# Peer to Peer File Sharing

This program is composed of a server and a client. The server is a CLI application that connect to a database that keeps a list of host, port and file that is currently being shared.

The client is a gui application which connects through CORBA communicates with the server to update the database of the file it is currently sharing. The client would also communicate through other clients to check if a file is available to be downloaded and to download a file from it.

Follow the steps below to run the program.

#### Notes:

1. This program is tested on a Debian computer
2. This program is compiled on Java 8
3. Uses sqlite for database
4. This program uses Intellij Swing Designer plugin to create the gui. If there are compiling errors please use intellij with the plugin https://www.jetbrains.com/help/idea/design-gui-using-swing.html
5. Directory structure is as follows:
   1. /out/production/Assignment2 - contains the compiled class. A sqlite database and the essential library to run the program
   2. /src - source code for the program
   3. /lib - libraries used by the program

### Server instructions

1. First ensure that corba is installed on the computer
2. Run the corba server with the following command
   1. orbd -ORBInitialPort 1050 -ORBInitialHost localhost&
3. Ensure that the identifier.sqlite database is on the same directory as the Main class file
4. Run the following command to run the server application
   1. java Main server -ORBInitialPort 1050 -ORBInitialHost localhost
5. The program should say it is ready and waiting

### Client instructions

1. The client command accepts the same ORB arguments and also arguments to the directory where files are shared and where it will be downloaded and the port number to run from. The syntax to run the client program is as follows > java Main client -ORBInitialPort {orb port} -ORBInitialHost {orb host} {directory} {port} Example:  
   > java Main client -ORBInitialPort 1050 -ORBInitialHost localhost /home/user/FileShare 2222
2. On load of client the application will look into the directory provided in the argument (the share directory) and will attempt to contact the server through corba and add the files to the database. The client will also launch a server socket that listens on the provided port
3. The gui is made up of:
   1. Sync button - clicking this button will update the filesharing server of the contents of the share directory which allows other clients to download it
   2. Available files - this list all files that can be downloaded. This will not list the files that are being shared by the client itself
   3. Fetch files - this will update the available files list from the file sharing server
   4. Download - this will download the files selected in the available files list
   5. Text area - this shows messages from the client
4. Once the client is loaded. You can click on “Fetch files” button to update the list of “Available files”
5. You can also click on “Sync” to allow other clients access to the files in your share directory
6. Select the files you want to upload from the list then click “Download” button
7. The client will then first check if the file is available by creating a socket connection and sending the string “check” followed by the file name.
8. If the file is available it will then create a socket connection then send the string “get” followed by the file name.
9. It will then check if a folder called “downloads” is in the share directory if there is none it will create it.
10. It will then create the file and save the file inside the downloads folder

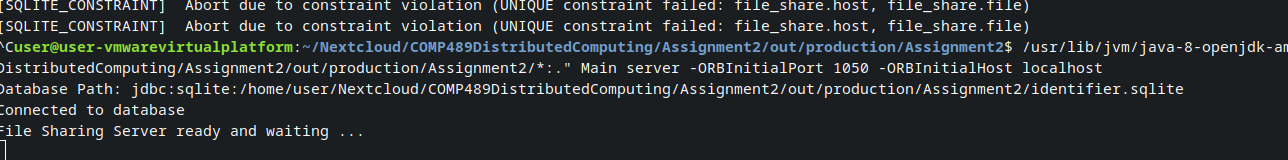
#### Known issues

1. The primary key of the database is the host and file. This means that running the program on the same computer will not update the database if there is an entry with only the port that is different
2. No mechanism currently to delete entries in the file sharing server though the client checks with the other client if a file is available

#### Test Plan

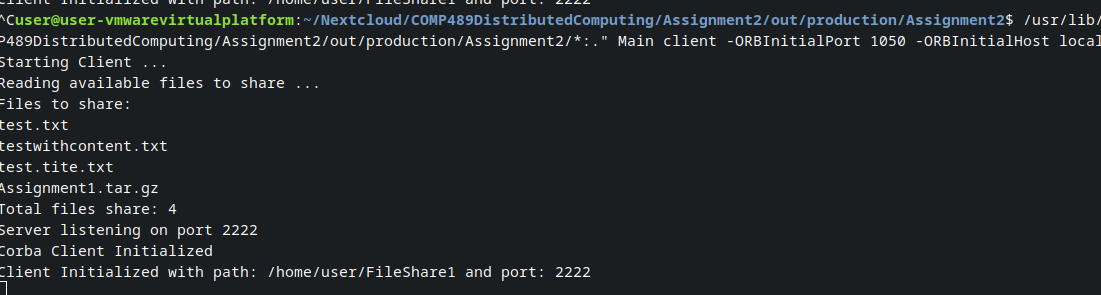
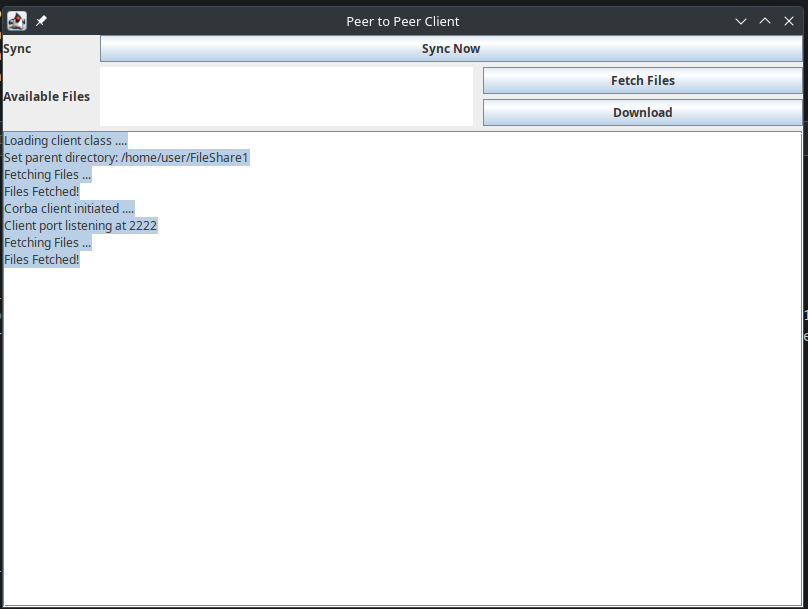
1. Create a share directory and fill it with sample files
2. Run the orb application on your computer. On linux run the following command: orbd -ORBInitialPort 1050 -ORBInitialHost localhost&
3. Run the server using the following command java Main server -ORBInitialPort 1050 -ORBInitialHost localhost
4. Note that if the command above fails this is most likely because of library files. The library files are in the same directory. You can then just add the argument -cp {directory of library files}/*:. eg. java -cp “/home/lib/*:.” Main server -ORBInitialPort 1050 -ORBInitialHost localhost
5. The result of the command should look like below:

Database Path: jdbc:sqlite:/home/user/Nextcloud/COMP489DistributedComputing/Assignment2/out/production/Assignment2/identifier.sqlite Connected to database  
File Sharing Server ready and waiting …

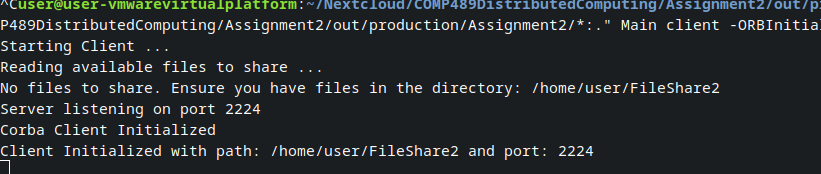
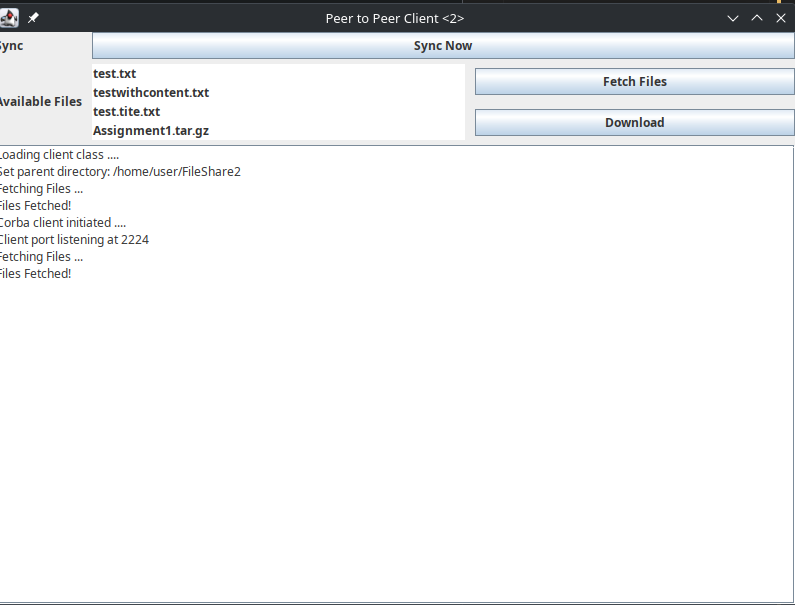


6. Next run the 1st client using the command: java -cp “/home/lib/\*:.” Main client -ORBInitialPort 1050 -ORBInitialHost localhost /home/user/FileShare1 2222

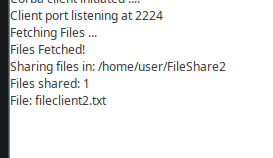
7. A command line and a gui should appear there should be no errors. Refer to pictures SuccessClient1 and SuccessCLI1



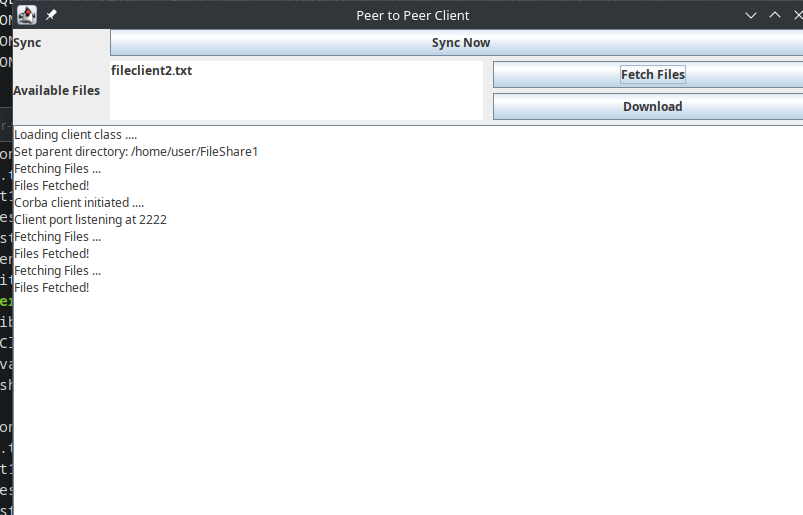
8. Run the command again while changing the parent directory parameter and the port number. Refer to pictures SucessClient2 and SuccessCLI2



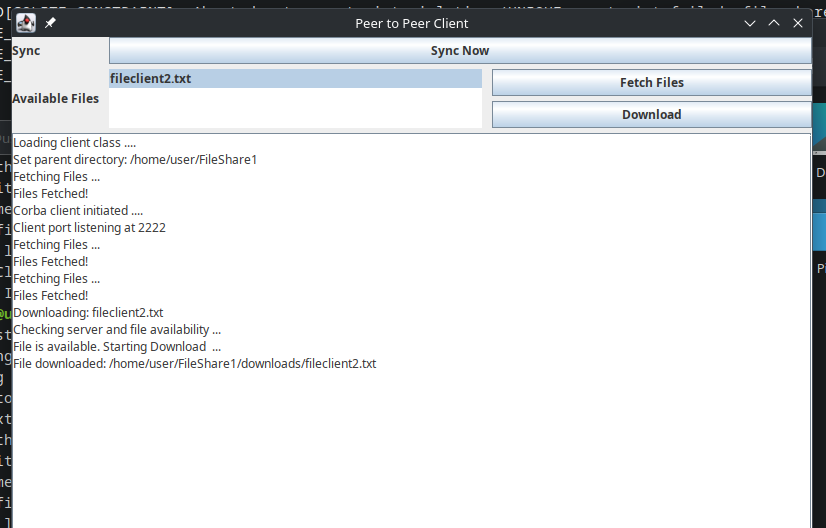
9. To test uploading a file. Copy a file in the parent directory of the 2nd client and click “Sync now”. It should say the files have been shared. Refer to picture SuccessShare2



10. On the first client click on Fetch Files. The new files that have been shared should appear on the available files list. Refer to picture SuccessFetch1



11. To test the download click on a file in the available files then click “Download”. The program should first check the availability and start the download. Any errors will be displayed in the gui or cli. Refer to picture SuccessDownload1



12. The program will create a downloads folder of the parent directory then put the file downloaded in that folder.