概要设计说明书

作者：王聪、周晓炜、金明洁、赵崇日、孔建军

1 引言

1.1 编写目的

确定整个程序设计框架，以及需要实现的功能，对程序接口进行定义，把类划分出来，定义部分方法以及出错处理。预期读者为相关项目开发及维护人员。

1.2 背景

说明：

a. 项目名称：XYLFTP

b. 任务提出者：王亚刚老师

开发者：见软件需求说明书 1.2.b

用户：客户端面向 Windows/Linux 普通用户。

中心：能运行 JVM 虚拟机的机器，主要面向 Linux/Windows 下的用户。

1.3 定义

FTP--File Transfer Protocol. See RFC 959.
JVM—Java Virtual Machine. See http://java.sun.com.

1.4 参考资料

a. TCP/IP Illustrated, Vol I, W. Richard Stevens
b. Java Network Programming, Elliotte Rusty Harold
c. 软件文档国家标准
d. RFC959

2 总体设计

2.1 需求规定

a.客户端接受从终端输入的命令，以及启动客户端所带参数。

xylftp [-h|--help][-V|—version][-v|--verbose][-u $USERNAME|--user=$USERNAME][-p $PASSWORD |--password=$PASSWORD][-d|--debug] [$HOST]

-u $username|—user=$username 使用$user 连接，当不使用-u 时使用默认的 anonymous 用户名。

-h|--help 显示客户端支持的所有命令。

-p $PASSWORD |--password=$PASSWORD 指定用$PASSWORD 密码登录，未指定时使用默认空密码。

-V|—version 显示版本信息。

-v|—verbose 显示更多额外信息。

-d|—debug：显示比—verbose 更多的信息，供开发者或网络管理员使用。

b.交互式命令：

help [$CMD] — 显示本客户端支持的命令列表，如果$CMD 指定，显示$CMD 的用法 。

quit — 退出本程序。

bye -- 等同于 quit。

open $HOST [$PORT] — 建立与$HOST 的 FTP 连接，如果启动此客户端时没有指定-u 或-p 的话。如果$PORT 指定，则连接$HOST 的$PORT 端口，否则就尝试默认的 21 号端口。

user $USERNAME [$PASSWORD] — 指定用$USERNAME 用户名进行连接，如果$PASSWORD 指定，尝试用指定密码，否则尝试默认的空密码。

passive — 指定使用被动模式。

pwd — 列出服务器端的当前路径。

cwd [$DIR] — 进入服务器$DIR 目录，如果$DIR 没有指定，则不做任何处理。

cd [$DIR] — 等同于 cwd。

cdup — 等同于 cwd ..。

mkdir $DIR — 在当前目录下新建一个名$DIR 的目录。

rmdir $DIR — 删除当前目录下的$DIR 目录。注：目录必须为空。

dir [$DIR| $FILE] — 列出服务器$DIR 目录下的文件或者$FILE 的相关信息，如果$DIR 和$FILE 没有指定，则列出当前目录下的文件。

ls [$DIR| $FILE] — 与 dir 等同。

delete $FILE — 删除一个文件。

size $FILE — 获取$FILE 文件的大小。

rename $OLDNAME $NEWNAME — 把名为$OLDNAME 文件重新命名为$NEWNAME。

chmod $MODE $FILE — 更改$FILE 的权限为$MODE。

get $FILE [$NEWFILENAME] -- 从服务器获取名为$FILE 的文件，保存至本地当前路径。

如果$NEWFILENAME 指定，保存的文件应命名为$NEWFILENAME，否则和$FILE 同名。

put $FILE [$NEWFILENAME] -- 将本地的$FILE 文件传送至服务器端的当前路径。如果

$NEWFILENAME 指定，传上去的文件应命名为$NEWFILENAME，否则和$FILE 同名。

close — 断开当前连接，并返回交互式界面。

!! -- 执行本地 shell。

? [$CMD] — 等同于 help。

c. 输出：通过终端输入命令，经检测如是本地命令则调用相应方法处理，并将结果直接显示到屏幕。若是交互式的命令，则需将命令发送至服务端并且将返回的相应信息输出。

详细说明可参见本项目需求分析。

2.2 运行环境

客户端适用于所有运行 JVM 的机器上。

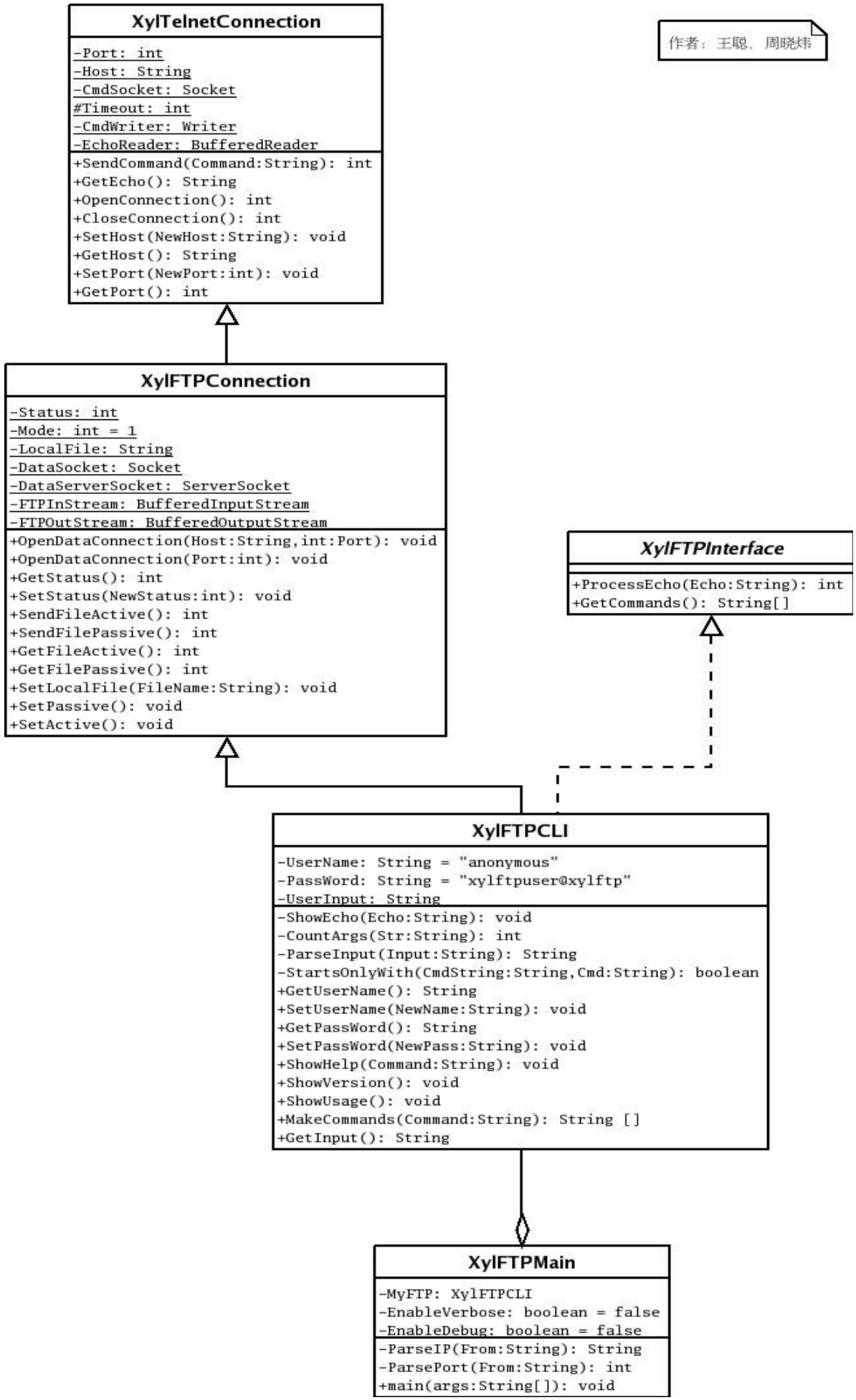
2.3 基本设计概念和处理流程

设计一个简单、实用的 FTP 客户端，支持常用的 FTP 命令，使用 Java 标准库开发，并且预留 GUI 接口。客户端首先接收命令行的输入，使用命令解释程序来解析命令，如果本地可以完成的命令，直接调用相应的方法完成，如果需要传送到远程 FTP 服务器，则传输相应的命令到远程 FTP 服务器。同时客户端不断

接收服务端的处理结果，并显示或存储相关数据。

2.4 结构

见类图


XylTelnetConnection
-Port: int
-Host: String
-CmdSocket: Socket
#Timeout: int
-CmdWriter: Writer
-EchoReader: BufferedReader
+SendCommand(Command:String): int
+GetEcho(): String
+OpenConnection(): int
+CloseConnection(): int
+SetHost(NewHost:String): void
+GetHost(): String
+SetPort(NewPort:int): void
+GetPort(): int
作者：王聪，周晓炜
XylFTPConnection
-Status: int
-Mode: int = 1
-LocalFile: String
-DataSocket: Socket
-DataServerSocket: ServerSocket
-FTPInStream: BufferedInputStream
-FTPOutStream: BufferedOutputStream
+OpenDataConnection(Host:String,int:Port): void
+OpenDataConnection(Port:int): void
+GetStatus(): int
+SetStatus(NewStatus:int): void
+SendFileActive(): int
+SendFilePassive(): int
+GetFileActive(): int
+GetFilePassive(): int
+SetLocalFile(FileName:String): void
+SetPassive(): void
+SetActive(): void
XylFTPInterface
+ProcessEcho(Echo:String): int
+GetCommands(): String[]
XylFTPCLI
-UserName: String = "anonymous"
-PassWord: String = "xylftpuser@xylftp"
-UserInput: String
-ShowEcho(Echo:String): void
-CountArgs(Str:String): int
-ParseInput(Input:String): String
-StartsOnlyWith(CmdString:String,Cmd:String): boolean
+GetUserName(): String
+SetUserName(NewName:String): void
+GetPassWord(): String
+SetPassWord(NewPass:String): void
+ShowHelp(Command:String): void
+ShowVersion(): void
+ShowUsage(): void
+MakeCommands(Command:String): String []
+GetInput(): String
XylFTPMain
-MyFTP: XylFTPCLI
-EnableVerbose: boolean = false
-EnableDebug: boolean = false
-ParseIP(From:String): String
-ParsePort(From:String): int
+main(args:String[]): void

2.5 人工处理过程

处理过程无需人工干预。

2.6 尚未问决的问题

类的详细设计，功能的进一步完善，这些工作留到详细设计阶段。

3 接口设计

3.1 用户接口

客户端提供两种用户接口，命令行参数和交互式输入命令，如 help，quit，bye 等，详见需求说明书。

3.2 外部接口

客户端需要 JVM 的支持，人机交互模式可通过命令行界面或 GUI 实现。

3.3 内部接口

1. Xy1FTPMain：主方法类

2. Xy1FTPCLI：命令行实现类

3. Xy1TelnetConnection：远程控制类

4. Xy1FTPconnection：连接类

5. Xy1FTPinterface：接口（可以实现为图形用户接口）

详细的类方法见 2.4 节类图。

4 运行设计

4.1 运行模块组合

a. 概述

首先由 Xy1FTPMain 类中的 main 方法开始，通过 GetInput 得到用户键入的命令，然后通过 ParseInput 解析，并判断该命令是否为本地命令，若是，则调用 ShowEcho 方法显示信息，若是服务器命令则根据调用 GetStatus 方法得到成员 Status 的值（0 为未连接，1 为连接上但未登录，2 为登录上，3 为连接上且正下载，4 为连接上且正上传）。当用户键入 user $USERNAME $PASS 后，通过 SendCommand 将建立连接的请求发送至服务端判断，并由服务端返回信息并选择建立连接。

建立连接并 SetStatus 后，用户输入的命令将在 Xy1FTPCLI 的方法 ParseInput 检测合法后，被 ParseInput 解析并通过 SendCommand 传输到服务端，客户端进入交互模式进行命令操作，服务端根据 Status 确定实施的操作。若是数据连接则调用 FTPInStream 和 FTPOutStream 读写磁盘（包括 dir 命令）并返回相应的回应信息。

要断开连接时，输入的 quit 或 bye 命令通过解析后传送给服务端，GetEcho 得到服务端的回应，并由 ProcessEcho 调用 ShowEcho 显示到屏幕。此时关闭 Socket 连接。

抽象接口 Xy1FTPInterface，现在仅用命令行界面实现，今后扩展为图形界面。

b. 关于异常处理

异常处理应该单独划分一个类来处理。

c. 命令执行流程示例

open $HOST [$PORT]

GetCommands 调用 GetInput 得到这个字符串后会把它传递给 ParseInput 分析，ParseInput 先判断当前的状态，如果是已连接（Status>0），则抛除异常。否则，则调用 MyFTP.OpenConnection($HOST)与 $HOST 上的$PORT 端口（如果$PORT 未指定，则使用 21 号端口）连接并用 SetStatus 把 Status 设为 1。

close

ParseInput 解析出是 close 以后，根据 Status 进行下一步处理，如果 Status 为 0 时，抛出异常。Status 为 1 时，调用 SendCommand 发送 QUIT 向服务端，调用 GetEcho 接受返回响应以后，调用 MyFTP.CloseConnection 关闭当前连接，并通过 SetStatus 设置 Status 为 0。如果 Status 为 3 或 4 时，则先调用 SendCommand 发送 ABOR 到服务器端，调用 GetEcho 接受返回响应后关闭数据连接，再发送 QUIT 命令关闭控制连接。

quit

ParseInput 解析出是 quit 以后，根据 Status 进行下一步处理，如果 Status 为 0 时，直接退出 main 方法退出程序。 Status 为 1 或 2 时，调用 SendCommand 发送 QUIT 向服务端，调用 GetEcho 接受返回响应以后，调用 MyFTP.CloseConnection 关闭当前连接，并退出程序。如果 Status 为 3 或 4 时，则调用 SendCommand 发送 ABOR 到服务器端，调用 GetEcho 接受返回响应，再关闭控制连接和数据连接，并退出

程序。

bye

与 quit 处理方法相同。

user $USERNAME [$PASSWORD]

GetInput 把它传递给 ParseInput 之后，ParseInput 根据 Status 的值进行判断，如果 Status 为 0，则把 Xy1FTPMain 类的成员 UserName 和 PassWord 设为指定的值。如果 Status 为 1 或 2，则把它翻译成 USER $USAERNAME 调用 SendCommand 发送至服务器，GetEcho 得到回应后再调用 SendCommand 发送 PASS $PASSWORD，如果$PASSWORD 被指定。如果 Status 的值大于 2，则抛出异常。

pwd

GetCommands 中的 ParseInput 直接解析为 PWD 调用 SendCommand 发送至服务器，由 GetEcho 接到回应后传递给 ProcessEcho 调用 ShowEcho 显示在屏幕上。

cwd [$DIR]

ParseInput 判断出是 cwd 后查看后面的$DIR 有没有指定，如没有，直接返回；如有，则把它翻译成 CWD $DIR 调用 SendCommand 发送至服务器，并由 GetEcho 接到回应后传递给 ProcessEcho 调用 ShowEcho 显示在屏幕上。

cd [$DIR]

等同于 cwd。

cdup

ParseInput 在接到 cdup 后，先查看该命令格式是否正确，是否满足发送它的条件，如不正确或不满足，则抛除异常。如都可以，则发送 CDUP 命令，最后把得到的响应显示到屏幕上。

mkdir $DIR

ParseInput 解析出是 mkdir 命令以后，先做一些检查，看是否满足发送 MKD 的条件，若满足把它

翻译成 MKD $DIR 之后发送到服务器端，把得到的响应显示在屏幕上。如不满足，则抛除异常。

rmdir $DIR

首先，ParseInput 解析出是 rmdir 命令以后，先做一些检查，看是否满足发送 RMD 的条件，若满足把它翻译为 RMD $DIR 之后发送到服务器端，把得到的响应显示在屏幕上。如不满足，则抛除异常。

dir [$DIR| $FILE]

ParseInput 将其解析，判断出是 dir 后查看后面的$DIR 或$FILE 有没有指定,如没有,直接返回。如有，则把他翻译成 DIR $DIR，调用 SendCommand 发送至服务端，服务端将指定端口进行连接。

客户端则通过调用之后 GetFile 将数据取回， 通过流 System.out 输出到屏幕上。

ls [$DIR| $FILE]

等同于 dir。

size $FILE

ParseInput 解析到是 size 后，先检查当前的 status 和输入的命令的格式，如格式错误或不满足发送 SIZE 的条件，则抛除异常。如满足，就想服务器发送 SIZE $FILE 命令，然后把得到的响应显示到屏幕上。

delete $FILE

ParseInput 解析到是 delete 后，先检查当前的 status 和输入的命令的格式，如格式错误或不满足发送 DELE 的条件，则抛除异常。如满足，就向服务器发送 DELE $FILE 命令，然后把得到的响应显示到屏幕上，无论 delete 是否执行成功。

chmod $MODE $FILE

ParseInput 解析到是 chmod 后，先检查当前的 status 和输入的命令的格式，如格式错误或不满足发送 SITE CHMOD 的条件，则抛除异常。如满足，就向服务器发送 SITE CHMOD $MODE $FILE 命令，然后把

得到的响应显示到屏幕上，无论 chmod 是否执行成功。

Passive

调用 SetPassive()方法来设定被动模式。

get $FILE [$NEWFILENAME]

首先通过 ParseInput 将其解析，经判断是 get 将其翻译成 RETR $FILE 后，查看后面是否有加 $NEWFILENAME，若有，则把 LocalFile 和指定文件关联，否则和$FILE 关联。调用 GetStatus 判断当前状态，得到 Status 的值若为 0 或 1，则抛除异常，若值为 2，则用 SendCommand 发送上面解析出的的命令，然后调用 GetFile 方法开始建立数据连接，并将服务端发过来的数据存入相应的文件 LocalFile。若 Status 为 3 或 4 则此时 GetFile 将出错，此时显示出错，抛出异常。若传输过程中网络连接断开，经过等待后若还无响应则抛出异常，提示网络连接异常。

put $FILE [$NEWFILENAME]

首先通过 ParseInput 解析该命令，判断是 put 后将其翻译成 STOR 马上调用 GetStatus，获得 Status 的值，若为 0 或 1，3，4 则调用抛除异常，提示未连接 ，若为 2 时，则调用 SendCommand 传送给服务端得到响应后调用 SendFile 方法和服务器建立 DataSocket 连接，并将文件上传到相应的文件中。若 put 后面$NEWFILENAME 则将其也发送 STOR $NEWFILENAME，否则发送 STOR $FILE。当服务端检测到结束符时，得知文件上传完毕。服务端发送一个信息，客户端通过调用 GetEcho 方法，并用 ShowEcho 显示出来。若中途网络连接断开，经过等待后若还无响应则抛出异常，提示网络连接异常。

rename $OLDNAME $NEWNAME

ParseInput 判断出 rename 命令后，先查看后边的$OLDNAME 和$NEWNAME 有没有指定。 如果没有指定，则抛除异常，然后再查看当前的状态，如不满足发送 RNFR 和 RNTO 命令的条件，则抛除异常。若上述条件都满足，则发送 RNFR $OLDNAME，如这一步错误，停止操作，如正确，继续发送 RNTO $NEWNAME。

help $COMMAND

ParseInput 判断出 help 命令后，先查看后边的$COMMAND 有没有指定。 如果没有指定，则在屏幕上打印全部命令的帮助信息，否则只打印给定命令的相关信息。

!!

ParseInput 判断出!!命令后，直接进入 shell 命令模式，直到敲入 exit 命令退出 shell 命令模式。exit 后程序继续执行。

? $COMMAND

与 help 处理方法相同。

4.2 运行控制

启动时通过命令行传递参数，进入交互模式后通过键入命令进行相关控制。参见 2.1 节。

4.3 运行时间

未知，由用户具体情况而定。

5 系统数据结构设计

5.1 逻辑结构设计要点

给出本系统内所使用的每个数据结构的名称、标识符以及它们之中每个数据项、记录、文卷和系的标识、定义、长度及它们之间的层次的或表格的相互关系。

5.2 物理结构设计要点

与服务端相互传送的数据，客户端直接使用系统调用方法存取数据，不考虑底层存取的物理关系。

5.3 数据结构与程序的关系

由于客户端采用 Java 语言编写，数据结构已封装在各个类中。故不用考虑数据结构与程序之间的关系。

6 系统出错处理设计

6.1 出错信息

客户端:

01–无法连接远程主机，请确认远程主机是否活动，请确认远程主机是否使用默认端口 21。

02–上传过程中出错，请确认网络处于连接状态，并重新上传。

03–下载过程中出错，请确认网络处于连接状态，并重新下载。

04–远程主机关闭，请与远程 FTP 管理员联系。

05–未知命令。

06–命令格式不正确。

07–权限不足出错，请确认使用的用户是否具有相应的读写权限。

6.2 补救措施

上传、下载过程中出错（包括文件传输中断，以及文件校验不符），则重新上传、下载，最多重试 3 次。3 次无效后，自动断开连接，释放系统资源，并向用户作出提示。

6.3 系统维护设计

为了系统维护的方便，在程序内部设计中做出合理的安排，具体包括在程序中，专门用于系统检查

与维护的检测点和专用模块。