Peer to Peer communication with Java

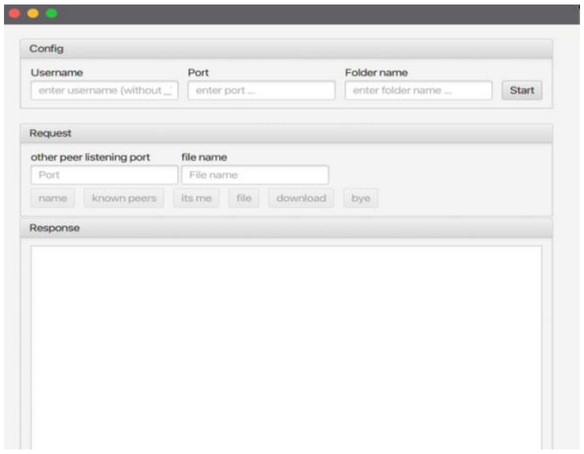
Authors : DESTIN Moïse <12108416> - ROUAICHI Imane <12107934> - ABADLIA Sarra <12114200>

Introduction : In this project, we have implemented the asked questions from project description. The project was implemented with Java version 11, it was written with IntelliJ IDEA version 2021.2.2 ultimate edition. We choose to implement the graphical user interface on JavaFx library, as repository manager, we use maven.

File and Folder Structure : The two main sections of the project are the java folder and the resource folder, the java folder contains all the java class of the project while the resources folder contains all the fxml (interface) file. Moreover, we have two default folders fileDir and fileDir2, considered as the local directory containing the files of a peer. We use the Response.java class to bind all types of messages we want to return through the project. Config.java have the general constant for the project.

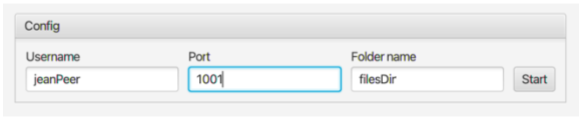
Project implementation

Interface : the project interface has a simple interface, divided into three sections, a configuration section where we configure the basic parameter for the peer, a request section where we perform all the main actions, the response section where we display the majority of responses from other peers.



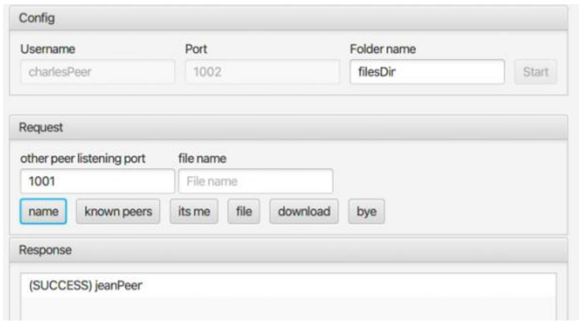
Lauching : To test the program, you can launch multiple instance with intellij idea by checking their internal option "allow multiple instance".

When you start the program, provide the port that we have considered as the identity of the peer, the username and your desire folder directory for your current peer, then click start. This action will start the local server in order to listen to eventual other peer request in the class Server.java.



Username: jeanPeer
Port: 1001
Folder name: filesDir

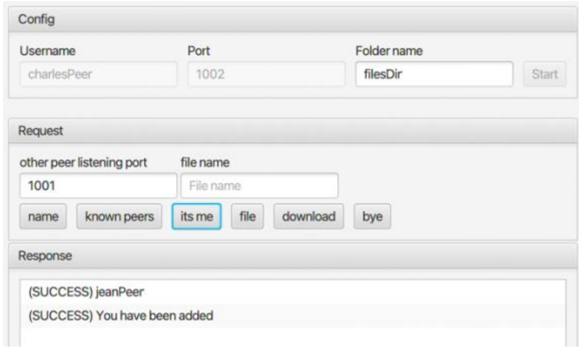
Name ? : For this action, you must give the port(peerId) you want to request and then click on the name action. We internally create a client object that is ready to send the message to the peer. We ask the peer for its name and display the response.



Username: charlesPeer
Port: 1002
Folder name: filesDir
other peer listening port: 1001

(SUCCESS) jeanPeer

It's me ! : For this action, you must give the port(peerId), we ask the provided peer to register us in its internal database of known peers if it can, and then we display the response.



Username: charlesPeer
Port: 1002
Folder name: filesDir
other peer listening port: 1001

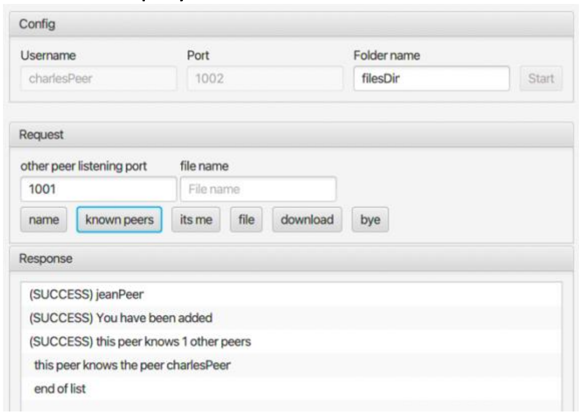
(SUCCESS) jeanPeer
(SUCCESS) You have been added

Known peers ? : For this action, you must give the port(peerId), we ask the provided peer for

Peer to Peer communication with Java

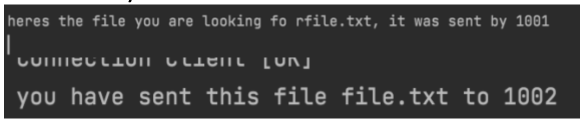
Authors : DESTIN Moïse <12108416> - ROUAICHI Imane <12107934> - ABADLIA Sarra <12114200>

the list of peers that it actually has in its database then we display the list.



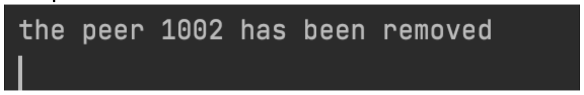
```
(SUCCESS) jeanPeer
(SUCCESS) You have been added
(SUCCESS) this peer knows 1 other peers
 this peer knows the peer charlesPeer
 end of list
```

File ? : For this action, additionnaly to the port field, you must provide the filename text field (with extension), we ask the provided peer if it has this filename in tis folder without expecting an immediate answer, in the peer server that we asked, if it doesn't actually have the file we requested, it send the requests to the other peers in its database, when a peer finds the file in its directory, it send a voila request to the original port (peerId) that requested the file, we use a counter to stop sending the request indefinitely.



```
heres the file you are looking fo rfile.txt, it was sent by 1001
connection client [OK]
you have sent this file file.txt to 1002
```

Bye ? : For this action, additionnaly to the port field, we ask the provided peer to delete this current peer id from its database and ask each of the peers in its database to do the same.



```
the peer 1002 has been removed
```

Flaws : We had put little to no effort into field validation as they are not considered the main focus of the project. Please follow the instructions carefully to test all features without any problems. The voila message is displayed in the console. The by action is logged in the console.