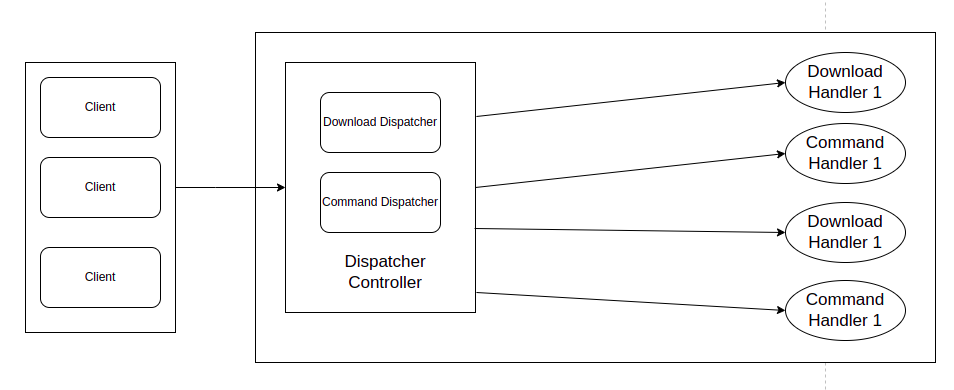
**Architecture File Server**



- In our application, It get one Dispatcher Controller to manage all requests from the client. There are 2 kinds of requests such as: Command and Data therefore the application needs to use 2 ports. One is for Command and one for Data.  
 + Command can be: client just requests to list out all files under folder or request to get size of one file.

+ Data: The request asks for file downloading.

**User Case:** user type commands like: get file1.txt, file2.txt

Step 1: Server starts listening on 2 ports. One is for the Command line and One is for downloading files.

Step 2: Client creates a new connection with the server on a particular port number.

Step 3: Request will come into server and go to Command Dispatcher. It will accept a connection. It also creates a new thread such as CommandDispatcherHandler to handle a new connection. This CommandDispatcherHandler only takes responsibility for “Command Request” between the server and this client.

Step 4: Client will send 2 “Get Command” such as get file1.txt and get file2.txt to the server.

Step 5: After the server receives “Get file1.txt” then the server will check if a file exists or not. There are 2 files so the client will create 2 new threads (2 new connections) for file downloading.

+ File does not exist then server will not do anything. The server just sends information to the client. Client received the message and also didn't do anything.

+ File exists. The server will send information to the client. Client will create a new thread in which creates a new connection to the server. A new connection will take responsibility for file downloading until finishing file downloading.

+ After finishing all file downloading. All threads will be destroyed to release memory.

**Tech stack**

**-** Jdk 11, Junit Test and Mockito

**-** Lombok

- Gradle 7.4.2

**What can I improve for the next version?**

- Each downloading should have one process id so it gets the interruption for file downloading then it can be easy to resume downloading after the network is ok. Don't need to download all over again.

- Use a Ring Buffer like LMAX-Disruptor (library from Guava) to manage download threading. It is better for managing concurrency.

- I now only allow one connection per one file. Next version, we should allow to configure multiple connection to download one file and using RandomAccessFile (download one file from multiple connection and using RandomAccessFile to write file).

- Refactor source code. Organise subclass and superclass better and Update to Swing GUI

- Allow the user to change the server folder on GUI to view files on the server. Now, I configure it in build.gradle file

- Allow the user to change the client folder on GUI to store downloaded files. Now, I configure it in build.gradle file