Objetivo do projeto é realizar o upload de um arquivo do cliente (Client) para o servidor (Server), e também fazer o download de um arquivo do servidor para o cliente.

Programa composto por três classes:

Server;

FileShered;

Client;

**Como rodar e como funciona o programa:**

1. Rode (Run) a classe ‘Server’. Ela é responsável por armazenar os arquivos enviados.
2. Informar a pasta do servidor que armazenará os arquivos
3. Rode (Run) a classe ‘Client’. Ela é responsável por solicitar o envio, listagem e download de arquivos do servidor.
4. Realizar o upload de um arquivo ‘up’ para o servidor (caso não exista arquivos na pasta).
5. Após, só entrar com os comandos list, down cod, up ou sair.

**Comentários sobre as classes e Métodos**

***Server***

**public** **void** informarServidor()

Solicita ao cliente a pasta do servidor. Essa pasta será nosso servidor e receberá os arquivos.

**public** **void** iniciarConexao()

Inicia a conexão do servidor com o cliente. ServerSocket, Socket, BufferedReader e PrintStream são iniciados nela para leitura dos dados.

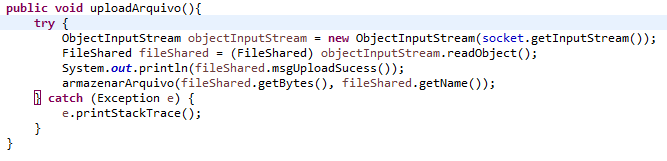
**public** **void** encerrarConexao()

Encerra a conexão do servidor com o cliente. ServerSocket, Socket, BufferedReader e PrintStream são fechados nela.

**public** **void** atualizarServidor()

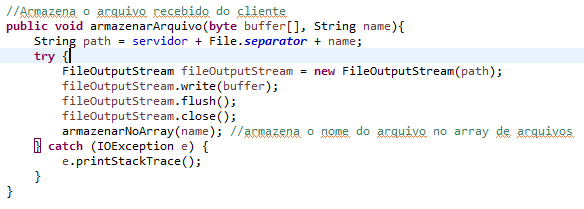
Verifica se existem arquivos armazenados na pasta do servidor. Se sim, adiciona ao array de arquivos. Dessa forma, sempre que o programa é iniciado, o array de arquivos é atualizado para que a listagem fique correta.

**public** **void** uploadArquivo ()

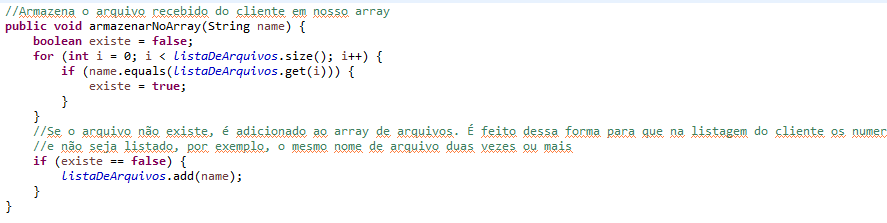


Recebe os dados do arquivo serializado enviados pelo cliente (upload). Uma instância da classe FileShared é criada e lê os dados serializados.

Após, chama a função armazenarArquivo e seta os bytes do arquivo e seu nome.

**public** **void** armazenarArquivo(**byte** buffer[], String name)

Recebe os dados do arquivo contidos no método anterior. A String path recebe o servidor e o nome do arquivo para que o mesmo, agora completo, seja escrito. Após, um objeto do tipo FileOutputStream é criado e recebe a varíavel path. O objeto então escreve os dados do arquivo que é armazenado na pasta do servidor.

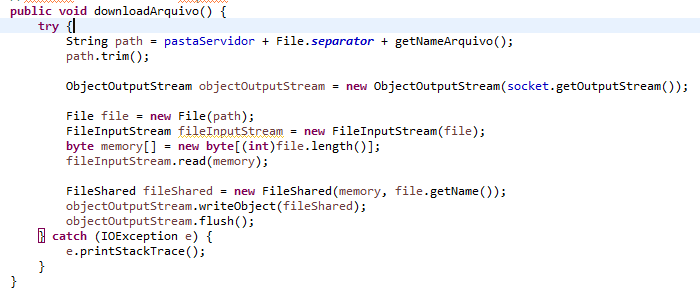
**public** **void** armazenarNoArray(String name) 

Verifica se o nome do arquivo não existe na pasta do servidor. Se não, adiciona ao array. Assim, os numeradores ficam corretos na listagem para o cliente.

**public** **void** listarArquivosArmazenados()

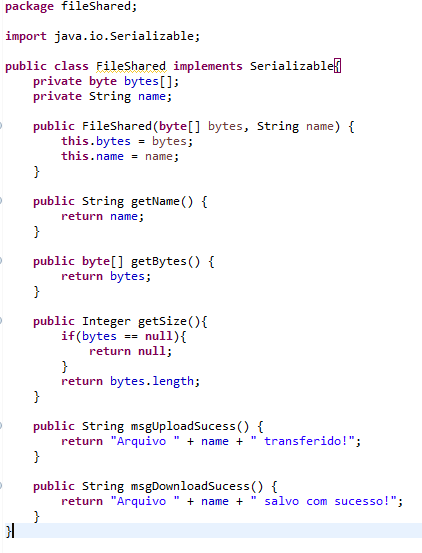
Lista os arquivos armazenados na pasta do servidor.

**public** **void** downloadArquivo()



Faz o download do arquivo para o cliente.

***FileShared***



Contém os métodos getters e setters para recebimento dos dados do arquivo. Funciona como a ponte entre cliente e servidor através do Serializable, que salva os dados enviados pelo cliente e podem ser recuperados pelo servidor e do contrário também.

***Client***

**public** **void** iniciarConexao()

Inicia a conexão do cliente com o servidor. Socket, BufferedReader e PrintStream são iniciados nela para leitura dos dados.

**public** **void** encerrarConexao()

Encerra a conexão do cliente com o servidor. Socket, BufferedReader e PrintStream são fechados nela.

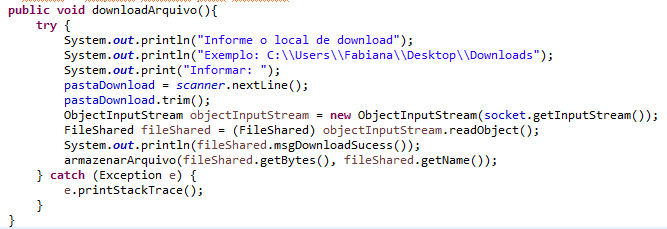
**public** **void** uploadArquivo()

Método que envia um arquivo para o servidor (upload).

É solicitado ao cliente o local + caminho do arquivo. Estes dados então são salvos em um arquivo File e setados em um arquivo do tipo FileInputStrem. Após, os dados referentes ao comprimento do arquivo e dados do arquivo são salvos em uma variável do tipo byte e lidos pelo arquivo do tipo FileInputStream.

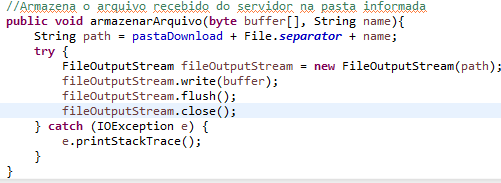
Uma instância da classe FileShared é criada e recebe os dados do arquivo e seu nome. Estes dados agora são escritos por um objeto do tipo ObjectOutputStrem na instância da classe FileShared. Agora, os dados poderão ser pegos pelo servidor, pois estão serializados.

**public** **void** downloadArquivo()



Recebe os dados do arquivo serializado enviados pelo servidor (download). Uma instância da classe FileShared é criada e lê os dados serializados.

Após, chama a função armazenarArquivo() e seta os bytes do arquivo e seu nome.

**public** **void** armazenarArquivo(**byte** buffer[], String name)

Recebe os dados do arquivo contidos no método anterior. A String path recebe a pasta de download e o nome do arquivo para que o mesmo, agora completo, seja escrito. Após, um objeto do tipo FileOutputStream é criado e recebe a varíavel path. O objeto então escreve os dados do arquivo que é armazenado na pasta de download.