

Report Peer to Peer project  
Version 0.1

Codey Jansen  
Matea Mijocevic

# Introduction

This document describes the application made for peer-to-peer file sharing based on the bit torrent protocol. Herin you can also find instruction on how to use the application.

Index

[1. Introduction 2](#_Toc54968261)

[3. Instructions 4](#_Toc54968262)

[4. Description of implementation 5](#_Toc54968263)

[4.1. Files 5](#_Toc54968264)

[5. Classes 6](#_Toc54968265)

[5.1. StartServer 6](#_Toc54968266)

[5.2. HashGenerator 6](#_Toc54968267)

[5.3. Client 6](#_Toc54968268)

[5.4. StartClient 6](#_Toc54968269)

# Instructions

The following steps are needed to make the application work:

1. Start the server with startserver.java
2. Start the client with startclient.java
3. Select file you want to send.

# Description of implementation

The code is divided in a client/server model. This means the server executes code which in turn will trigger code used on the client. In this scenario the server is the file holder wich will send the file to the client.

The server sends a hash tot he client, the client checks this hash for corruptions/anomolies, The client acknowledges the the hash, the server sends the file and the client recieves.

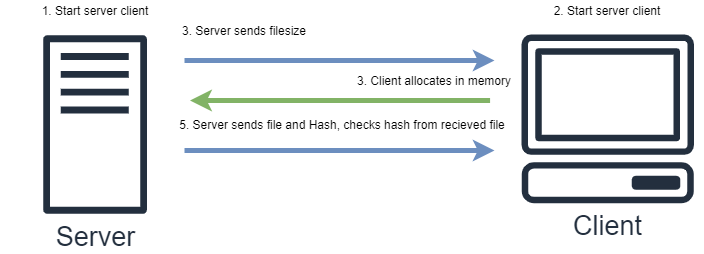


Figure 1 Logical drawing

## Files

This paragraph will list the different java files used for the application. Furthermore we will list the modules used in the application and describe the function of every module.

### Server

#### Modules

|  |  |  |
| --- | --- | --- |
| **Name** | **Function** | **Additional remarks** |
| File |  |  |
| BufferedInputStream |  |  |
| Fileinputstream |  |  |
| ObjectOutputStream |  |  |
| OutputStream |  |  |
| ServerSocket |  |  |
| ArrayList |  |  |
| hash |  |  |

### Client

#### Modules

|  |  |  |
| --- | --- | --- |
| **Name** | **Function** | **Additional remarks** |
| File |  |  |
| Inputstream |  |  |
| Fileinputstream |  |  |
| RandomAccessFile |  |  |
| InetAddress |  |  |
| Socket |  |  |
| ArrayList |  |  |
| IOException |  |  |
| Hash |  |  |

### Hashgenerator

This Java file generates the hashes used in the application.

#### Modules

|  |  |  |
| --- | --- | --- |
| **Name** | **Function** | **Additional remarks** |
| File |  |  |
| Inputstream |  |  |
| Fileinputstream |  |  |
| Security.messagedigest |  |  |
| Util.arraylist |  |  |
| Util.arrays |  |  |
| Util.list |  |  |

# Classes

In this chapter we describe classes used, the function of every class, the attributes and description of each method.

## StartServer

**Class StartServer**: Instantiate class server defined in server.java. Use method runServer from class Server with the attribute port 8080.

## Server

**Class server**: Loads file, calls hash generator from Hashgenerator.java and saves it to array (SHA1). Opens outputstream and sends file while printing the sending size.

Method runServer: The runServer function creates a new server and opens a port on said server based on the port Integer that it gets from calling function. After opening said socket it sends it a new ArrayList of the String datatype and closes the server and socket. If anything goes wrong the function will throw an Exception.

## StartClient

**Class StartClient**: Instantiate class client defined in client.java. Use method runClient from class client with the attribute port 8080.

## Client

**Class client:** The allocateFileMemory function creates a new instance of the File class. The function also allocates memory for the file based on the sizeInBytes argument and returns the new instance of the File class. If anything goes wrong the method will return a IOException.

The getFile method retreives a instance of the File class based on the data it gets from the socket it calls. After getting the file it hashes the new File instance using the create SHA1 method from the HashGenerator class. After hashing the file it compares the newly made hash with the summary hash the method has gotten in the form of an argument. The method does this by again calling the HashGenerator but using the compareHash method this time. If the two hashes are identical the method print that the file is not corrupted. If not, the method prints that the files are corrupted. If anything goes wrong while executing the method it will throw an Exception.

## HashGenerator

Class HashGenerator: The hashgenerator class creates a SHA1 hash for the server to use. In the method compareHash it compares the hash previously generated.