# Java作业-1-设计文档

| **版本** | **时间** | **责任人** | **备注** |
| --- | --- | --- | --- |
| 0.1V1 | 2021/08/01 | 周志民 | 创建文档 |

## 作业背景:

基于 Java 实现 Socket Server 和 Socket Client，以完成文件上传、文件下载、列举文件列表等功能。

**1. client 端向用户提供三条指令：**

1. put file {file\_path}：上传本地文件
2. get file {file\_name} {file\_path} ：下载文件到本地指定路径
3. list files：列举已上传的文件列表

**2. server 端提供三个接口：**

a) 上传文件接收

b) 文件下载

c) 列举文件列表，要求上传成功后，能在服务端文件系统中看到已上传的文件。

## 基本要求

1. 实现 Client 端，注意处理指令不存在、路径不存在、文件不存在等异常情况的处理。
2. 实现 Server 端，注意通信协议报文的定义、文件完整性校验等。
3. 支持列举已上传文件的文件名、文件大小、上传者 IP、上传者端口等详细信息。
4. 支持小文件（1KB）操作。
5. 给出设计档、 测试用例。

## 术语:

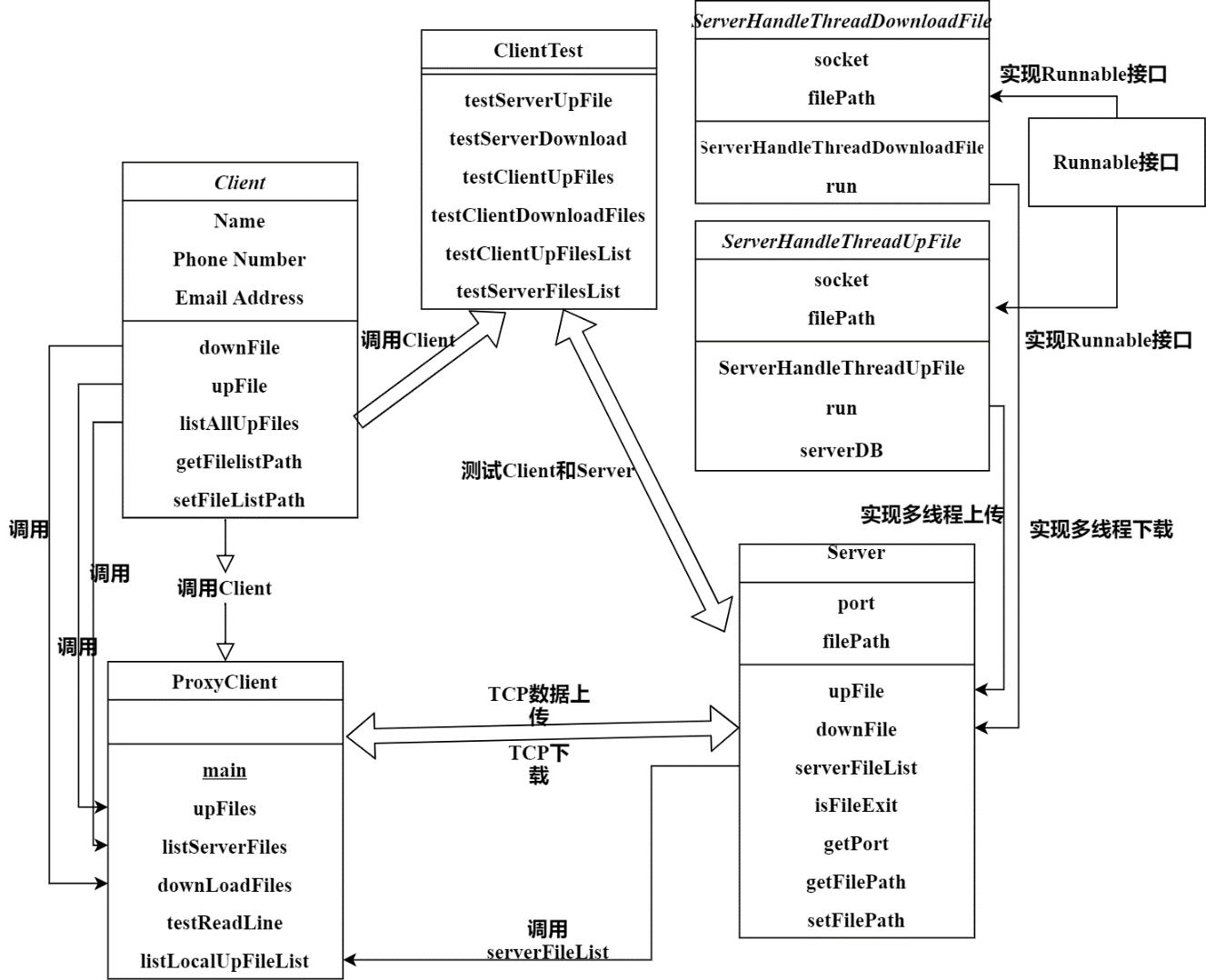
Client 端：客户端，Server 端： 服务端

## 需求分析:

* client 端实现三个命令，上传、下载和列出已上传的文件
* Server 端实现三个命令：列出本地已有文件，提供Client端上传和下载服务

## 方案设计

因为使用的是java.net.Socket，java.net.ServerSocket，采用TCP方式进行文件的传输，因为TCP是可靠文件传输，文件传输不会出现错乱和丢失现象。这样基本要求的第二条：“注意通信协议报文的定义、文件完整性校验。”是自动完成的，所以下面的文件完整性只简单的使用了接收和发送的大小是否一样进行校验，未对通信协议报文的定义进行手动修改。完成作业的总共写了四个类，Client类和Server类通过TCP进行文件传输，ClientTest 类和ProxyClient类对相应的文件传输进行校验。



* Client类 实现文件的上传和下载，
* Server类 实现文件接收和提供Client 上传和下载服务，
* ClientTest 类 使用Junit 单元测试模块对 上传下载和查询进行测试
* ProxyClient类使用Client类对上传和下载和查询在终端进行交互式测试

#### Server 端

|  |  |
| --- | --- |
| Server 类 | 功能 |
| private final int port = 9099; | 端口号 |
| private String filePath = "E:\\server\_files\\"; | //Server 默认存储文件所在的地方 |
| public void upFile() | //为客户端提供上传服务，对于对文档校验，采用接收到和发送的文件大小是否一样进行判断，因为不能使用其他的API，暂时未找到更好的方法。 |
| public void downFile() | //为客户端提供下载服务，文档校验同上。 |
| public boolean isFileExit(String fileName) | //接收下载请求时，判断一下是否相关的文件是否存在 |
| public void serverDB(…) | // 接收文件时记录上传者和文件的信息 |
| public void serverFileList() | //列出服务器端已上传的文件和上传者信息 |

#### Client 端

|  |  |
| --- | --- |
| Client 类 |  |
| private final String ip = "127.0.0.1"; | //使用本机IP地址 |
| private final int port = 9099; | // 设置远程端口号 |
| private String fileListPath = "E:\\client\_files\\upClientFiles.db"; | //已上传文件名保存的文件地址 |
| public void addListFiles(String filename) | //当上传文件时把已上传的文件记录到本地的upClientFiles.db文件中。 |
| public void listAllUpFiles() | //列出所有已上传的文件 |
| public void upFile(String filePath) | //向服务端上传文件，对于对文档校验，采用接收到和发送的文件大小是否一样进行判断，因为不能使用其他的API，暂时未找到更好的校验方法。 |
| public void downFile(String filePath, String filename) | //从服务端下载文件，文档校验同上。 |

#### ClientTest 类

|  |  |
| --- | --- |
| 使用Junit 单元测试模块对 上传下载和查询进行测试 | 功能测试 |
| @Test public void testServerUpFile() | *// 启动服务器接收客户端发送的文件，需手动终止运行* |
| @Test public void testServerDownload() | *// 启动服务器处理客户端下载请求，需手动终止运行* |
| @Test public void testClientUpFiles() | *// 测试客户端上传文件* |
| @Test public void testClientUpFilesList() | *// 列出客户端已经上传的文件列表* |
| @Test public void testServerFilesList() | *// 测试 输出 服务器端现有的文件列表* |
| @Test public void testClientDownloadFiles() | *// 测试 从服务器端 下载文件到客户端不同目录* |

#### ProxyClient类，在终端提供交互式上传和下载

对上传和下载和查询进行终端交互式测试

|  |  |
| --- | --- |
| ProxyClient类 | 功能 |
| public static void main(String[] args) | //采用输入0,1,2,3,4 选择查询，上传，下载和退出功能 |
| public void upFiles(Client client) | //从客户端上传文件，检查是否为文件绝对路径，是否真实存在 |
| public void listServerFiles() | //列出服务器端已上传的文件和上传者信息 |
| public void downLoadFiles(Client client) | //从客户端下载文件，检查终端输入的路径是否存在，输入文件名远程服务器上是否存在 |
| public String testReadLine() | //用于读取终端操作 |
| public void listLocalFileList(Client client) | //列出本地已上传文件名 |

## 数据存储

* 服务端已有文件列表，存储在"E:\\server\_files\\Server.db"文件中，使用Java IO流进行读写。
* 客户端已上传文件列表，存储在"E:\\client\_files\\

upClientFiles.db" 文件中，使用Java IO流进行读写。

## 交互流程

|  |
| --- |
| 1. 在终端输入不同数字，代表不同的功能：   请输入数字：  0: 上传文件  1: 下载文件  2: 列出服务器已有文件  3: 列出本地已上传文件  4: 退出   1. 在终端输入 0：   "请输入文件绝对路径,包含文件名，例如：D:/1.txt"  会判断输入是否合乎规定，输入错误会要求重新输入   1. 在终端输入 1：   "请输入下载文件保存的路径："  会判断输入是否是真实存在的路径，输入错误会要求重新输入  "请输入下载文件名称："  会判断输入文件名是否是真实存在的远程，输入错误会要求重新输入   1. 在终端输入 2：   "文件名" + " \t" + "上传者 IP" + " \t" + "上传者端口号" + " \t" + "文件大小"  会判断远程是否有文件，没有提示上传，有列出文件列表   1. 在终端输入3：   "本地已上传文件"  会判断是否已上传文件，没有提示上传，有列出已上传文件名   1. 在终端输入 4：   “退出交互式测试程序” |

## 问题疑惑

“支持小文件（1KB）操作”，没有找到小文件与大文件的区别，使用字节流进行读写好像与文件大小没啥关系。还是因为使用了Java.net的相应模块，屏蔽掉了这个问题。但是更底层方法暂时还不了解。