TORrent – file transfer application

Technical Documentation

Author: Marcin Osucha

Programming language: Java

Version: 14.0.1

GUI Framework: JavaFX

Table of Contents

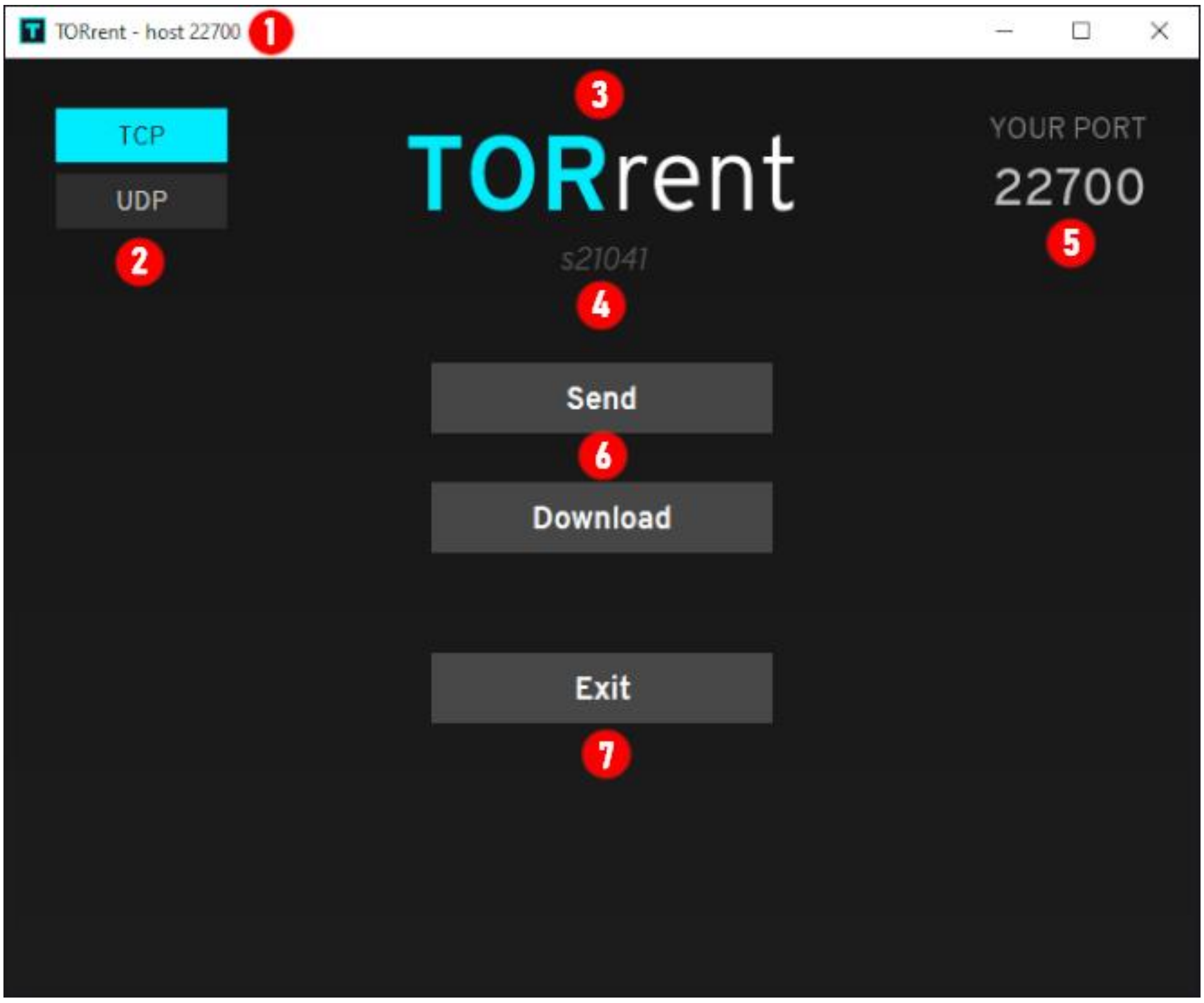
1. Initialization

1. TCP multithreaded server start

TCP server launch first after starting the program. It draws a random port number between 10000 and 60000, then starts a ServerSocket's work on this port and waits for messages. After that application creates a folder on path *user*/Desktop/TORrent/TCP/*port_number*.

2. GUI open

After launching a TCP server, program builds layout and opens the window.



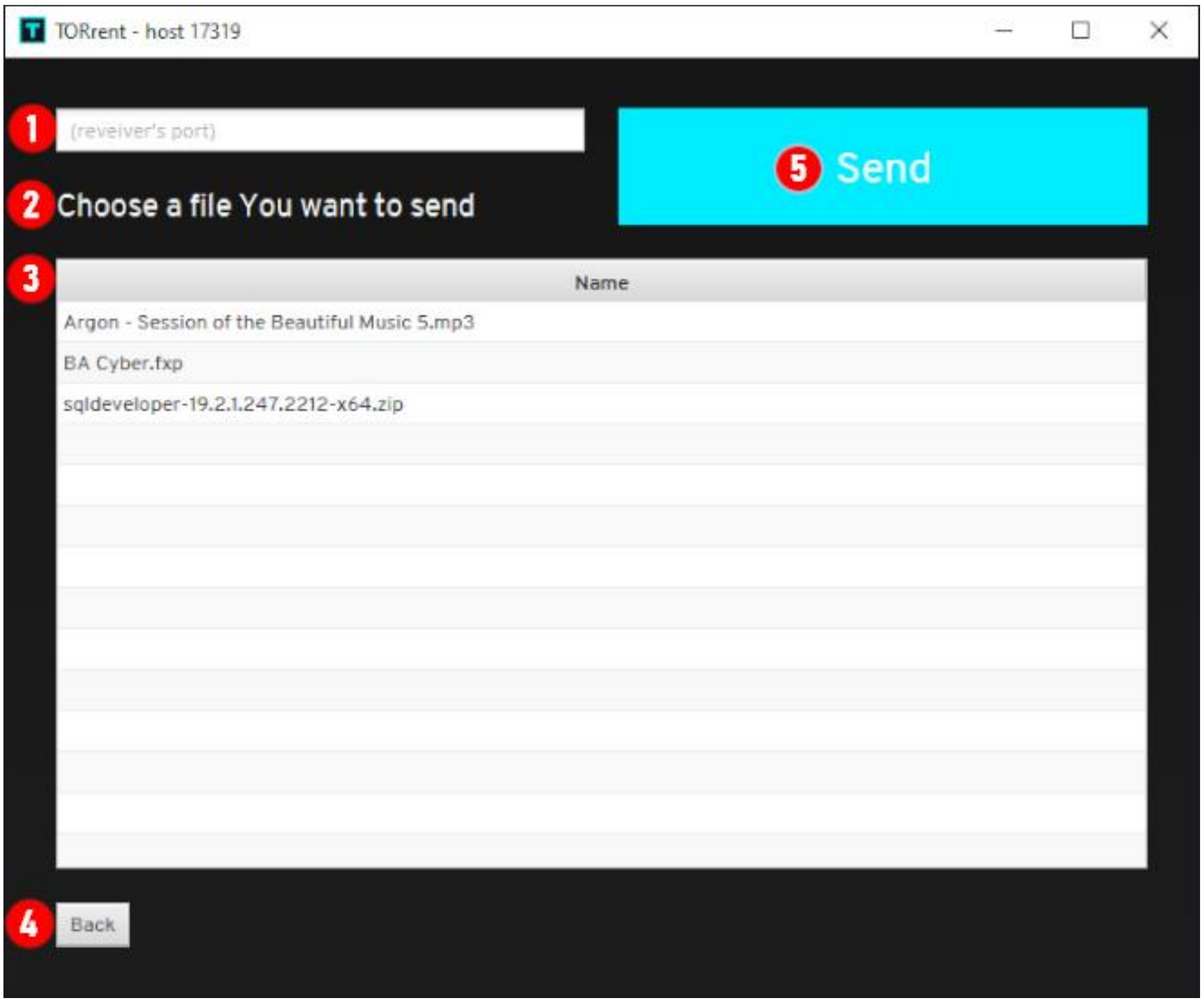
Elements:

1. **Window header** – shows program name and port number program is working on.
2. **TCP and UDP buttons** – those features are not available yet. In the future they will be used to change using protocol.
3. **Program name**
4. **Author ID**
5. **Port number** – always shows port number the program is running on.
6. Action buttons: **Send, Download** (described below).
7. **Exit** button - terminates the program.

2. Send button – sending a file

1. Layout switch

After pressing the send button, program changes layout.


TORrent - host 17319
1
(reveiver's port)
2 Choose a file You want to send
3
Name
Argon - Session of the Beautiful Music 5.mp3
BA Cyber.fxp
sqldeveloper-19.2.1.247.2212-x64.zip
4 Back
5 Send

Elements:

1. **Text field** – here user types port number of the server he wants send a file to.
2. **Label** – informs user what he should do.
3. **Table –** displays user’s files from folder created at the begin. By default table is empty, because there is nothing in user’s folder. He can add files there.
4. **Back button** – return to main menu.
5. **Send button –** initializes sending the file.

2. Sending a file

After pressing the button, program checks if

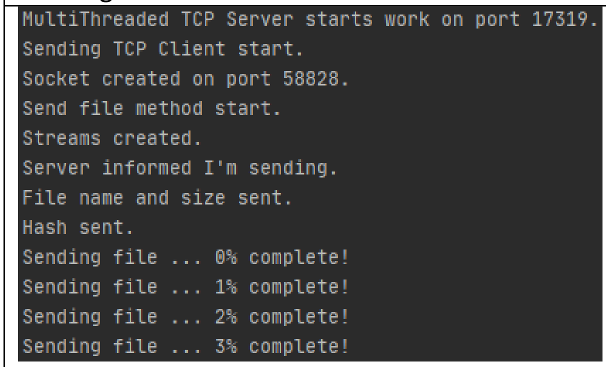
a) typed port is correct – only five-digit string is accepted. Program tries to run a TCP client on this port. If typed port is incorrect, client will not start, application will handle an exception, user will get a suitable message in console and the alert window will open.
b) any file in the table is selected. If not, the alert window will open.

If all conditions are met, connection between hosts begins.

1. Client sends a message "I send" to inform server about its order (flag).
2. Multithreaded server gets a message and starts the appropriate action – awaiting next messages.
3. Client sends file name, size and MD5 hash in sequence.
4. Server gets all 3 messages in sequence.
5. Client sends a file. Alert window informs user that the file was sent.
6. Server gets file, then checks if received hash equals own hash. If so, server informs user about correct saved file in its folder.
7. Connection closes.

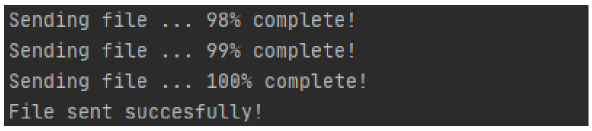
During transmission users on both sides get information in the console like actual working tasks, percentage progress of transmission, remain time.

Sending host

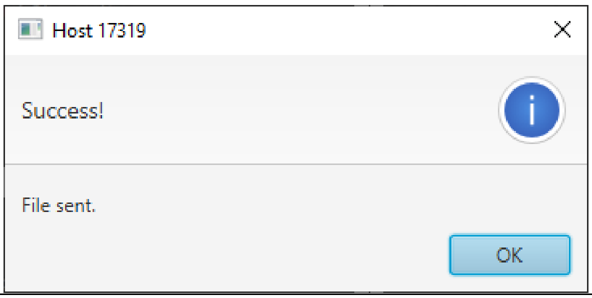


```
MultiThreaded TCP Server starts work on port 17319.
Sending TCP Client start.
Socket created on port 58828.
Send file method start.
Streams created.
Server informed I'm sending.
File name and size sent.
Hash sent.
Sending file ... 0% complete!
Sending file ... 1% complete!
Sending file ... 2% complete!
Sending file ... 3% complete!
```

…



```
Sending file ... 98% complete!
Sending file ... 99% complete!
Sending file ... 100% complete!
File sent succesfully!
```



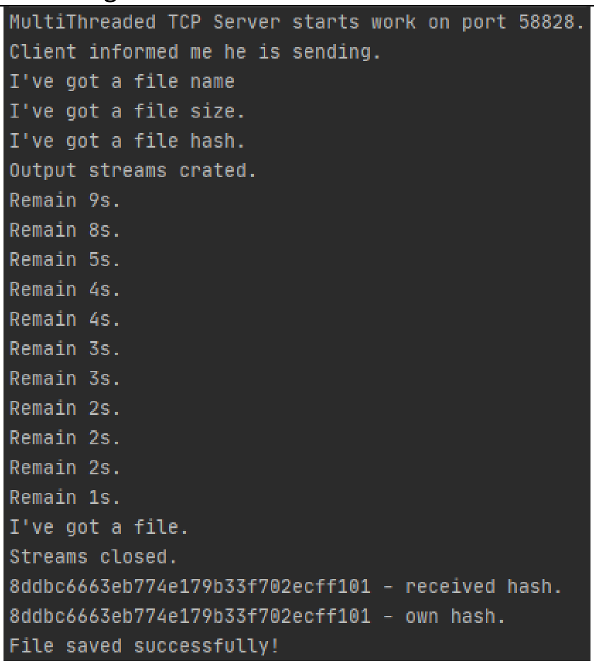
Host 17319

Success!

File sent.

OK

Receiving host

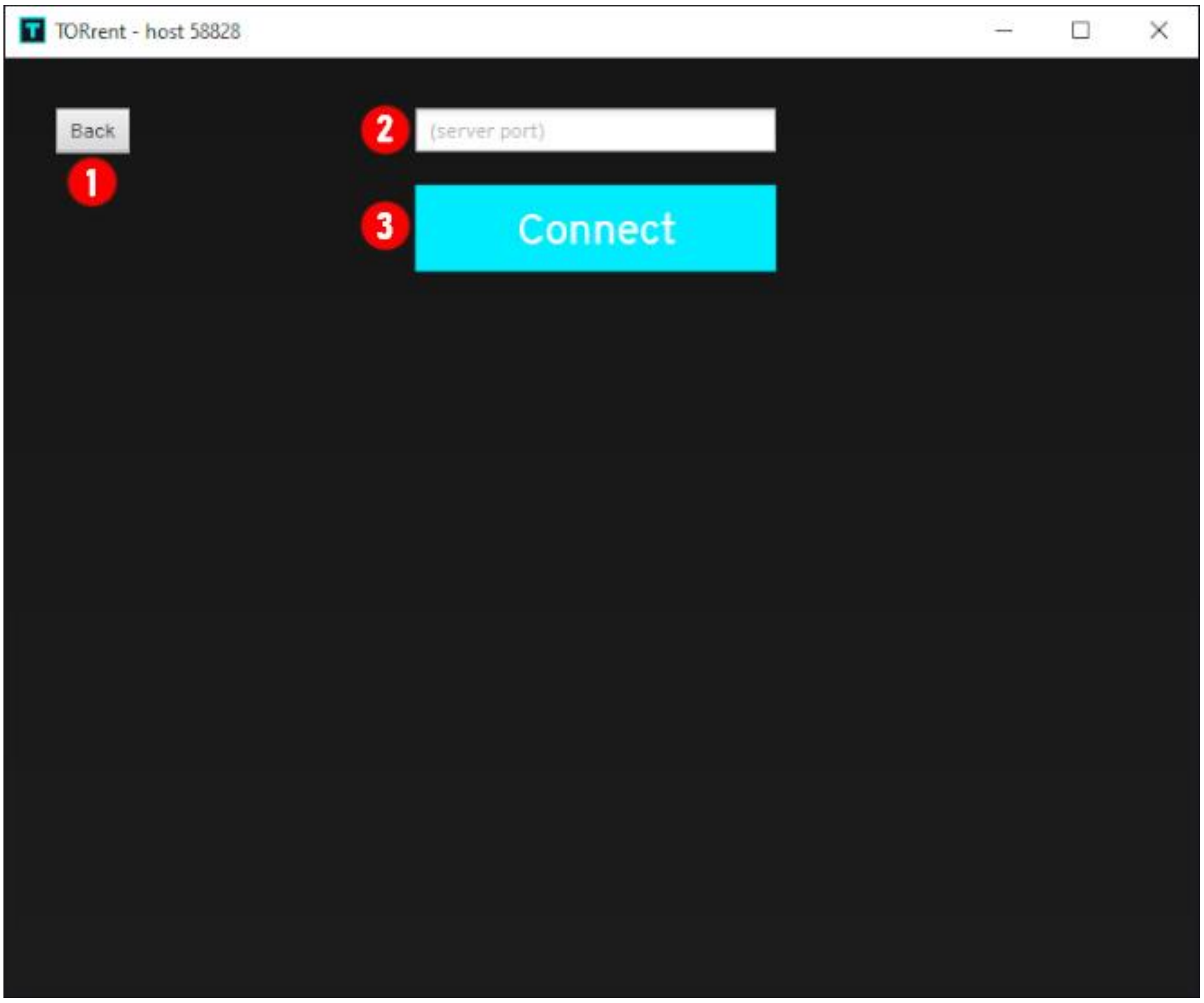


```
MultiThreaded TCP Server starts work on port 58828.
Client informed me he is sending.
I've got a file name
I've got a file size.
I've got a file hash.
Output streams crated.
Remain 9s.
Remain 8s.
Remain 5s.
Remain 4s.
Remain 4s.
Remain 3s.
Remain 3s.
Remain 2s.
Remain 2s.
Remain 2s.
Remain 1s.
I've got a file.
Streams closed.
8ddbc6663eb774e179b33f702ecff101 - received hash.
8ddbc6663eb774e179b33f702ecff101 - own hash.
File saved successfully!
```

3. Download button – downloading a file

1. Layout switch

Pressing the download download button results changing layout to downloading one.



Elements:

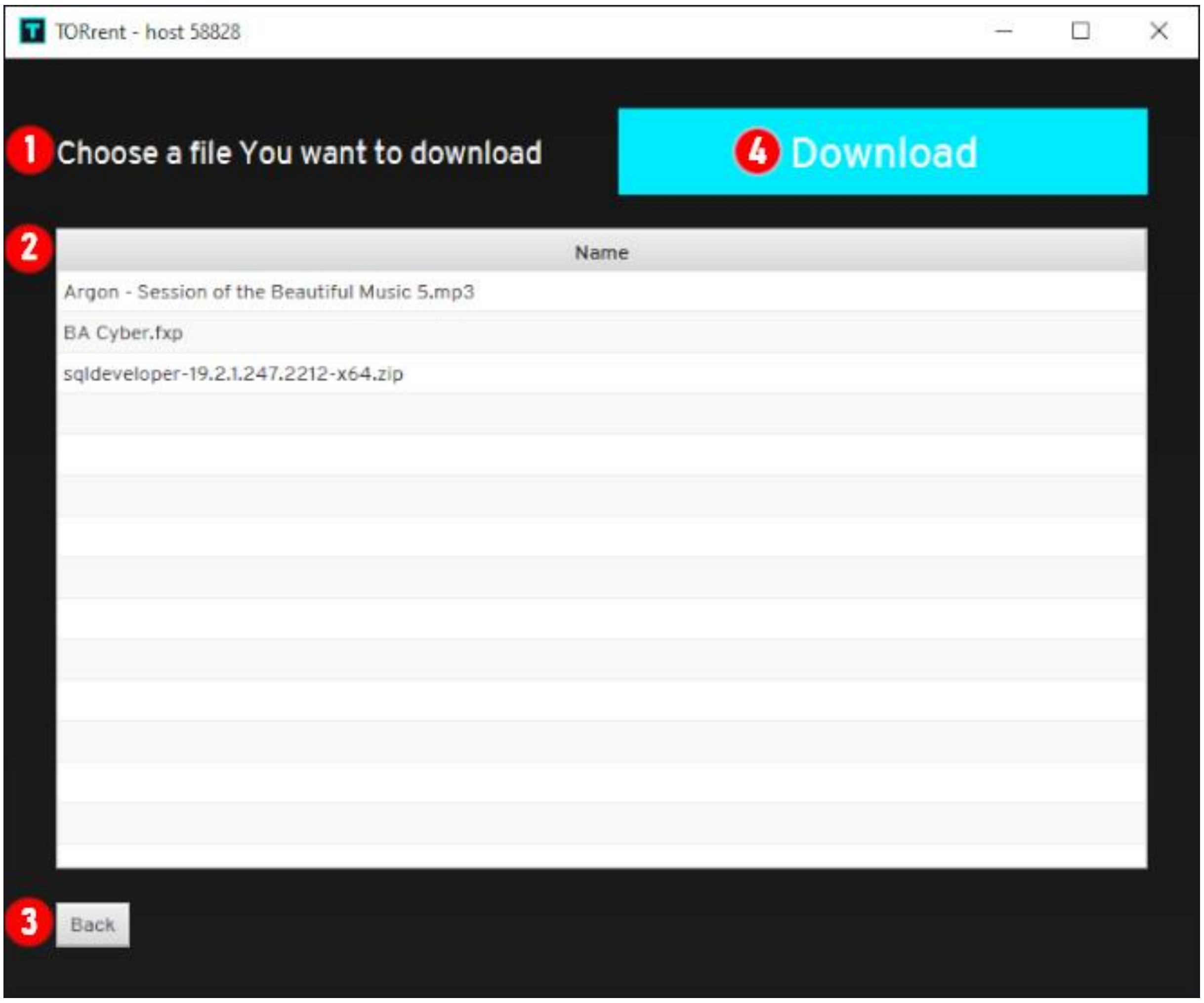
1. **Back button** – return to main menu.
2. **Text field** – here user types port number of the server he wants connect to.
3. **Connect button –** initializes connecting to server.

2. Establishing the connection to the server

1. Program checks if typed port is correct (same as when sending).
2. Client sends a message “I download” to inform server about its order (flag).
3. Multithreaded server gets a message.
4. Server sends the number of its files.
5. Client gets a message.
6. Server sends a file names in a loop.
7. Client gets a file names in a loop.
8. There is a change of layout. Connection is not being closed. Server waits for client’s message.

3. Layout change

The layout is being changed.



Argon - Session of the Beautiful Music 5.mp3
BA Cyber.fxp
sqldeveloper-19.2.1.247.2212-x64.zip

Elements:

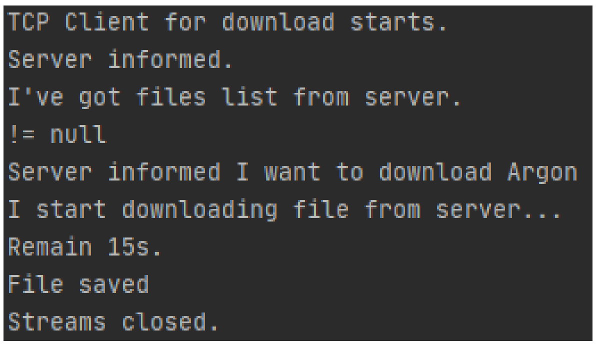
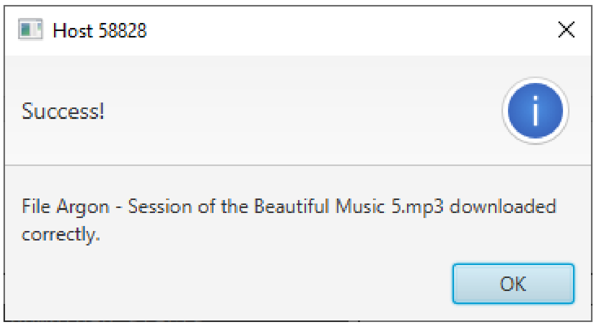
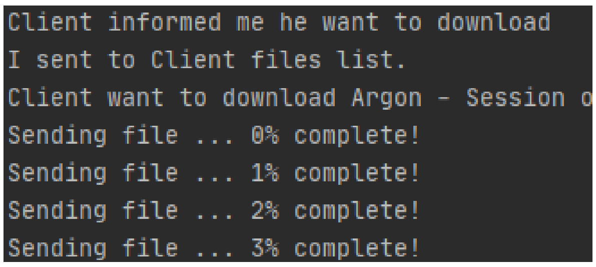
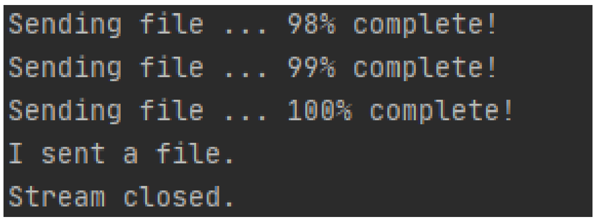
1. **Label** – informs user what he should do.
2. **Table –** displays file names of the server user connected to.
3. **Back button –** return to main menu.
4. **Download button –** initializes download action.

4. File download

1. Program checks if user has selected a file from table. If not, an alert window with suitable information will open.
2. Client sends the selected file name to the server.
3. Server gets a file name.
4. Server sends a file selected by name.
5. Client gets the file. An alert window informs user about correct transmission.
6. Connection closes.

During transmission users on both sides get information in the console like actual working tasks, percentage progress of transmission, remain time.

Downloading host	Sending host


`TCP Client for download starts.`
`Server informed.`
`I've got files list from server.`
`!= null`
`Server informed I want to download Argon`
`I start downloading file from server...`
`Remain 15s.`
`File saved`
`Streams closed.`

Host 58828
Success!
File Argon - Session of the Beautiful Music 5.mp3 downloaded correctly.
OK
	
	`Client informed me he want to download`
	`I sent to Client files list.`
	`Client want to download Argon - Session o`
	`Sending file ... 0% complete!`
	`Sending file ... 1% complete!`
	`Sending file ... 2% complete!`
	`Sending file ... 3% complete!`
	…
	
	`Sending file ... 98% complete!`
	`Sending file ... 99% complete!`
	`Sending file ... 100% complete!`
	`I sent a file.`
	`Stream closed.`