1. FTPClient: This is the main class that represents the FTP client. It has properties for the FTP server's host, username, and password. It uses the FileTransfer class to transfer files and the DirectoryTransfer class to transfer directories.
2. FileTransfer: This class represents a file transfer operation between the local machine and the remote FTP server. It has properties for the local file path, remote file path, username, and password.
3. DirectoryTransfer: This class represents a directory transfer operation between the local machine and the remote FTP server. It has properties for the local directory path, remote directory path, username, and password. It also uses the Archive class to archive the directory and the ArchiveCompressionCodec class to compress the archive.
4. Archive: This class represents an archive file. It has properties for the archive file path and the archive file type.
5. ArchiveCompressionCodec: This class represents a compression codec for archive files. It has a property for the compression codec type.

The above class model covers all the functional requirements mentioned earlier:

1. File Transfer: The FileTransfer class handles file transfer between the local machine and the remote FTP server.
2. Directory Transfer: The DirectoryTransfer class handles directory transfer between the local machine and the remote FTP server.
3. Archiving and Compression: The DirectoryTransfer class uses the Archive class to create an archive file for the directory to be transferred, and the ArchiveCompressionCodec class to compress the archive.
4. Authentication and Security: The FTPClient, FileTransfer, and DirectoryTransfer classes have properties for the FTP server's host, username, and password, which are used for authentication and security.
5. Error Handling and Recovery: Error handling and recovery can be implemented at the class level by handling exceptions and errors thrown by the underlying FTP library or the file system.
6. Logging and Reporting: Logging and reporting can be implemented at the class level by logging events and errors to a file or the console.

FTPClient

Member Variables:

* **host**: A string representing the host name or IP address of the FTP server.
* **username**: A string representing the username for the FTP server.
* **password**: A string representing the password for the FTP server.
* **current\_directory**: A string representing the current working directory on the FTP server.

Member Functions:

* **FTPClient()**: A constructor that takes the **host**, **username**, and **password** parameters as arguments and initializes the member variables.
* **~FTPClient()**: A destructor that cleans up any resources used by the **FTPClient** object.
* **connect()**: A function that establishes a connection to the FTP server.
* **disconnect()**: A function that terminates the connection to the FTP server.
* **is\_connected()**: A function that returns a boolean value indicating whether a connection to the FTP server has been established.
* **list\_directory()**: A function that returns a vector of strings representing the files and directories in the current working directory on the FTP server.
* **change\_directory()**: A function that changes the current working directory on the FTP server to the specified path.
* **get\_current\_directory()**: A function that returns the current working directory on the FTP server.
* **upload\_file()**: A function that uploads a file from the local system to the FTP server.
* **download\_file()**: A function that downloads a file from the FTP server to the local system.
* **archive\_directory()**: A function that archives a directory on the FTP server.
* **compress\_file()**: A function that compresses a file on the FTP server.
* **authenticate()**: A function that authenticates the user credentials with the FTP server.
* **is\_authenticated()**: A function that returns a boolean value indicating whether the user is authenticated with the FTP server.
* **get\_error\_message()**: A function that returns a string representing the last error message that occurred.
* **log()**: A function that logs a message to a file or console.

FileTransfer

In this example, the **FileTransfer** class contains the following member variables:

* **hostname**: a string representing the FTP server hostname
* **port**: an integer representing the port number to use for the FTP connection
* **username**: a string representing the FTP server username
* **password**: a string representing the FTP server password
* **remoteDir**: a string representing the remote directory path on the FTP server
* **localDir**: a string representing the local directory path on the client machine

The class also contains a constructor that initializes the member variables with the given parameters.

The class has four member functions:

* **connect()**: a function that establishes a connection to the FTP server using the specified hostname, port, username, and password.
* **disconnect()**: a function that terminates the connection to the FTP server.
* **upload(std::string filename)**: a function that uploads a file with the given filename to the remote directory on the FTP server.
* **download(std::string filename)**: a function that downloads a file with the given filename from the remote directory on the FTP server to the local directory on the client machine.

These functions return a boolean value indicating whether the operation was successful or not.

This header file declares the **FileTransfer** class, which has four public member functions: **connect**, **disconnect**, **uploadFile**, and **downloadFile**. It also has a default constructor, a constructor that takes four parameters (server address, port number, username, and password), and a destructor.

The private members of the **FileTransfer** class include the server address, port number, username, and password. There may also be additional private members for the implementation, such as a socket or file transfer protocol specific data structures or objects.

DirectoryTransfer

In this example, the **DirectoryTransfer** class has two private member variables: **localDirectory** and **remoteDirectory**, which represent the local and remote directories, respectively. The class also has two public member functions: **sendDirectory** and **receiveDirectory**, which take an **FTPClient** object as an argument.

The **sendDirectory** function would be responsible for transferring a local directory to the remote server via FTP. The **receiveDirectory** function, on the other hand, would be responsible for downloading a remote directory to the local machine. Both functions would return a boolean value indicating whether the transfer was successful or not.

This header file defines the **DirectoryTransfer** class and declares its constructor and two public member functions: **sendDirectory()** and **receiveDirectory()**.

Archive

ArchiveCompressionCodec