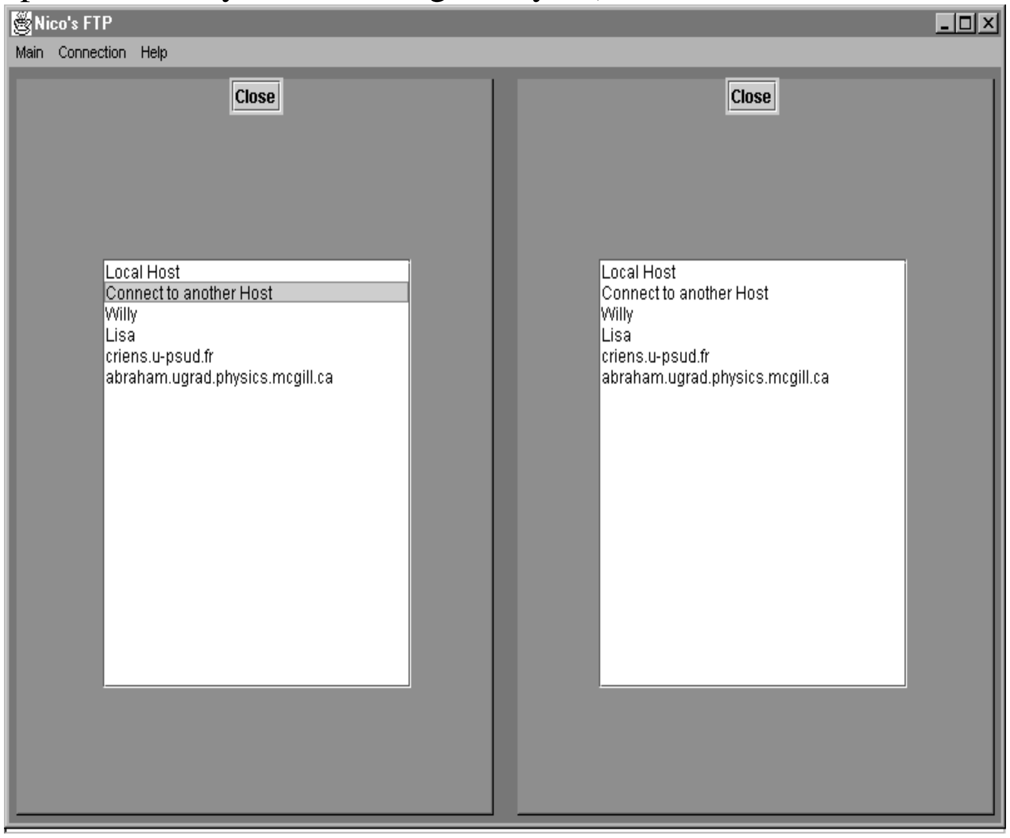
For a non-connected window, the tool bar only offer one possibility: closing the current “connection window”, this may be useful if one just want to browse the files on a remote host or if after connecting to many different hosts one wants to reduce the number of windows.

The application menu also offers the possibility to connect to a remote host. When a host is selected in this menu, the first non-connected window is connected to the specified host. If all windows are connected, a new one is created.

The menu also offers two other topics: “main” and “help”. Under main, all the customisation features will be implemented later. At the moment, only the “quit” option is enabled.

The help menu is also under construction, especially because I am waiting for the release of a stable version of JDK 1.2 (at the moment only release are available with pre-specification, so I preferred to wait).

In the help menu, one can also access the “credits” and the registration explanations (I will probably release this software as a shareware for the windows version when I will have added a few more features and found a .exe generator; the retail price will probably be something like CA$10.00 or 6.00 Euro (the future European Currency Unit, starting next year).



Local Host
Connect to another Host
Willy
Lisa
criens.u-psud.fr
abraham.ugrad.physics.mcgill.ca

Local Host
Connect to another Host
Willy
Lisa
criens.u-psud.fr
abraham.ugrad.physics.mcgill.ca

A screen copy of the main screen of the application, before any connection.

Once a connections has been established, the background colour if the connection window becomes blue and a tree containing all the files of the login directory is displayed.

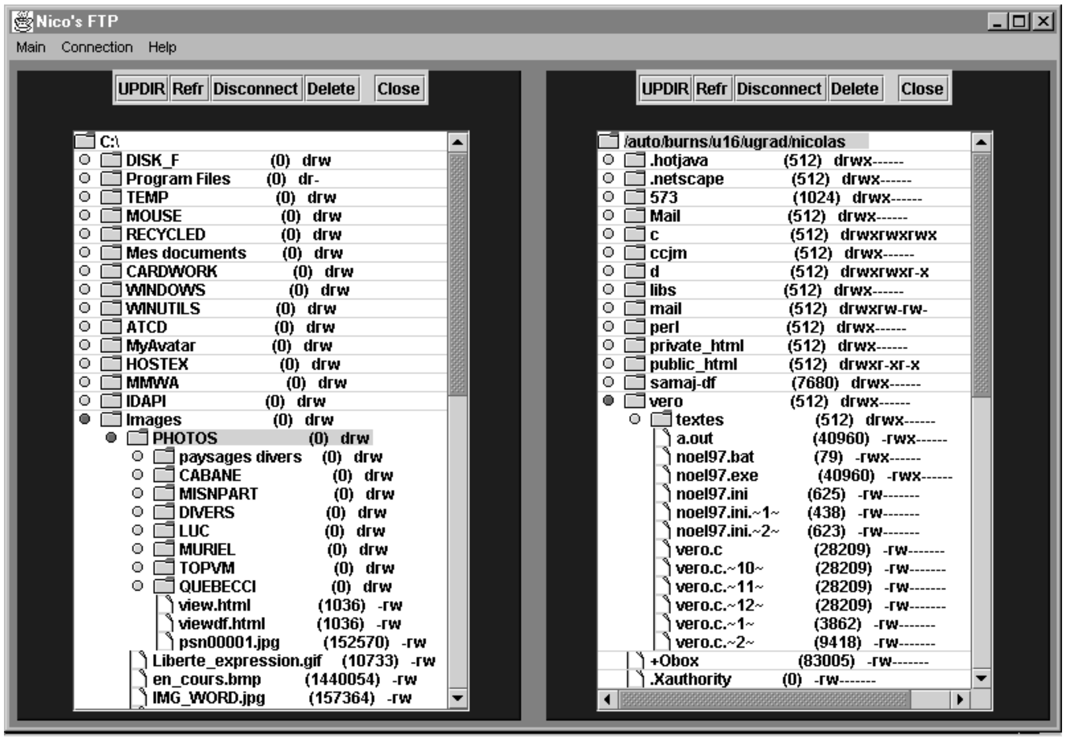
To display this, the list of files returned by the `NLST` FTP command is sorted alphabetically then the `LIST` command is sent and its answer is analysed. As there are no specification for the `LIST` command answer's format (except that this command is not aimed at an automated processing), the software has to make guesses to identify the size of the file and the access right (in the case of a UNIX like structure). If the directory is empty, then a file named ".." is shown the let the user know that the directory has been expanded successfully but is empty.

Once these elements have been found, a new list is returned with the directories first and then the files.

Files can be selected and then dragged to another connection window. If this window is connected, the two host will communicate together to initiate the transfer of all selected files. At the moment, the transfer of directories with all their files is not implemented, but this will be done soon as it seems to be quiet easy to do.

By double-clicking on the a directory folder, one can see the content of the directory (at the moment when one doubles click on a directory for the first time, the tree is rebuilt with all its branches collapsed and one needs to double-click a second time to see the content of this directory).

After the connection, a tool appears at the top of the connection window. This toolbar contains the following items: “updir” that let the user browse the upper directory, “Refr” that let the user refresh the current list of display (but with the same problem than for the directory expanding: the tree is rebuilt with all its branches collapsed…), “Disconnect”, to let the user terminate a connection (after clicking on this button, the connection window returns to its not-connected state), “Delete” that delete all selected files and empty directories and “Close” that terminates the connection (sends the “QUIT” FTP command to the remote host) and close the connection window.



```
C:\
  DISK_F                 (0)  drw
  Program Files          (0)  dr-
  TEMP                   (0)  drw
  MOUSE                  (0)  drw
  RECYCLED               (0)  drw
  Mes documents          (0)  drw
  CARDWORK               (0)  drw
  WINDOWS                (0)  drw
  WINUTILS               (0)  drw
  ATCD                   (0)  drw
  MyAvatar               (0)  drw
  HOSTEX                 (0)  drw
  MMWA                   (0)  drw
  IDAPI                  (0)  drw
  Images                 (0)  drw
    PHOTOS               (0)  drw
      paysages divers    (0)  drw
      CABANE             (0)  drw
      MISNPART           (0)  drw
      DIVERS             (0)  drw
      LUC                (0)  drw
      MURIEL             (0)  drw
      TOPVM              (0)  drw
      QUEBECCI           (0)  drw
      view.html          (1036)  -rw
      viewdf.html        (1036)  -rw
      psn00001.jpg       (152570)  -rw
    Liberte_expression.gif  (10733)  -rw
    en_cours.bmp         (1440054)  -rw
    IMG_WORD.jpg         (157364)  -rw
```

```
/auto/burns/u16/ugrad/nicolas
  .hotjava               (512)  drwx------
  .netscape              (512)  drwx------
  573                    (1024)  drwx------
  Mail                   (512)  drwx------
  c                      (512)  drwxrwxrwx
  ccjm                   (512)  drwx------
  d                      (512)  drwxrwxr-x
  libs                   (512)  drwx------
  mail                   (512)  drwxrw-rw-
  perl                   (512)  drwx------
  private_html           (512)  drwx------
  public_html            (512)  drwxr-xr-x
  samaj-df               (7680)  drwx------
  vero                   (512)  drwx------
    textes               (512)  drwx------
      a.out              (40960)  -rwx------
      noel97.bat         (79)  -rwx------
      noel97.exe         (40960)  -rwx------
      noel97.ini         (625)  -rw-------
      noel97.ini.~1~     (438)  -rw-------
      noel97.ini.~2~     (623)  -rw-------
      vero.c             (28209)  -rw-------
      vero.c.~10~        (28209)  -rw-------
      vero.c.~11~        (28209)  -rw-------
      vero.c.~12~        (28209)  -rw-------
      vero.c.~1~         (3862)  -rw-------
      vero.c.~2~         (9418)  -rw-------
  +Obox                  (83005)  -rw-------
  .Xauthority            (0)  -rw-------
```

A screen copy of the application with two "connection windows" connected to the local host (left) and to a remote host (right)
On this picture, one can see the file trees with some directories expanded (images and images\photos on the local host and vero for the remote host)

Another important feature that I gave to my code is that by the use of an included file with all text and text related constant, it will be very easy to translate. As the High-Energy Physics community is made of people from various country, it is nicer to be able to offer them (and also to general public) software in their own language. Easy translation is also (I think) a way to save cost when trying to make the software comply with national/provincial laws such as bill 101 in province of Québec.

As many FTP server automatically disconnect an host that has not sent any command during 900 seconds, the application automatically logs back in when such disconnection message is sent. (At the moment this functionality still experience some problems)

5 This is still a work in progress

I believe that a software as a piece of fine art, is never completely finished, more features can always be added or parts of the algorithm be improved.

Here is a first release of the software that does what I wanted: transfer files between computers. However, I am not satisfied, and I will improve this application when I will find time to do so (and prior to releasing as shareware).

The things I would like to improve are:

- The addition of the host name to the `.netrc` file when requested
- The possibility of specifying a specific login directory for each host recorded in the `.netrc`
- The possibility of bundling hosts together, for example, if you often transfer files from a server A to a server B, it would be great if by a single click you could login to both A and B

- Automatically expand the file tree when its content is updated
- Make all the files of a directory to be transferred/deleted when the directory's folder is transferred/deleted
- Fixing the resizing problem when a connection window is closed or opened
- Fixing the auto re-login feature
- Resize some dialog boxes and changes some colours
- Add progress bar on all FTP commands (at the moment the user can not know what is going on), and also add dialog boxes show error messages that may be sent by the remote host.
- Checking the pieces of code that still use the JDK 1.0 API and rewrite them using the JDK 1.1 or 1.2 standard.

In addition, there are some features that I think would be nice to add:

- Possibility of creating directories and to rename files
- Display of a diagram showing how much space is used by each directory and its content (including subdirectories), this feature will be a graphical display of what the UNIX command "`du`" does (this will be very useful to find where all the space on my hard disk has gone!).
- Automatic download of an HTML file and all files referred by this file on the same server (other HTML files and pictures referred by tags such as <A HREF=…> or <IMG SRC=…>).

I am also open to any suggestions that could be made by any FTP software user… (I have a permanent e-mail where I can receive such suggestions at nico@backpackers.com).

6 Portability

Thanks to the "Write once, run everywhere" rule, as the FTP software is running on my computer, it should be running on any computer of any kind without any problems! To have an idea of the computers for which it is possible to find a Java Virtual Machine interface, one can have a look on Java's web site at the portability list:

http://java.sun.com/cgi-bin/java-ports.cgi

If I trust what Java as shown at the JavaOne conference (and there are no reasons not to trust it), my application will also soon be able to run onto devices such as a TV, a microwave or even a cellular phone… May be, thanks to my software, one will some day be able to download cooking recipes from his Java-enhanced microwave…

But, even if the future looks bright, when I tried to generate a small embedded stand alone `.exe` file I discovered that the execution of any Java software made with the JDK requires the installation, by the end-user, of either the JDK (more than 10Mo compressed) or the JRE (only 3 Mo!). And as I use JFC, one will need to install the swing kit that uses 7.5 additional Mo, or at least the classes included with it (3Mo).

So, I tried to use some commercial software that claim to be able to generate `.exe` file from Java programs. After trying a few trial versions of these softwares, I discovered that none of them is compatible with the JDK 1.1.5 and the JFC.

So, yes, my software is very portable, but one needs to bring more than 10Mo of file with a software that does not weight more than a few hundreds by itself… (even on skinner, the SOCS's computer on which the JDK is installed, my software can not run due to the fact that the JFC are missing…)

According to the information released by sun in the Java newsgroups, they should release soon a compiler that will be able to generate `.exe` files. I am looking forward for this in order to be able to release my software as stand alone application.

The speed of the application is also one of my big concerns: at the moment, the application runs quiet slowly. For most of the task it is not a problem, because a FTP transfer is always slow due to the slowness of Internet, but, when one has to wait several seconds before a window appears or the display is refreshed, it is may be too much… Java claims that "Just In Time" compiler will be able to boost the speed of the applications. I hope so…

7 Conclusion

When I started this project, I wanted to learn Java as a tool to make fun applets on my web pages and a tool to build graphical application on top of the user-not-so-friendly applications that we usually use in High Energy Physics, a kind on the icing on the top of the cake, because I thought that particle physicist will always have to write there software in Fortran and the data recording interfaces in assembler languages.

Four month later I still believe that Java is a great graphical language, but I also discovered that this language is adapted to automated systems programming and small data acquisition devices. So, I believe that Java may be also a programming language that I may need as particle physicist when I will need to program such devices.

During this project, I also learned lots of stuff about the Internet and the Internet protocols. As Internet is an ever-growing thing that will probably play an increasing role in all parts of our life, I probably acquired a knowledge that will be very worth in a few years…

8 Appendix

8.1 Some useful URLs and bibliography

It would be unbelievable to do Java Programming without having a glance at the web site where Java is born: http://java.sun.com

Sun and JavaSoft provides also a place where Java Developers can meet. It is called the “Java Developers Connection” and it is accessible at http://developer.javasoft.com the access of the site is restricted to members, but membership is free and without any obligations…

The Java Language Specifications are available at
http://java.sun.com/docs/books/jls/html/index.html

To learn Java it can be useful to have a look at the Java Tutorial, made by sun
http://java.sun.com/docs/books/tutorial/getStarted/index.html

Reference pages about the different classes provided with the JDK are provided at:
http://java.sun.com/products/jdk/1.1/docs/api/a-names.html

And they can be downloaded with the JDK (more than 10 Megs!) at
http://java.sun.com/docs/books/tutorial/information/download.html
and many other useful documents related to the Java language and its specification can be found at http://java.sun.com/docs/index.html

More information about the swing components can be found at http://java.sun.com/docs/books/tutorial/ui/swing/

The rules to be followed to obtain the 100% Java Certification are gathered in the 100% Pure Java Cookbook at
http://www.suntest.com/100percent/cpd/doc/cbook/cookbook.html

Many interesting applications or devices compatible with the Java technology have been shown at the JavaOne conference in April 1998 and they are gathered in the JavaTechTown:
http://java.sun.com/features/1998/04/javatechtown.html

While developing this software I also used a book (in French): “Java, le livre d’Or” by P.Longuet, published in France by Sybex. This book has been very useful for me to learn Java and also as reference book while internet was shut down during the Ice Storm, but even if it has been published in 1996, this book only covers the JDK 1.1 release and thus is already out-of-date…

The RFC 959 “FTP specification” and RFC 854 “Telnet Protocol Specification”, both by J.Postel *et al.* have also been very useful reference documents. Both can be found at:
http://www.cis.ohio-state.edu/hypertext/information/rfc.html

8.2 The source code

Here is a sample of the source code as of end of April 1998, I know that some of the tricks I need are not the best I could have done, I need to add comments to some parts or to rewrite some parts, but the following code is working…

Only the most important classes and the constants file are provided. All other classes can be found on the Disk included with this report (the application uses a total of 15 classes plus a file of constants).

```
//#include "c:\disk_f\programmation\Java\FTP\consts.java"
#include "c:\disk_f\programmation\Java\FTP\textes.java"

import java.io.*;
import java.util.*;
import java.awt.*;
import connections.*;
import host_list.*;
import dialg.*;
import com.sun.java.swing.*;
import com.sun.java.swing.tree.TreePath;

  /**
   * the NicoFTP class in the class that creates the graphical
interface of the application
   */
public class NicoFTP extends Frame
{

  Dimension winSize; /** the size of the window */

  Vector vectConnections = new Vector(); /** All connections are
gathered in a vector */

  connections currentConnection; /** Here we store the current
connection */
  //  connections target,source; /** the connections that
exchanged data in a dragg operation */

  host_list hostRecords;

  DISPLAY_TYPE display;
  GridBagConstraints dispConstraints;

  boolean dragged; /** remembers if an object has been dragged
recently */

  //defining the constructor
  public NicoFTP()
  {
    int i;

    //defining the display
    display=new DISPLAY_TYPE();
    //display=new DISPLAY_TYPE(4,4);
    //    display=new DISPLAY_TYPE(1,4*INITIAL_CONNECTIONS);
        dispConstraints = new GridBagConstraints();
    setLayout(display);
    dispConstraints.fill=GridBagConstraints.NONE; /* The empty
space is left empty */
    dispConstraints.anchor=GridBagConstraints.CENTER; /* The
components are centered */

    setTitle(TITRE);
    winSize=new Dimension(INITIAL_WIDTH,INITIAL_HEIGHT);
    setSize(winSize);
    setResizable(true);

    hostRecords = new host_list();

    //definig the Menu
    MenuBar mb = new MenuBar();

    Menu m1= new Menu (MENU_MAIN);
    mb.add(m1);
    {
      Menu i11 = new Menu(MENU_MAIN_PARAMS);
      m1.add(i11);
      {
        MenuItem i111 = new MenuItem(MENU_MAIN_PARAMS1);
        i11.add(i111);
        MenuItem i112 = new MenuItem(MENU_MAIN_PARAMS2);
        i11.add(i112);
      }
      MenuItem i12 = new MenuItem(MENU_MAIN_QUIT);
      m1.add(i12);
    }

    Menu m2= new Menu (MENU_CONNECT);
    mb.add(m2);
    {
      Vector list; /* we will get the list of the host */
      int hostIndice; /* and we will display them, one by one, so
we will need a counter */

      MenuItem i21 = new MenuItem(MENU_CONNECT_NEW);
      m2.add(i21);
      MenuItem i22 = new MenuItem("-");
      m2.add(i22);
      /*      MenuItem i22l1 = new MenuItem(MENU_CONNECT_LOCAL);
      m2.add(i22l1);
      */
      hostIndice=0;
      list = hostRecords.hostList();
      while(hostIndice<list.size())
        {
          MenuItem hostItem = new MenuItem((list.elementAt
(hostIndice)).toString());
          m2.add(hostItem);
          hostIndice++;
        }
    }
    Menu m3 = new Menu(MENU_HELP);
    mb.add(m3);
    {
      MenuItem i31 = new MenuItem(MENU_HELP_FTP);
      m3.add(i31);
      MenuItem i32 = new MenuItem(MENU_HELP_CREDITS);
      m3.add(i32);
      MenuItem i33 = new MenuItem(MENU_HELP_BUGS);
      m3.add(i33);
      MenuItem i34 = new MenuItem(MENU_HELP_REG);
      m3.add(i34);
    }
    // display of the menu
    setMenuBar(mb);

    //Defining the sub-frames (connections)
    for (i=0; i< INITIAL_CONNECTIONS; i++)
      {
        currentConnection = new connections(winSize.width/
(INITIAL_CONNECTIONS),winSize.height-MENU_HEIGHT,i*(winSize.width/
(INITIAL_CONNECTIONS)),MENU_HEIGHT,hostRecords,display,this);
        vectConnections.addElement(currentConnection);
      }

    show(); /* and finally we display the window */

  } // end of the constructor

  //events handler

  //events related to the menu
  public boolean action(Event evt, Object arg)
  {
    Vector list; /* we will get the list of the host */
    int hostIndice=0; /* and check them one by one */

    if (evt.target instanceof MenuItem)
      {
        String Cde = (String)arg;
        //Main menu
        if (Cde.equals(MENU_MAIN_QUIT))
          System.exit(0);
        if (Cde.equals(MENU_MAIN_PARAMS1))
          NotAvailable();
        if (Cde.equals(MENU_MAIN_PARAMS2))
          NotAvailable();

        //Connections
        if (Cde.equals(MENU_CONNECT_NEW))
          {
            addConnection();
          }
        list = hostRecords.hostList();
        while(hostIndice<list.size())
          {

            if(Cde.equals((list.elementAt(hostIndice)).toString
()))
            {
              int i=0;
              AFF(("connection "+((list.elementAt(hostIndice)).
toString())));
```

```
              while( i < vectConnections.size())
                { /* we look for the first non connected
connection */
                  currentConnection=(connections)
vectConnections.elementAt(i);
                  if(currentConnection.connected)
                    {
                      i++;
                      currentConnection=null;
                    }
                  else
                    {
                      i=vectConnections.size();
                      AFF(("trouve"));
                    }
                } /* while */
              if(currentConnection==null)
                {
                  currentConnection=addConnection();
                }
              if(currentConnection!=null)
                {
                  AFF(("hostIndice "+hostIndice));
                  currentConnection.openConnection(hostIndice);
                }
            }
          hostIndice++;
        }

        //Help
        if (Cde.equals(MENU_HELP_FTP))
          NotAvailable();
        if (Cde.equals(MENU_HELP_CREDITS))
          credits();
        if (Cde.equals(MENU_HELP_BUGS))
          bugs();
        if (Cde.equals(MENU_HELP_REG))
          registration();
      }
    return(true);
  }

  //credits
  public void credits()
  {
    infoBox credits = new infoBox
(this,CREDIT_WIDTH,CREDIT_HEIGHT,CREDIT_TITLE,CREDIT_TEXT);
  }

  //bugs
  public void bugs()
  {
    infoBox bugs = new infoBox
(this,BUGS_WIDTH,BUGS_HEIGHT,BUGS_TITLE,BUGS_TEXT);
  }

  //registration
  public void registration()
  {
    infoBox reg = new infoBox
(this,REG_WIDTH,REG_HEIGHT,REG_TITLE,REG_TEXT);
  }

  //NotAvailable
  public void NotAvailable()
  {
    infoBox na = new infoBox
(this,NA_WIDTH,NA_HEIGHT,NA_TITLE,NA_TEXT);
  }




  // Events in case of Hard close of the window
  public boolean handleEvent(Event evt)
  {
    if ((evt.id == Event.WINDOW_DESTROY)||((evt.id ==
Event.ALT_MASK)&&(evt.id == Event.F4)))
      {
        destroy();
        return(true);
      }
    else
      {
        // AFF(("event: "+evt.id));
      }
    return super.handleEvent(evt);
  }


  /**
   * update is a method that check for each connection if the
current display is up-to-date
   */
  protected void upDate()
  {
    Component c;
    GridBagConstraints dispConstraints = new GridBagConstraints();
    setLayout(display);
    JToolBar tb;

    removeAll(); /* removes all the components of this window */

    for (int i=0; i < vectConnections.size(); i++)
      {
        currentConnection=(connections) vectConnections.elementAt
(i);

        dispConstraints = new GridBagConstraints();
        dispConstraints.fill=GridBagConstraints.NONE;
        dispConstraints.gridx=i;
        dispConstraints.gridy=0;
        dispConstraints.gridheight=1;
        dispConstraints.gridwidth=1;
        dispConstraints.weightx=1.0;
        dispConstraints.weighty=1.0;
        dispConstraints.insets=new Insets(0,0,0,0);
        tb=currentConnection.getToolBar();
        if(tb!=null)display.setConstraints(tb,dispConstraints);
        if(tb!=null)add(tb);

        dispConstraints.fill=GridBagConstraints.NONE;
        c=currentConnection.compo();
        dispConstraints.gridx=i;
        dispConstraints.gridy=1;
        dispConstraints.gridheight=1;
        dispConstraints.gridwidth=1;
        dispConstraints.weightx=1.0;
        dispConstraints.weighty=1.0;
        dispConstraints.insets=currentConnection.insets;
        display.setConstraints(c,dispConstraints);
        add(c);
      }
  } /* upDate */

  /**
   * addConnection is a method that adds a connection in the
connections' list
   */
  public connections addConnection()
  {
    int i=vectConnections.size();
    currentConnection = new connections(winSize.width/(i),
winSize.height-MENU_HEIGHT,i*(winSize.width/(i)),
MENU_HEIGHT,hostRecords,display,this);
    vectConnections.addElement(currentConnection);

    repaint();
    return currentConnection;
  }

  // The graphical part of the window
  public void paint(Graphics g)
  {
    rePaint(g);
  } /* paint */

 /**
  * rePaint repaints the current graphical Display
  */
  public void rePaint(Graphics g)
  {
    winSize=getSize();
    setBackground(Color.gray);

    for (int i=0; i < vectConnections.size(); i++)
      {
        currentConnection=(connections) vectConnections.elementAt
(i);
        currentConnection.setSize(winSize.width/
(vectConnections.size()),winSize.height-MENU_HEIGHT,i*
(winSize.width/(vectConnections.size())),MENU_HEIGHT);
        currentConnection.draw(g);
      }
    upDate();
  } /* rePaint */
  //Handling the dragging of the Mouse
  /**
   * mouseEntry detects when a mouse enters an area
   * and detects if this corresponds to a dragging event...
   */
  public void mouseEntry(connections target)
  {
    connections source;

    if((dragged)&&(target.isMouseDroppedIn()))
      {
        int i=0;
        source=null;
        AFF(("dragging to "+target.theHost.machine));
        while((i < vectConnections.size())&&(source==null))  /* A
loop is done to find which connection has been dragged */
          {
            currentConnection=(connections)
vectConnections.elementAt(i);
            if((currentConnection.draggedOut)&&
(currentConnection.connected))
              { /* if something has been dragged out of this
connection and this connection is connected */
                    AFF(("dragging from
"+currentConnection.theHost.machine));
                    source=currentConnection;
                    source.draggedOut=false;
                    fileXfer(source,target);
              } /* if draggedOut */
            i++;
          } /* while i<size && source==null */
      } /* if dragged */
        dragged=false;
  } /* mouseEntry */

  /**
   * fileXfer copies all the file selected in source to target.
   * (in fact, just the kind of transfer needed is selected)
   */
  public void fileXfer(connections source,connections target)
  {
    if(source.equals(target)==false)
      { /* no transfer in the same tree... */

        /* first some operations are done to prepare the transfer
*/
        target.goXferDir(); /* the target is asked to go in its
transfer directory */
        source.stackSelection(); /* the list of the file to be
transfered is put in a stack */

        /* a loop is done to deal with each copy separately */
        while((source.stack).size()>0)
          {
            String fileName; /* the name of the files to be copied
*/
            String sourcePath;
            sourcePath=source.makePath((TreePath)
source.stack.elementAt(0),true);
            fileName=source.getFileName((TreePath)
source.stack.elementAt(0));

            AFF(("sourcePath "+sourcePath));

            /* then the mode of transfer is selected */
            if((source.theHost.machine.equals(LOCAL_MACHINE)))
              {
                if((target.theHost.machine.equals(LOCAL_MACHINE)))
                  {
                    doubleLocalXfer
(source,target,sourcePath,fileName);
                  } /* local->local */
                else
                  {
                    localRemoteXfer
(source,target,sourcePath,fileName);
                  } /* local->remote */
              }
            else
              {
                if((target.theHost.machine.equals(LOCAL_MACHINE)))
                  {
                    remoteLocalXfer
(source,target,sourcePath,fileName);
                  } /* remote->local */
                else
                  {
                doubleRemoteXfer
(source,target,sourcePath,fileName);
                  } /* remote->remote */
              }
            source.stack.removeElementAt(0);
          } /* while stack size > 0 */
        target.ftp.setDir(target.ftp.currentTreePath);
      } /* source != target */
  } /* fileXfer */

  /**
   * doubleLocalXfer handles a file transfer between two local
host
   */
  public void doubleLocalXfer(connections source,connections
target,String sourcePath,String fileName)
  {
    RandomAccessFile sourceFile;
    RandomAccessFile targetFile;
    boolean noError;
    byte buffer[] = new byte[BUFFER_SIZE];

    try  /* let's try to open the file containing the source */
      {
        sourceFile = new RandomAccessFile(sourcePath,"r");
      }
    catch  (IOException e)
      {
        sourceFile = null;
```

```
          AFF((e.getMessage()));
        }
      catch  (SecurityException e)
        {
          sourceFile = null;
          AFF((e.getMessage()));
        }

      try  /* let's try to open the file where goes the target */
        {
          targetFile = new RandomAccessFile(new File
(target.ftp.currentPath,fileName),"rw");
        }
      catch  (IOException e)
        {
          targetFile = null;
          AFF((e.getMessage()));
        }
      catch  (SecurityException e)
        {
          targetFile = null;
          AFF((e.getMessage()));
        }

      noError=true;
      while (noError)
        {
          int lengthRead=0;

          try /* let's try to read the file*/
            {
              lengthRead=(sourceFile.read(buffer));
              noError=(lengthRead!=-1); /* the line is read from the
file */
              /* the end of stream is reached if read returns -1 */
            }
          catch  (EOFException e)
            {
              noError=false;
            }
          catch  (IOException e)
            {
              AFF((e.getMessage()));
              noError=false;
            }

          try /* let's try to read the file*/
            {
              targetFile.write(buffer,0,lengthRead);
            }
          catch  (IOException e)
            {
              AFF((e.getMessage()));
              noError=false;
            }
          buffer=new byte[BUFFER_SIZE];
        } /* while noError */
      try
        {
```

```
        sourceFile.close();
      }
    catch (IOException e)
      {
        AFF((e.getMessage()));
      } /* catch close */
    try
      {
        targetFile.close();
      }
    catch (IOException e)
      {
        AFF((e.getMessage()));
      } /* catch close */

  } /* doubleLocalXfer */



  /**
   * localRemoteXfer handles a file transfer from a local to a
remote host
   */
  public void localRemoteXfer(connections source,connections
target,String sourcePath,String fileName)
  { /* we are sure that target is a FTP connection */
    ((module_ftp)target.ftp).sendFile(sourcePath,fileName);
  } /* localRemoteXfer */



  /**
   * remoteLocalXfer handles a file transfer from a remote to a
local  host
   */
  public void remoteLocalXfer(connections source,connections
target,String sourcePath,String fileName)
  { /* we are sure that source is a FTP connection */
    String thisPath=target.ftp.currentPath.concat
(target.ftp.localPathSeparator);
    if(!(thisPath.endsWith(target.ftp.localPathSeparator)))
      {
        thisPath=thisPath.concat(target.ftp.localPathSeparator);
      }
    ((module_ftp)source.ftp).receiveFile
(sourcePath,fileName,thisPath);
  } /* remoteLocalXfer */



  /**
   * doubleRemoteXfer handles a file transfer between two remote
host
   */
  public void doubleRemoteXfer(connections source,connections
target,String sourcePath,String fileName)
  { /* we are sure that both source and target are FTP connections
*/
    String port; /* the port number on which the target listens */
    AFF(("double Remote "));
    port=((module_ftp)target.ftp).pasvRemote();
    ((module_ftp)source.ftp).sendRemote(port,sourcePath);
    ((module_ftp)target.ftp).receiveRemote(fileName);
    ((module_ftp)source.ftp).receive();
    ((module_ftp)target.ftp).receive();

  } /* doubleRemoteXfer */

  /**
   * removeConnection removes a connection from the connection
vector
   */
  public void removeConnection(connections c)
  {
    int i=0;
    while((i>=0)&&(i < vectConnections.size()))
      { /* a loop is done to find which connection is to be
removed */
        currentConnection=(connections) vectConnections.elementAt
(i);
        if(currentConnection.equals(c))
          {
            currentConnection.disconnect(); /* the connection is
closed */
            if (vectConnections.size()>1)
              { /* and then removed if it makes sens (ie there
will be some connections left) */
                vectConnections.removeElementAt(i);
              }
            i=-1; /* i is set to a negative value to terminate the
loop */
          }
        else
          {
            i++;
          }
      }
    repaint();
  } /* removeConnection */


  //The destructor
  public void destroy()
  {
    AFF(("Debut du processus de deconnection"));
    for (int i=0; i < vectConnections.size(); i++) /* a loop is
done to close all connections */
      {
        currentConnection=(connections) vectConnections.elementAt
(i);
        currentConnection.disconnect();
      }
    AFF(("Au revoir"));
    System.exit(0);
  }

  //main
  public static void main(String args[])
  {
    NicoFTP Win = new NicoFTP();
  }
}
```

☆☆☆☆☆

```
#include "c:\disk_f\programmation\Java\FTP\textes.java"

import java.awt.*;
import host_list.*;
import com.sun.java.swing.*;
import com.sun.java.swing.tree.TreePath;
import com.sun.java.swing.tree.*;
import MListener.*;
//import MMotionListener.*;
import module_com.*;
import aHost.*;
import java.util.*;
import sizeableScrollPane.*;
import java.awt.event.*;
import toolBarActionListener.*;

/**
 * The connections class contains all the elements used by each
sub frame of the main window
 */
public class connections
{
  int height;
  int width;
  /**
   * urc_x and urc_y are the coordinates of the upper right corner
   * of the display area of this connection
   */
  int urc_x;
  int urc_y;

  /**
   * this boolean knows if this connexions is currently connected
or not
   */
  boolean connected;

  host_list hostRecords;

  DISPLAY_TYPE displayArea;

  /**
   * The component that corresponds to the visual part of this
connection eg: a ScrollPane
   */
  Component connectVisuCompo;
  /**
   * The component that has the informative part of the
connection: eg a JList
   */
  Component connectListCompo;
  Insets insets; /** the insets corresponding to the current
displayed tool */
  JToolBar toolBar; /** the toolbar of this connection */

  module_com ftp; /** the module that handles all communications
with the file system */

  aHost theHost; /** the host to which this connection is
connected */

  NicoFTP parent;

  boolean dragSeen;/** remembers if an object is currently been
dragged or not */
  //  boolean mouseIn; /** remenbers if the mouse is in this
connection or not */
  boolean draggedOut; /** remembers if something has just been
dragged */

  Point dropPoint; /** remembers the point at which the mouse
entered this connection */
  Vector stack; /** the stack in which all the files to be
transfered are stored */

  // constructor
  public connections(int w, int h, int x, int y,host_list
hR,DISPLAY_TYPE dA,NicoFTP par)
  {
    setSize(w,h,x,y);
    hostRecords=hR;
    displayArea=dA;
    connectVisuCompo=null; /* when we start this connection has no
value */
    connectListCompo=null; /* when we start this connection has no
value */
    connected=false; /* when we start the connection is not
connected */
    ftp=null;
    theHost=null;
    parent=par;
    dragSeen=false;
    //    mouseIn=false;
    stack=null;
    //    connectPanel=null;
    insets=null;
    toolBar=null;
  }

  //Graphical functions
  public void setSize(int w,int h,int x,int y)
  {
    width=w;
    height=h;
    urc_x=x;
    urc_y=y;
  }

  /**
   * The function "draw" draws the frame of each connexion
   */
  public void draw(Graphics g)
  {
    g.setColor(Color.black);
    g.draw3DRect(urc_x+BORDER,urc_y+BORDER,width-(2*BORDER),
height-(2*BORDER),true);
    if(connected)
      {
        g.setColor(Color.blue);
      }
    else
      {
        g.setColor(Color.green);
      }
    g.fill3DRect(urc_x+BORDER,urc_y+BORDER,width-(2*BORDER),
height-(2*BORDER),true);

    if(toolBar!=null)toolBar.repaint();
    if(connectListCompo!=null)connectListCompo.repaint();
    if(connectVisuCompo!=null)connectVisuCompo.repaint();
  }/* draw */


  //Functions that deals with the content of the connection
  /**
   * The component method returns:
   *   a list of the selectable host if the connexion is not
connected or
   *   a tree of the file on the server if it is.
   */
  public void makeCompo()
  {
    if(connected)
      {
        JTree tree = new JTree(ftp.buildTheTree());
        tree.addTreeExpansionListener(new treeExpander(ftp));
        tree.setRootVisible(true);
        tree.addSelectionRow(0);
        tree.expandRow(0); /* ensure that the first object in the
tree is expanded */
        tree.setRowHeight(TAILLE_POLICE);
        if(ftp.currentTreePath==null)
          { /* the current path has not been initialized, we do it
now */
            ftp.currentTreePath=tree.getPathForRow(0); /* we are
presently at the root */
          }

        //Adding the mouse Listeners
        MListener ml= new MListener(this);
        tree.addMouseListener(ml);

        //Create the scroll pane and adds the scroll pane to the
tree
```

```
        Dimension dim = new Dimension(((int)((((4*width)/5)-
(2*BORDER)))),((int)((.9*(height))-(2*BORDER))));
        sizeableScrollPane scrollPane = new sizeableScrollPane
(tree,dim);

        insets=new Insets(((int)((.05*(height))-(2*BORDER))),
((int)((((1*width)/10)-(2*BORDER)))),((int)((.05*(height))-
(2*BORDER))),((int)((((1*width)/10)-(2*BORDER)))));


        connectVisuCompo=scrollPane;
        connectListCompo=tree;
      } else {  /* not connected */
        JList list = new JList(hostRecords.hostList());
        list.setFixedCellHeight(TAILLE_POLICE);
        MListener ml= new MListener(this);
        list.addMouseListener(ml);

        //Create the scroll pane and add the scroll pane to the
list.
        Dimension dim = new Dimension(((int)((((2*width)/3)-
(2*BORDER)))),((int)((3*(height)/5)-(2*BORDER))));


        insets=new Insets(((int)((1*(height)/5)-(2*BORDER))),
((int)((((1*width)/6)-(2*BORDER)))),((int)((1*(height)/5)-
(2*BORDER))),((int)((((1*width)/6)-(2*BORDER)))));
        sizeableScrollPane scrollPane = new sizeableScrollPane
(list,dim);

        connectVisuCompo=scrollPane;
        connectListCompo=list;
      } /* not connected */
    parent.upDate();
    parent.repaint();
    parent.show();
    connectListCompo.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
    parent.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));

  } /* makeCompo */


  /**
   * The makeToolBar mathods sets the connection's toolBar to what
it must be
   */
  public JToolBar makeToolBar()
  {
    JButton jb;
    JToolBar tb;

    tb=new JToolBar();
    tb.setBorderPainted(false);
    tb.setFloatable(false);

    if(connected)
      {
        jb = new JButton(TB_UPDIR);
        jb.setToolTipText(TB_UPDIR_TIP);
        jb.addActionListener (new toolBarActionListener
(this,TB_UPDIR_NUMBER));
        tb.add(jb);

        jb = new JButton(TB_REFRESH);
        jb.setToolTipText(TB_REFRESH_TIP);
        jb.addActionListener (new toolBarActionListener
(this,TB_REFRESH_NUMBER));
        tb.add(jb);

        /*        jb = new JButton(TB_ABORT);
        jb.setToolTipText(TB_ABORT_TIP);
        jb.addActionListener (new toolBarActionListener
(this,TB_ABORT_NUMBER));
        tb.add(jb);
        */

        jb = new JButton(TB_DISCONNECT);
        jb.setToolTipText(TB_DISCONNECT_TIP);
        jb.addActionListener (new toolBarActionListener
(this,TB_DISCONNECT_NUMBER));
        tb.add(jb);

        jb = new JButton(TB_DELETE);
        jb.setToolTipText(TB_DELETE_TIP);
        jb.addActionListener (new toolBarActionListener
(this,TB_DELETE_NUMBER));
        tb.add(jb);

        tb.addSeparator();

      } /* if connected */
    else
      {
        /* no specific button if not connected */

      } /* else not connected */

    jb = new JButton(TB_CLOSE);
    jb.setToolTipText(TB_CLOSE_TIP);
    jb.addActionListener (new toolBarActionListener
(this,TB_CLOSE_NUMBER));
    tb.add(jb);

    toolBar=tb;

    return tb;

  } /* makeToolBar */

  /**
   * compo returns the visual Component of this connection;
   * if needed, creates it!
   */
  public Component compo()
  {
    if (connectVisuCompo==null)
      {
        makeCompo();
      }
    connectListCompo.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
    parent.setCursor(new Cursor(Cursor.DEFAULT_CURSOR));
    return connectVisuCompo;
  } /* compo */

  /**
   * getToolBar returns the toolBar of this connection;
   * if needed, creates it!
   */
  public JToolBar getToolBar()
  {
    if (toolBar==null)
      {
        toolBar=makeToolBar();
      }
    return toolBar;
  } /* getToolBar */


  //The toolBar events
  /**
   * toolBarAction execute the action corresponding to the number
passed as parameter
   */
  public void toolBarAction(int actionNumber)
  {
    switch(actionNumber)
      {
      case TB_UPDIR_NUMBER:;
        waitCursor();
        ftp.currentTreePath=new TreePath((((JTree)
connectListCompo).getPathForRow(0)).getPathComponent(0));
        ftp.cdup();
        ftp.fileStruct=null;
        makeCompo();
        break;
      case TB_DISCONNECT_NUMBER:;
        waitCursor();
        disconnect();
        break;
      case TB_REFRESH_NUMBER:; /* Refreshes the root */
        waitCursor();
        ftp.setDir(((JTree)connectListCompo).getPathForRow(0));
        break;
      case TB_ABORT_NUMBER:;
        AFF(("ab"));
        break;
      case TB_DELETE_NUMBER:;
        waitCursor();
        AFF(("delete"));
        stackSelection();
        deleteSelection();
        ftp.setDir(((JTree)connectListCompo).getPathForRow(0));
        break;

      case TB_CLOSE_NUMBER:;
        parent.removeConnection(this);
        break;
      default:;
        AFF(("defaut"));
        break;
      }
  } /* toolBarAction */


  //The mouse events
  /**
   * click handle a double-click on an item
   */
  public void click(Point p)
  {
    if(!(connected))
      {
        JList local=(JList) connectListCompo;
        openConnection(local.locationToIndex(p));
      }
  } /* click */

  /**
   * drag handles the dragging of an object
   */
  //  public void drag(Point p)
  public void drag()
  {
            /*AFF(("dragseen "+dragSeen)); */
    if((connected))
      {
        if(!dragSeen) /* avoids multiple handling of this event */
          {
            connectListCompo.setCursor(new Cursor
(Cursor.HAND_CURSOR));
            parent.setCursor(new Cursor(Cursor.HAND_CURSOR));
            dragSeen=true;
          } /* !dragSeen */
      } /* connected */
  } /* drag */

  /**
   * release handles the release of an object
   */
  public void release(Point p)
  {

        if((connected))
          {
            connectListCompo.setCursor(new Cursor
(Cursor.DEFAULT_CURSOR));
            draggedOut=((((JTree)(connectListCompo)).
getPathForLocation(p.x,p.y))==null); /* it has been dragged out if
the mouse has been release out of the scope of this object */
            parent.dragged=draggedOut;
          }
        dragSeen=false;
  } /* release */

  /**
```

```
   * isMouseDroppedIn returns true if the mouse has been dropped
in this component
   */
  public boolean isMouseDroppedIn()
  {
    if(connected)
      {
        return (dropPoint!=null);
      }
    else
      {
        return false;
      }
  } /* isMouseDroppedIn */

  /**
   * openConnection opens a connection either with the local host
or over the network
   */
  public void openConnection(int index)
  {
    if((!(ftp==null)))
      {
        AFF(("Erreur: tentative de double connection"));
        return;
      }
    ftp = hostRecords.getModule(index,this);
    if (ftp!=null)
      {
        waitCursor();
        theHost= hostRecords.getHost(index);
        connected=ftp.openHost(theHost);
      }
  } /* openConnection */

  /**
   * confirmConnection receives a message from the ftp module
confirming
   * that the connection has been established
   */
  public void confirmConnection()
  {
    connected=true;
    makeToolBar();
    makeCompo();
  } /* confirmConnection */

  /**
   * goXferDir makes the current directory to become the directory
   * where the next transfer must occur.
   */
  public void goXferDir()
  {
    TreePath path;

    path=((((JTree)connectListCompo).getClosestPathForLocation
(dropPoint.x,dropPoint.y)));
    AFF(("Path xfer: "+makePath(path,false)));
    ftp.chdir(makePath(path,false));
    dropPoint=null;

  }/* goXferDir */


  /**
   * stackSelection puts all the selected file into a vector
called the "stack"
   */
  public void stackSelection()
  {
    TreePath listPaths[];
    int i;

    listPaths=(((JTree)connectListCompo).getSelectionPaths());
    stack=new Vector(); /* all the former transmisions that have
not been done are cancelled */
    i=0;
    if(listPaths!=null)
      {
        while(i<listPaths.length)
          { /* each element of the selection is added to the stack
*/
            stack.addElement(listPaths[i]);
            AFF(("Path relatif: "+makePath(listPaths[i],true)));
            i++;
          } /* while */
      } /* listPaths */
  }/* stackSelection */

  /**
   * deleteSelection sends the deletion command for all the files
in the stack
   */
  public void deleteSelection()
  {
    while(stack.size()!=0)
      {
        Object element;
        element=stack.elementAt(0);
        //        AFF(("element "+element.toString()));
        stack.removeElementAt(0);
        ftp.delete(makePath((TreePath)element,true));
      } /* while stack.size ! 0 */
  } /* deleteSelection */

  /**
   * makePath gives a string representing the  path to the file
   * if filename==true, this string includes the filename
   * thus one just need to send the RETR command to get the file
   */
  public String makePath(TreePath path,boolean filename)
  {
    if(ftp.localPathSeparator==null)
      {
        /*
         * case where we are still in the login directory
         * then just the last component of the treepath is to be
returned
         * nothing is returned if filename is false
         */
        if(filename)
          {
            return  (((oneFile)((DefaultMutableTreeNode)
path.getLastPathComponent()).getUserObject()).name);
          }
        else
          {
            return "";
          }
      } /* localPathSeparator == null */
    else
      { /* localPathSeparator!=null */
        String toBeReturned;
        int counter;
        int i;

AFF((path.toString()));

        counter=((path.getPathCount())-1);
        if(!filename)counter--;

        toBeReturned="";
        for(i=0;i<counter;i++)
          {
            toBeReturned+=(((oneFile)((DefaultMutableTreeNode)
path.getPathComponent(i)).getUserObject()).name);
            if(!(toBeReturned.endsWith(ftp.localPathSeparator)))
toBeReturned+=ftp.localPathSeparator;
          }
        toBeReturned+=(((oneFile)((DefaultMutableTreeNode)
path.getPathComponent(i)).getUserObject()).name);
        if((!filename)&&(!(toBeReturned.endsWith
(ftp.localPathSeparator))))toBeReturned.concat
(ftp.localPathSeparator);
        return toBeReturned;
      }
  } /* makePath */

  /**
   * getFileName returns the name of the file pointed by this
TreePath
   */
  public String getFileName(TreePath path)
  {
    return (((oneFile)((DefaultMutableTreeNode)
path.getLastPathComponent()).getUserObject()).name);
  } /* getFileName */


  /**
   * disconnect closes the ftp/local connection and reset all
values to their default
   */
  public void disconnect()
  {
    if (connected)
      {
        ftp.closeHost();
        connected=false;
        ftp=null;
        toolBar=null;
        theHost=null;

        makeToolBar();
        makeCompo();
      }
  } /* disconnect */

  /**
   * waitCursor changes the current cursor to the wait cursor
   */
  public void waitCursor()
  {
    if(connectListCompo!=null)connectListCompo.setCursor(new
Cursor(Cursor.WAIT_CURSOR));
    parent.setCursor(new Cursor(Cursor.WAIT_CURSOR));
  } /* waitCursor */

  //The destructor
  public void destroy()
  {
    disconnect();
  }
} /* end of the classe definition */
```

✯ ✯ ✯ ✯ ✯

```
//#include "c:\disk_f\programmation\Java\FTP\consts.java"
#include "c:\disk_f\programmation\Java\FTP\textes.java"

import com.sun.java.swing.tree.DefaultMutableTreeNode;
import com.sun.java.swing.tree.*;
import com.sun.java.swing.*;
import aHost.*;
import java.util.*;
//import oneFile.*;

  /**
   * The module com manages all kinds of connections;
   * its purpose is to be extended either in module FTP or module
Local...
   * therefore, it has been defined to be abstract...
   */
public abstract class module_com extends Object
{
  Vector fileStruct=null; /** contains the file structure of the
current connection */
  String localPathSeparator=null; /** The path separator used by
the system to which this connection is connected */
```

```
  String currentPath=""; /* this string must not finish with a
separator */
  TreePath currentTreePath=null;
  DefaultMutableTreeNode top;
  connections parent=null;


  public abstract void openPort();
  public abstract boolean openHost(aHost theHost);
  public abstract void closeHost();
  public abstract int receive();


  /**
   * getWD returns the name of the current directory on the server
   * and sets currentPath to the right value
   */
  public abstract String getWD();


  /**
   * fileList returns a vector containing oneFile items and
representing
   * the content of the current directory
   */
  public abstract Vector fileList();

  /**
   * sortFileList take a vector of oneFile and sorts its
component.
   * Directory are put first and then non directory files...
   */
  public Vector sortFileList(Vector list)
  {
    Vector directories=new Vector(); /* the vector where goes all
directories */
    Vector files=new Vector(); /* the vector where goes all non-
directories */

    for(int i=0;i<list.size();i++)
      {
        oneFile thisElement;

        thisElement=(oneFile)list.elementAt(i);
        if(thisElement!=null){
          if(thisElement.isDir)
            {
              directories.addElement(thisElement);
            }
          else
            {
              files.addElement(thisElement);
            } /* else thisElement.isdir==false */
        } /*thisElement!=null */
        else
          {

          }
        } /* for i<list.size() */
    for(int i=0;i<files.size();i++)
      {
        directories.addElement(files.elementAt(i));
      } /* for i<files.size() */

    return directories;
  } /* sortFileList */

  /**
   * chdir changes the current directory to the one specified in
target
   */
  protected abstract void chdir(String target);


  /**
   * cdup goes to the upper directory
   */
  protected abstract void cdup();

  /**
   * goToTarget goes to the directory pointed by target assuming
that we are currently in the
   * directry pointed by currentTreePath
   */
  public void goToTarget(TreePath target)
  {
    int idx=0; /* the index of the element currently analysed in
the path */
    int targMax=target.getPathCount(); /* the number of element of
the target */
    int curMax=currentTreePath.getPathCount(); /* the number of
element of the current path */


    /* first we look at all the things that these two path have in
common */
    while((idx<targMax)&&(idx<curMax)&&(target.getPathComponent
(idx)==currentTreePath.getPathComponent(idx)))
      {
        idx++;
      } /* while */


    /* then we go up for all the part of currentTreePath that are
left */
    if(idx<curMax)
      {
        for(int j=idx;j<curMax;j++)
          {
            cdup();
          }
        currentPath=getWD();
      }

    /* then we go down for all element in target */
    for(int j=idx;j<targMax;j++)
      {
        String cdName;

        DefaultMutableTreeNode dmtn;
        dmtn=(DefaultMutableTreeNode) target.getPathComponent(j);

        cdName=(((oneFile)dmtn.getUserObject()).name); /* the name
of the directory to which we have changed to */
        chdir(cdName);

        if(localPathSeparator==null)
          {
            /* The path Separator will be determined */
            String svdPath=currentPath; /* the old path is saved
*/
            currentPath=getWD();

            if(currentPath.startsWith(svdPath)==false)
              { /* this case has to be implemented later */
                localPathSeparator=null;
                AFF(("error on the path name"));
              }
            else
              { /*
                 * the local separator is the difference between:
                 *     the old pathname+the name of the directory
and
                 *     the new path name
                 */
                if(currentPath.endsWith(cdName))
                  {
                    localPathSeparator=currentPath.substring
(svdPath.length(),currentPath.length()-cdName.length());
                  }
                else
                  {
                    localPathSeparator=currentPath.substring
(svdPath.length()+cdName.length());
                    currentPath=currentPath.substring
(currentPath.length()-localPathSeparator.length()); /* the
currentPath must not finish with the separator */
                  }
              } /* currentPath starts with svdPath */
          } /* localPathSeparator==null */

        currentPath.concat(localPathSeparator);
        currentPath.concat(cdName);
      } /* for chdir */

    currentTreePath=target;
  } /* goToTarget */

  /**
   * expandDir changes the current directory to the one specified
in Target
   *   and reads the contents of this new dir if needed
   */
  public void expandDir(TreePath target)
  {
    /*
    AFF((target.toString()));
    AFF((((DefaultMutableTreeNode)target.getLastPathComponent()).
toString()));
    AFF((((DefaultMutableTreeNode)target.getLastPathComponent()).
getUserObject().toString()));
    */

    if(((DefaultMutableTreeNode)target.getLastPathComponent()).
getUserObject()!=null)
      {
        if(((oneFile)((DefaultMutableTreeNode)
target.getLastPathComponent()).getUserObject()).subDir==null)
          {
            setDir(target);
          }
      }
    else
      { /*object == null */
        AFF(("error!!! "+(target.getLastPathComponent()).toString
()));
      }
  } /* expandDir */

  /**
   * setDir reads the content of the content of the directory
specified by target
   * this may be used the refresh the content of the current
directory
   */
  public void setDir(TreePath target)
  {
    goToTarget(target); /* first we go the the directory pointed
by "target" */

    oneFile of=((oneFile)((DefaultMutableTreeNode)
target.getLastPathComponent()).getUserObject());
    of.subDir=sortFileList(fileList());

    DefaultMutableTreeNode parentNode=(DefaultMutableTreeNode)
((DefaultMutableTreeNode)target.getLastPathComponent()).getParent
();
    DefaultMutableTreeNode thisNode=(DefaultMutableTreeNode)
((DefaultMutableTreeNode)target.getLastPathComponent());
    thisNode=(buildNode(of.subDir,thisNode));

    boolean[] expandedRows=new boolean[((JTree)
parent.connectListCompo).getRowCount()];
    int row=((JTree)parent.connectListCompo).getRowForPath
(currentTreePath);
    for (int r=0;r<expandedRows.length;r++)
      {
        expandedRows[r]=((JTree)parent.connectListCompo).
isExpanded(r);
        if(expandedRows[r]==true)
          {
            //            AFF((" row "+r+" expanded"));
          }
      } /* for all rows */
    parent.makeCompo();

    /*    for (int r=0;r<expandedRows.length;r++)
       {
         if((expandedRows[r])&&(r!=row))
           {
AFF(("expanding "+r));
             ((JTree)parent.connectListCompo).expandRow(r);
           }
       }*/ /* for all rows */
    //AFF(("expanding my row"));
             ((JTree)parent.connectListCompo).expandRow(row);
             //AFF(("all rows have been expanded"));

  } /* setDir */


  /**
   * buildNode builds the node representing the given vector.
   * This function is recursively called if there are sub-nodes
   */
  public DefaultMutableTreeNode buildNode(Vector
Node,DefaultMutableTreeNode top)
  {

    int index; /* The index of the last element read in the vector
*/
    oneFile theObject;
    DefaultMutableTreeNode currentNode;

    currentNode=null;
    index=0;

    if (Node==null)
      { /* if this has not already been explored */
        /* nothing is added */
        return top;
      }
    else
      {
        while(index<Node.size())
          {
            theObject=(oneFile)Node.elementAt(index);
            index++;

            if(theObject!=null)
              {
                /* adding the name */
                currentNode = new DefaultMutableTreeNode
(theObject);
                if (theObject.isDir) /* has subDir */
                  {
                    currentNode.setAllowsChildren(true);
                    if(theObject.subDir!=null)
                      { /* if the children are known, they are
added */
                    top.add(buildNode((Vector)
theObject.subDir,currentNode));
```

```
                      }
                    else
                      {
                        currentNode.add(new DefaultMutableTreeNode
(NOT_LOADED_SUBDIR));
                      }
                  } /* vector */
                else
                  {
                    currentNode.setAllowsChildren(false);
                  }
                top.add(currentNode);
              } /* if object != null */
            else
              {
                AFF(("null object!!!"));
              }
          } /* while */
        return top;
      }

  } /* buildNode */

  /**
   * BuildTheTree builds the tree representing the current
connection
   */
  public DefaultMutableTreeNode buildTheTree()
  {
    chkFileStruct();
    top = new DefaultMutableTreeNode(((oneFile)((Vector)
fileStruct).elementAt(0))); /* the 0th element of fileStruct is
always the name of the root, whereas the firstone is the vector
containing the content of the current directory */

    return buildNode(((oneFile)((Vector)fileStruct).elementAt(0)).
subDir,top);

  } /* buildTheTree */

  /**
   * waitCursor changes the current cursor to the wait cursor
   */
  public void waitCursor()
  {
    parent.waitCursor();
  } /* waitCursor */


  /**
   * Checks if the file structure has already been read or not
   */
  public void chkFileStruct()
  {
    if(fileStruct==null)
      {
        oneFile rootFile=new oneFile(getWD());
        rootFile.subDir=sortFileList(fileList());
        rootFile.isDir=true;
        fileStruct = new Vector();
        fileStruct.addElement(rootFile);
      } /* fileStruct == null */
  } /* chkFileStructure */


  /**
   * deletes the file whose path & name is passed as argument
   */
  public abstract void delete(String fileName);

} /* end class */
```

✯ ✯ ✯ ✯ ✯

```
#include "c:\disk_f\programmation\Java\FTP\const_ftp.java"
//#include "c:\disk_f\programmation\Java\FTP\consts.java"
#include "c:\disk_f\programmation\Java\FTP\textes.java"

import java.util.*;
import java.io.*;
import java.io.InputStream.*;
import java.awt.*;
import java.net.*;
import module_com.*;
import aHost.*;
import addHost.*;
import connections.*;
//import com.sun.java.swing.*;
import com.sun.java.swing.JTree;
import com.sun.java.swing.tree.DefaultMutableTreeNode;
import com.sun.java.swing.JScrollPane;
import com.sun.java.swing.JPanel;
import com.sun.java.swing.JFrame;

public class module_ftp extends module_com
{
  /**
   * The module ftp manages the FTP connection
   */

  Socket soc;
  ServerSocket DTP;

  InputStream IS;
  OutputStream OS;
  DataInputStream DIS;
  DataOutputStream DOS;
  PrintStream PS;

  boolean connected;

  String message; /* contains the last message returned by the
computer */

  host_list hostList;
  String lastCommande; /** the last command sent to the host */

  public module_ftp(host_list hl,connections par)
  {
    fileStruct=null;
    hostList=hl;
    parent=par;
    localPathSeparator=null;

    try /* first we initialize the DTP that listen for data from
the other host */
      {
        DTP = new ServerSocket(0);
      }
      catch (IOException e)
        {
          DTP = null;
        }
  }


  public void openPort()
  {
    int portNumber; /* this is used to open the Server Socket */
    String addr; /* this string will be used to save the address
of the port */

    try
      {
        addr=((InetAddress.getLocalHost()).getHostAddress());
      }
    catch (UnknownHostException e)
      {
        AFF(("pb hote inconnu:"+e.getMessage()));
        addr=null;
      }
    addr=addr.replace('.',',');
    portNumber=DTP.getLocalPort();
    addr+=","+(portNumber/256)+","+(portNumber%256); /* we define
the Data Port */
    send("PORT "+addr);
    receive();
  } /* openPort */

  public boolean openHost(aHost theHost)
  {
     int resp; /* The response received from the Host */

     resp=999; /* initialization to an error code */

     if((theHost.machine==null)||((theHost.machine).equals
(NEW_HOST_MACHINE)))
       {
         AFF(("theHost est null"));
         theHost=null;
         requestInfo(theHost);
         connected=false;
         return false;
       }
     try  /* we try to open the host */
       {
         soc = new Socket(theHost.machine,FTPPORT);  /* if the
open command has been sent, then the host name is opened on port
FTPPORT */

         IS=soc.getInputStream();
         OS=soc.getOutputStream();
         DIS = new DataInputStream(IS);
         DOS = new DataOutputStream(OS);
         PS=new PrintStream(OS);

        connected=true;

       }
     catch (UnknownHostException e)
       {
         AFF(("Hote inconnu")); /* if the host is unknown */
       }
     catch (IOException e)
       {
         AFF(("pb IO")); /* if an IO problem has occured */
       }
     resp=receive();
     openPort();
     if((resp==220)&&(theHost.login!=null)){
       send("USER "+theHost.login);
       resp=receive();
       if(resp==331)
         {
           if(theHost.password!=null)
             {
               send("PASS "+theHost.password);
               resp=receive();
             } /* pass */
           else
             {
               requestInfo(theHost);
             } /* password == null */
         } /* 331 */
     } /* login user */
     else
       {
         requestInfo(theHost);
       } /* login == null */
     if(resp==230)
       {
         send("TYPE I"); /* type is set to image, to allow the
transfer of binary data */
         receive();
         parent.confirmConnection();
       }
     return (resp==230); /* if the last code received is a 230
message then the connection is OK */
  } /* openHost */

  public void closeHost()
  {
    try  /* we try to close the host */
      {
        send("QUIT");
        receive();
        connected=false;
        soc.close();
        DTP.close();
      }
    catch (IOException e)
      {
        AFF(("pb IO")); /* if an IO problem has occured */
      }
    fileStruct=null;
  } /* closeHost */


  /**
   * The command function receives the commands from the other
modules
   */
  public boolean command(String commande)
  {
    boolean processed=false;
    byte i=0;

    while (commande.substring(i)==" ")i++;
    if(commande.length()>=(i+4)){
      if (((commande.substring(i,i+4)).equalsIgnoreCase(OPEN2)))
        {
          Integer j;
          i+=4;
          i++;
          AFF(("opening port "+FTPPORT+" of lisa")); /* if the
open command has been sent, then the host name is opened on port
FTPPORT */
          try  /* we try to open the host */
            {
              /* if the open command has been sent, then the host
name is opened on port FTPPORT */
              soc = new Socket("lisa.cs.mcgill.ca",FTPPORT);
              IS=soc.getInputStream();
              OS=soc.getOutputStream();
              DIS = new DataInputStream(IS);
              DOS = new DataOutputStream(OS);
              PS=new PrintStream(OS);

              connected=true;

            }
          catch (UnknownHostException e)
            {
              AFF(("Hote inconnu")); /* if the host is unknown */
            }
          catch (IOException e)
            {
              AFF(("pb IO")); /* if an IO problem has occured */
            }
          receive();
          openPort();
          send("USER nicolas");
          receive();
          send("PASS vero12");
          receive();
          processed=true;
          i=0;
        }
      if (((commande.substring(i,i+4)).equalsIgnoreCase(OPEN)))
        {
          Integer j;
          i+=4;
          i++;
          AFF(("opening port "+FTPPORT+" of "+commande.substring
(i,(commande.length())))); /* if the open command has been sent,
then the host name is opened on port FTPPORT */
          try  /* we try to open the host */
            {
              /* if the open command has been sent, then the host
name is opened on port FTPPORT */
              soc = new Socket(commande.substring(i,
(commande.length())).trim(),FTPPORT);
              IS=soc.getInputStream();
              OS=soc.getOutputStream();
              DIS = new DataInputStream(IS);
              DOS = new DataOutputStream(OS);
              PS=new PrintStream(OS);

              connected=true;
            }
          catch (UnknownHostException e)
            {
              AFF(("Hote inconnu")); /* if the host is unknown */
              AFF(("message: "+e.getMessage()));
            }
          catch (IOException e)
            {
              AFF(("pb IO")); /* if an IO problem has occured */
            }
          receive();
          openPort();
          processed=true;
        }
    }
    if(commande.length()>=(i+5)){
      if (((commande.substring(i,i+5)).equalsIgnoreCase(CLOSE))){
        try  /* we try to close the host */
          {
            command("QUIT");
            soc.close();
            DTP.close();
          }
        catch (IOException e)
          {
            AFF(("pb IO")); /* if an IO problem has occured */
          }
        processed=true;
      }
    }
    if(processed==false)
```

```
      {
        int recv;
        send(commande.substring(0).trim());
        recv=standartReceive();
        if(recv==150)
          {
            dataReceive(null);
            recv=standartReceive();
          }
        processed=true;
      }

    return processed;
  } /* end command */

  public void send(String data)
  {
    lastCommande=data;
    if (connected){
      try
        {
          PS.print(data); /* the data is sent to the host */
          OS.write(10);   /* the Line feed caracter */
        }
      catch (IOException e)
        {
          AFF(("pb IO")); /* if an IO problem has occured */
        }
    } else {
      AFF(("non connecte"));
    }
  } /* send */

  public int standartReceive()
  { /** standartReceive receives data on the command port.
     * The function returns the code returned by the server and 
the message is saved
     * in the global variable message.
     */

    boolean transmission=true; /* this variable records if we are 
still transmitting data or not */
    byte BData[] = new byte[1];
    String head=""; /* the reply-code is saved here, (this is 
necessary to deal with multi-line reply */
    try
      {
        while (transmission){
          message="";
          DIS.read(BData,0,1);
          while((BData[0] != 10)){
            message+=(char)BData[0];
            DIS.read(BData,0,1);
          }
          AFFS((message));
          transmission=((message.substring(3,4).equals("-"))||
(((head.equals("")==false))&&(message.substring(0,3).equals(head)
==false)));
          if((transmission)&&(head.equals(""))){
            head=message.substring(0,3);
          }
        } /* while transmission */
        head=(message.substring(0,3));
      } /* try */
    catch (IOException e)
      {
        AFF(("pb IO \n")); /* if an IO problem has occured */
        head="599"; /* 5xx are the errors */
      } /* catch */
        return (Integer.decode(head)).intValue();
  } /* standartReceive */

 /** dataReceive receives stuff on the Data Port.
     *  If the string targetFile is null then the data are
returned in a vector
     *      (each item being separated by a CR) else it saves the
data into a file
     *  The vector returned contains the data if the parameter
wasn't null else
     *      it contains the name of the file.
     *  If an error occured, a null vector is returned
     */
  public Vector dataReceive(String targetFile)
  {
    boolean listen=false;
    byte BData[] = new byte[BUFFER_SIZE];
    //    String data;
    RandomAccessFile tFile;
    Vector dataVect;
    InputStream DTPIS;
    Socket DTP_input;
    //    FilterInputStream DTPDIS;

    tFile = null;
    dataVect = new Vector();

    if (targetFile!=null)
      {
        try  /* let's try to open the file were the data will be
saved */
          {
            AFF(("saving the data in "+targetFile));
            tFile = new RandomAccessFile(targetFile,"rw");
            /** at the moment no check is performed to ensure that
the file doesn't already exist */
          }
        catch  (IOException e)
          {
            tFile = null;
            AFF((e.getMessage()));
            return null;
          }
        dataVect.addElement(targetFile);
      } /* targetFile != null */

    try
      {
        DTP.setSoTimeout(SOCKET_TIMEOUT);
        DTP_input=DTP.accept();

        DTPIS=DTP_input.getInputStream();
        //        DTPDIS = new FilterInputStream(DTPIS);
      } /* try */
    catch (IOException e)
      {
        AFF((e.getMessage())); /* if an IO problem has occured */
        return null;
      }

    //    data="";

    boolean escape=false;

    try
      {
        while(!escape)
          {
            int lengthRead; /* the number of byte read */

            lengthRead=DTPIS.read(BData);
            escape=(lengthRead==-1); /* the end of stream is
reached if read returns -1 */

            if (lengthRead>0)
              {
                if(targetFile==null)
                  {
                    String data=new String(BData,0,lengthRead);
                    String subData; /* will contain a substring of
Data */

                    while(data.length()>0)
                      {
                        int min; /* points the fisrt occurence of
an unwanted char */
                        int pos; /* points the position of an
unwanted char */
                        min=data.indexOf('\n');
                        min=((min==-1)?data.length():min);
                        pos=data.indexOf('\f');
                        min=((pos<min)&&(pos!=-1)?pos:min);
                        pos=data.indexOf('\r');
                        min=((pos<min)&&(pos!=-1)?pos:min);
                        /* now the first unwanted char has been
found; data will be cut here */

                        if(min>0)
                          {
                            subData=data.substring(0,min);
                            data=data.substring(min);

                            if(subData.length()>0)
dataVect.addElement(subData);
                            //                            if
(subData.length()>0)AFF((subData));
                          }
                        else
```

```
                          { /* if min==0 then the first char is
unwanted and this discarded */
                            data=data.substring(min+1);
                          }
                      }
                 }  /* targetFile == null */
               else
                 {
                   AFF(("number of bytes received "+lengthRead));
                   tFile.write(BData,0,lengthRead);
                   BData=new byte[BUFFER_SIZE];
                 }  /* targetFile != null */
             } /* if lengthRead > 0 */
       } /* while !escape */
       DTPIS.close();
       DTP_input.close();
       DTP.close();
       DTP = new ServerSocket(0);
       openPort();
     } /* try */
   catch (IOException e)
     {
       AFF((e.getMessage())); /* if an IO problem has occured */
       return null;
     }
   return dataVect;
 } /* dataReceive */


 public int receive()
 {
   int returnedValue;

   if (connected) {
     returnedValue=standartReceive();
     if(returnedValue==421) /* 421 is the value returned in case
of timeout */
       {
         AFF(("connecion remotely closed after timeout; re-
opening"));
         if(!(openHost(parent.theHost)))
           {
             connected=false;
             parent.connected=false;
             return 599;
           }
         else
           {
             send(lastCommande);
             returnedValue=standartReceive();
             if(returnedValue==421)
               {
                 connected=false;
                 parent.connected=false;
                 return 599;
               }
             else
               {
                 return returnedValue;
```

```
                }
            }
        } /* if a timeout occured */
      else
        {
          return returnedValue;
        }
    } /* if connected */
    else
      {
        return 599; /* if not connected then an error message is
returned */
      }
  } /* receive */


  /**
   * sortFileNames receives a list of fileNames and sorts them
alphabeticaly
   */
  private Vector sortFileNames(Vector list)
  {
    Vector toBeReturned=new Vector();
    int idx;
    int row;

    for(idx=0;idx<list.size();idx++)
      {
        String element=(String)list.elementAt(idx);

        row=0;
        while((row<toBeReturned.size())&&(element.compareTo
((String)toBeReturned.elementAt(row))>0))
          {
            row++;
          } /* while */
        toBeReturned.insertElementAt(element,row);
      } /* for */
    return toBeReturned;
  } /* sortFileName */


  /**
   * listDir is a function that returns the content of the current
directory
   */
  public Vector listDir()
  {
    int answer;
    send("NLST");
    answer=receive();
    if(answer==150)
      { /* 150 means that the list directory data will be sent on
the DTP */
        receive();
        return sortFileNames(dataReceive(null));
      }
    else
      {
        /* else the list is not available */
        return null;
      }
  } /* listDir */

  /**
   * expandedListDir is a function that returns the content of the
current directory
   *   returned by the function LIST. According to the FTP
standard, this content is NOT
   *   for automated processing...
   */
  public Vector expandedListDir()
  {
    send("LIST");
    receive();
    receive();
    return dataReceive(null);
  } /* expandedListDir */

  /**
   * fileList returns a vector containing oneFile items and
representing
   * the content of the current directory
   */
  public Vector fileList()
  {
    Vector simpleList; /** the vector containing the list from
NLST */
    Vector advancedList; /** the vector containing the list from
LIST */

    Vector toBeReturned=new Vector(); /** the vector that contains
the oneFiles */

    simpleList=listDir();
    advancedList=expandedListDir();

    if(simpleList==null)
      { /* a null listDir probably means an empty directory. Let's
assume that. */

        /* one file is put in the directory to say the directory
is empty */
        toBeReturned.addElement(new oneFile
(EMPTY_DIRECTORY_CONTENT));
      } /* listDir==null */
    else
      { /* listDir!=null */
        if((((advancedList.elementAt(0)).toString()).length())
<SIZE_ELEMENT_SKIPPED_IN_LIST))
          { /* if the first element is very short it may be the
total number of files. It is then removed */
            advancedList.removeElementAt(0);
          }
        if(simpleList.size()!=advancedList.size())
          { /* if there is a discrepency on the size, one more try
is done */
```

```
            AFF(("discrepency "+simpleList.size()+"
"+advancedList.size()));
            simpleList=listDir();
            advancedList=expandedListDir();
            if(((((advancedList.elementAt(0)).toString()).length
())<15))
              { /* if the first element is very short it may be
the total number of files. It is then removed */
                advancedList.removeElementAt(0);
              }
          } /* discrepancy on the lists' size */
        /*
         * now the size of simpleList will be assumed to be the
directory's size
         * but the order of the LIST (advancedList) will be
kept...
         */
        toBeReturned.setSize(simpleList.size());
        for(int i=0;i<simpleList.size();i++)
          {
            oneFile of=new oneFile((String)simpleList.elementAt
(i));
            int j=0;
            while((j>=0)&&(j<advancedList.size()))
              { /* we try to find which element is the one
corresponding to the simpleList element */
                if(of.isName((String)advancedList.elementAt(j)))
                  {
                    of.analyse((String)advancedList.elementAt(j));
                    advancedList.setElementAt("",j);
                    if(j<toBeReturned.size())
                      {
                        toBeReturned.setElementAt(of,j);
                      }
                    else
                      {
                    toBeReturned.addElement(of);
                      }
                    j=-1;
                  }
                else
              {
                j++;
              }
              } /* while j<advancedList.size() */
            if(j!=-1)
              {/* if this element has not already been added it is
added now */
                toBeReturned.addElement(of);
                //                AFF(("serious problem of
determination: "+of.name));
              }
            if(of.rights==null)
              {
                //                AFF(("problem of determination:
"+of.name));
              }
          } /* for simple list */
      } /* listDir!=null */
```

```
    return toBeReturned;
  } /* fileList */


  /**
   * getWD returns the name of the current directory on the server
   * and sets currentPath to the right value
   */
  public String getWD()
  {
    String toBeReturned;
    int idx; /* an index used to read message */

    send("PWD"); /* asks the server to print its current working
directory */
    if(receive()==257)
      { /* extraction of the name of the directory */
        idx=message.indexOf("\""); /* the first double quote is
the begining of the wd */
        idx=message.indexOf("\"",(idx+1)); /* seeking for the next
double-quote */
        while(message.charAt(idx+1)=='\"') /* if there are two
double quotes, this occurence has to be skipped */
          {
            idx=message.indexOf("\"",(idx+1)); /* seeking for the
next double-quote */
          } /* skipping the embedded double quotes */
        toBeReturned=(message.substring((message.indexOf("\"")+1),
idx));
        idx=0;
        while(toBeReturned.indexOf("\"\"",idx)!=(-1))
          {
            idx=toBeReturned.indexOf("\"\"",idx);
            toBeReturned=(toBeReturned.substring(0,idx)).concat
(toBeReturned.substring(idx+1));
          } /* deleting the extra quotes*/
        currentPath=toBeReturned;
        return toBeReturned;
      } /* receive == 257 */
    else
      { /* an error has occured */
        return null;
      } /* error */
  } /* getWD */

  /**
   * receiveFile receives a file named fileName from the remote
host in sourcePath
   * to the local host in the targetPath.
   * NO ERROR ANALYSIS IS DONE AT THE MOMENT!!!
   */
  public void receiveFile(String sourcePath,String fileName,String
targetPath)
  {
    send("RETR "+sourcePath);
    receive();
    dataReceive(targetPath.concat(fileName));
    receive();
    AFF(("file received..."));
  } /* receiveFile */

  /**
   * sendFile sends a file named fileName from the local host in
the directory sourcePath
   * to the current directory of the host.
   */
  public void sendFile(String sourcePath,String fileName)
  {
    RandomAccessFile sourceFile;
    boolean escape;
    byte buffer[] = new byte[BUFFER_SIZE];
    OutputStream  DTPOS;
    Socket DTP_output;
    //    DataOutputStream  DTPDOS;

    send("STOR "+fileName);
    receive();

    escape=false;
    try  /* let's try to open the file containing the source */
      {
        sourceFile = new RandomAccessFile(sourcePath,"r");
        AFF(("opening "+sourcePath));
      }
    catch  (IOException e)
      {
        sourceFile = null;
        AFF((e.getMessage()));
        escape=true;
      }
    catch  (SecurityException e)
      {
        sourceFile = null;
        AFF((e.getMessage()));
        escape=true;
      }

    try
      {
        DTP.setSoTimeout(SOCKET_TIMEOUT);
        DTP_output=DTP.accept();

        DTPOS=DTP_output.getOutputStream();
        //        DTPDOS = new DataOutputStream(DTPOS);
      }
    catch (IOException e)
      {
        AFF((e.getMessage())); /* if an IO problem has occured */
        escape=true;
        DTP_output=null;

        DTPOS=null;
        //        DTPDOS = null;
      }

    AFF(("debut de l'envoi du fichier"));
    while(!escape){
      int lengthRead=0; /* the number of byte read */

      try /* let's try to read the file*/
        {
          lengthRead=sourceFile.read(buffer);/* the line is read
from the file */
          escape=(lengthRead==-1);/* the end of stream is reached
if read returns -1 */
        }
      catch (EOFException e)
        {
          AFF((e.getMessage()));
          escape=true;
        }
      catch (IOException e)
        {
          AFF((e.getMessage()));
          escape=true;
        }

      if(lengthRead>0)
        {
          try
            {
              AFF(("avant write "+lengthRead));
              DTPOS.write(buffer,0,lengthRead);
              AFF(("apres write"));
            } /* try */
          catch  (IOException e)
            {
              AFF((e.getMessage()));
              escape=true;
            } /* catch IOEception */
          AFF((buffer));
        } /* lengthRead > 0 */
          if(lengthRead==0)
            {
              AFF(("buffer length 0"));
              escape=true;
            }
          buffer=new byte[BUFFER_SIZE];

    } /* while !escape */

    try
      {
        DTPOS.flush();
        DTPOS.close();
        //        DTPDOS.close();
        DTP_output.close();
        DTP.close();
        DTP = new ServerSocket(0);
        openPort();
      } /* try */
    catch (IOException e)
      {
        AFF((e.getMessage())); /* if an IO problem has occured */
      }  /* catch close */
    /*
```

```
    try
      {
        sourceFile.close();
      }
    catch (IOException e)
      {
        AFF((e.getMessage()));
      } */ /* catch close */
    receive();
    AFF(("file sent..."));
  } /*sendFile */


  /**
   * pasvRemote makes this host enters in PASV mode, and returns a
String that is its port number
   */
  public String pasvRemote()
  {
    /* the global variable message contains the answer of the host
to the PASV command */

    send("PASV");
    if(receive()==227)  /* 227 is the normal reply to PASV */
      {
        //        AFF(("reponse: "+message));
        //AFF((" ( "+message.lastIndexOf("(")));
        //AFF((" ) "+message.lastIndexOf(")")));
        AFF(("port returned: "+message.substring
(message.lastIndexOf("(")+1,message.lastIndexOf(")"))));
        return message.substring(message.lastIndexOf("(")
+1,message.lastIndexOf(")"));
      }
    else
      {
        AFF(("an error occured; error message: "+message));
        return "";
      }
  } /* pasvRemote */

  /**
   * receiveRemote receives a file named fileName on its data
port, but coming from a remote host
   * so this host enters in PASV mode, and returns a String that
is its port number
   */
  public void receiveRemote(String fileName)
  {
    send("STOR "+fileName);
    receive();
  } /* receiveRemote */

  /**
   * sendRemote sends a file named fileName on its data port which
has been opened on port port
   */
  public void sendRemote(String port,String sourcePath)
  {
    send("PORT "+port);
    receive();
    send("RETR "+sourcePath);
    receive();
  } /* sendRemote */


  /**
   * chdir changes the current directory to the one specified in
target
   */
  public void chdir(String target)
  {
    if(!target.equals(""))
      { /* if the target is not an empty string */
        send("CWD "+target);
        receive();
      }
  } /* chdir */

  /**
   * cdup goes to the upper directory
   */
  public void cdup()
  {
    send("CDUP");
    receive();
  } /* cdup */

  /**
   * requestInfo
   */
  public void requestInfo(aHost theHost)
    {
      addHost newHost = new addHost(this,theHost,hostList);
      return;
    } /* requestInfo */

  /**
   * deletes the file whose path & name is passed as argument
   */
  public  void delete(String fileName)
  {
    send("DELE "+fileName);
    receive();
  } /* delete */

} /* end class */
```

✮ ✮ ✮ ✮ ✮

```
#include "c:\disk_f\programmation\Java\FTP\textes.java"

import java.io.*;
import aHost.*;
import java.util.*;
import com.sun.java.swing.tree.DefaultMutableTreeNode;
//import com.sun.java.swing.tree.*;
```

```
//import com.sun.java.swing.*;
import connections.*;

public class module_local extends module_com
{
  /**
   * The module local manages the communications with the local
host
   */
  public module_local(connections par)
  {
    parent = par;
    localPathSeparator=null;
  }

  public void openPort()
  { /* no operations needed as the local machine is always opened!
*/
  }/* openPort */

  public void closeHost()
  { /* no operations needed as the local machine can not be closed
*/
  }/* closeHost */

  public int receive()
  { /* no operations needed as the local machine never returns
string answers to the commands sent  */
    /*if this method is called, an error code is returned*/
    return 599;
  }/* openPort */

  public boolean openHost(aHost theHost)
  {
    parent.confirmConnection();

    return true; /* the local host can always be accessed! */
  } /* openHost */


  /**
   * listDir is a function that returns the content of the current
directory
   * obsolete?
   */
  private Vector ObsListDir()
  {
    File thisDir;
    boolean noError;
    String[] listFiles;
    byte idx;
    Vector toBeReturned;

    //    try  /* let's try to open the local host current dir */
    thisDir = new File(currentPath);
    listFiles = thisDir.list();

    idx=0;
    toBeReturned = new Vector();
    while(idx<(listFiles.length))
      {
        toBeReturned.addElement(listFiles[idx]);
        idx++;
      }

    return toBeReturned;
  } /* listDir */



  /**
   * fileList returns a vector containing oneFile items and
representing
   * the content of the current directory
   */
  public Vector fileList()
  {
    String localCurrentPath=currentPath;
    File thisDir;
    boolean noError;
    String[] listFiles;
    byte idx;
    Vector toBeReturned;
    oneFile thisFile;

    if(localCurrentPath.endsWith(localPathSeparator))
      {
        localCurrentPath=localCurrentPath.substring(0,
(localCurrentPath.length()-localPathSeparator.length()));
      }

    //    AFF((localCurrentPath));

    thisDir = new File(localCurrentPath);
    listFiles = thisDir.list();



    idx=0;
    toBeReturned = new Vector();
    if(listFiles!=null)
      {
        while(idx<(listFiles.length))
          {
            thisFile=new oneFile(listFiles[idx]);
            thisDir=new File(localCurrentPath,listFiles[idx]);
            thisFile.size=thisDir.length();
            thisFile.isDir=thisDir.isDirectory();
            thisFile.rights=new String("");
            thisFile.rights+=(thisDir.isDirectory()?"d":"-");
            thisFile.rights+=(thisDir.canRead()?"r":"-");
            thisFile.rights+=(thisDir.canWrite()?"w":"-");
            toBeReturned.addElement(thisFile);
            idx++;
          } /* while */
      } /* listFiles != null */
    else
      {
        AFF(("list null "+localCurrentPath));
      }
    return toBeReturned;
  } /* fileList */

  /**
   * getWD returns the name of the current directory on the server
   * and sets currentPath to the right value
   */
  public String getWD()
  {
    //    AFF(("currentPath :"+currentPath));

    if((localPathSeparator!=null)&&(currentPath!=null))
      {
        if(currentPath.endsWith(localPathSeparator)==false)
          {
            currentPath=currentPath.concat(localPathSeparator);
            //    AFF(("currentPath :"+currentPath));
          }
      }

    File F=new File(currentPath);
    localPathSeparator=F.separator;
    currentPath=F.getAbsolutePath();

    /* the path separator is to be removed of the path name */
    if((currentPath.endsWith(localPathSeparator))&&
(currentPath.length()>3))
      {
        currentPath=currentPath.substring(0,(currentPath.length()-
localPathSeparator.length()));
      }
    //    AFF(("currentPath :"+currentPath));
    return currentPath;
  } /* getWD */


  /**
   * chdir changes the current directory to the one specified in
target
   */
  public void chdir(String target)
  {
    //    AFF(("currentPath "+currentPath+"    target "+target));
    if(target.startsWith(currentPath))
      {
        currentPath=target;
      }
    else
      {
        currentPath=(new File(currentPath,target)).getPath();
      }
    //    AFF(("currentPath "+currentPath));
  } /* chdir */


  /**
   * cdup goes to the upper directory
   */
  public void cdup()
  {
    File F=new File(currentPath);
    if(F.getParent()!=null)currentPath=F.getParent();

    /* the path separator is to be removed of the path name */
    if((currentPath.endsWith(localPathSeparator))&&
(currentPath.length()>4))
      {
        currentPath=currentPath.substring(0,(currentPath.length()-
localPathSeparator.length()));
      }

    //    AFF(("up "+currentPath));

  } /* cdup */

  /**
   * deletes the file whose path & name is passed as argument
   */
    public void delete(String fileName)
  {
    AFF(("deleting "+fileName));
    File thisFile = new File(fileName);

    thisFile.delete();
  } /* delete */


} /* end class */
```

✯✯✯✯✯

```
  /**
   * the textes.java file contains all the Strings used in the
application.
   * As the textes are all in a same file, it will be easier to
translate
   * this application in an other language
   */

#define TITRE "Nico's FTP"
#define MENU_MAIN "Main"
#define MENU_MAIN_PARAMS "Parametres"
#define MENU_MAIN_PARAMS1 "Colors"
#define MENU_MAIN_PARAMS2 "Font"
#define MENU_MAIN_QUIT "Quit (Alt+F4)"

#define MENU_CONNECT "Connection"
#define MENU_CONNECT_NEW "New"
#define MENU_HELP "Help"
#define MENU_HELP_FTP "FTP Help"
#define MENU_HELP_CREDITS "Credits"
#define MENU_HELP_BUGS "Bugs & FeedBack..."
#define MENU_HELP_REG "Registration"
```

```
#define LOCAL_NAME "Local Host"
#define LOCAL_MACHINE "local"  // This line must not be translated

#define NEW_HOST_NAME "Connect to another Host"
#define NEW_HOST_MACHINE "newHost"  // This line must not be
translated

#define TITLE_NEW_HOST "Addition of a new host"
#define TEXTE_NEW_HOST "Please enter the informations about the
new host"
#define TEXTE_NEW_HOST_NAME "Name"
#define TEXTE_NEW_HOST_MACHINE "Machine"
#define TEXTE_NEW_HOST_LOGIN "Login"
#define TEXTE_NEW_HOST_PASSWORD "PassWord"
#define ECHO_CHARACTER 'x'
#define BUTTON_ADD_NEW_HOST "Add this Host to my .netrc but do not
connect"
#define BUTTON_ADD_CONNECT_NEW_HOST "Add this Host to my .netrc
and connect"
#define BUTTON_CONNECT_NEW_HOST "Just connect to this Host"
#define WIDTH_NEW_HOST 600
#define HEIGHT_NEW_HOST 300
#define SIZE_FIELD_NEW_HOST 40

#define INITIAL_WIDTH 800 /* The initial width */
#define INITIAL_HEIGHT 550 /* The initial heigth */
#define INITIAL_CONNECTIONS 2  /* The initial number of connexions
*/
#define BORDER 10 /* The thickness of the border between the
connexions */
#define MENU_HEIGHT 40 /* The height of the Menu bar */
#define RECORD_FILE ".netrc" /* The name of the record file */

#define NO_ERROR 0
#define ERROR_NOT_AN_ELEMENT 1 /* this error is returned when we
try to add to an host something that doesn't correspond to any
host propertie */
#define ERROR_ALREADY_FIXED 2  /* this error is returned when we
try to add to an host something that already correspond to one of
this host properties */

//#define DISPLAY_TYPE GridLayout
#define DISPLAY_TYPE GridBagLayout
//#define DISPLAY_TYPE FlowLayout

#define AFF(x) System.out.println("Line: "+__LINE__+"
("+__FILE__+")");System.out.println x ;
#define AFFS(x) System.out.println x ;
#define TAILLE_POLICE 14 /** Taille de la police fixee
autoritairement; a changer...*/

#define BUFFER_SIZE   4096 /* files are copied by blocks of 4Ko */
#define SOCKET_TIMEOUT 1000

#define FTPPORT 21 /* the default FTP port */

// Values used for the info box
#define INFOBOX_BUTTON_OK "OK"
```

```
// credits
#define CREDIT_HEIGHT 100
#define CREDIT_WIDTH  500
#define CREDIT_TITLE  "Credits"
#define CREDIT_TEXT   "Thanks to Professor Ratzer, Francois Belair
and Sun\n"

// bugs
#define BUGS_HEIGHT 300
#define BUGS_WIDTH  600
#define BUGS_TITLE  "Bugs, FeedBack and suggestions..."
#define BUGS_TEXT   "This software is provided free of charge, but
\"as is\".\nIf you find a bug,\nplease inform Nicolas Delerue by
email.\nYou will then receive a free copy\nof this software as
soon as the bug will be fixed\nIf you have a feedback or a
suggestion\nto do about this software,\nplease also contact
Nicolas Delerue"

// Registration
#define REG_HEIGHT 400
#define REG_WIDTH  600
#define REG_TITLE  "Registration"
#define REG_TEXT   "This software is provided free of charge, but
\"as is\".\nIf you want to receive a technical support\n about it,
please register to Nicolas Delerue\nYou can distribute as many
copies as you\n want of this software, as long as you do not ask
money for it\nAll registered users receive\nby e-mail a free copy
of each new release of this software\nRegistration is also a way
of supporting and encouraging the developpement of free software\n
Registration cost is:\nCA$ 10.00 or US$ 8.00\n or 6.00 Euro
(European Currency Unit as of January 1st, 1999)"

// Notavailable
#define NA_HEIGHT 300
#define NA_WIDTH  300
#define NA_TITLE  "Sorry, Feature not Available..."
#define NA_TEXT   "This software is still under developpement\nand
the feature you have requested\nis not available at the moment.\nI
suggest you to register\n(see registration or contact Nicolas
Delerue)\nAll registered users receive by e-mail a free copy\nof
each new release of this software"

//ToolBar
#define TB_UPDIR "UPDIR"
#define TB_UPDIR_TIP "Goes to the upper directory"
#define TB_UPDIR_NUMBER 11

#define TB_DISCONNECT "Disconnect"
#define TB_DISCONNECT_TIP "Disconnect from the current host"
#define TB_DISCONNECT_NUMBER 12

#define TB_REFRESH "Refr"
#define TB_REFRESH_TIP "Refresh the current directory file list"
#define TB_REFRESH_NUMBER 13

#define TB_ABORT "Abort"
#define TB_ABORT_TIP "ABORT the last command sent"
#define TB_ABORT_NUMBER 14
```

```
#define TB_DELETE "Delete"
#define TB_DELETE_TIP "Delete the selected files"
#define TB_DELETE_NUMBER 15


#define TB_CLOSE "Close"
#define TB_CLOSE_TIP "Close this connection window"
#define TB_CLOSE_NUMBER 31

// These values are used for the spacing in the oneFile methods
#define SPACES_BEFORE_SIZE 5 //when a string is read there must be
a least this number of spaces before the size
#define SIZE_COLUMN_START 30 // For the layout of the oneFile
representation (getRepr());
#define COMPULSORY_SPACES 5

#define NOT_LOADED_SUBDIR "loading..."

#define SIZE_ELEMENT_SKIPPED_IN_LIST 15 // a fisrt element smaller
than that is skipped after a LIST command

#define EMPTY_DIRECTORY_CONTENT "." // content displayed for an
empty directory
```

8.3 Acknowledgements

I would like to thanks Professor Ratzer for letting me do such a project in his cs573 course, without this course I would never have found the time to learn the Java language, the FTP protocol and to write a software using these tools.

François Belair gave me valuable advice during this project and I would like to thank him.

I am also graceful to Sun and JavaSoft who provided a so nice programming language…