MANUAL
A Simple Decentralized Peer-to-Peer File Sharing System

In this program, I have design a simple decentralized P2P file sharing system that has a single component which behave as a Server (Index & Peer Server) as well as a Client.

Server (Index as well as Peer Server) – As Index Server, it indexes the contents of all the peers that register with it. It also provides search facility to peers to get the contents from other peers.
It uses MultiMap data structure to store single key multiple values pair. Key will be file names and values are the peerIDs containing that particular file.

It provides the below two interfaces to the user:

Registry(String[] fileNames, String peerID) – This function takes a list of file names along with Peer's port number to register in MultiMap.

Search(String filename) – This function takes a file name from the user to search in the MultiMap. The function then returns the peerIDs containing the file to the requested peer in the form of Collection.

As a Peer Server, it will wait for file requests from other peers and send the requested file in response.
It provides one interface to the Peer Client:

Obtain (String filename) – This function takes the file name to be sent to the user.

Client (Peer Client) – The Client can use Index Server's registry and search facility to register its own flies and search to locate files with other peers to download. And for downloading file from other peer it will use Peer Server's obtain method.

Steps to be taken before running the code:

1. Extract the file PROG3_DHANDHARIA_SHORABH.ZIP
2. Edit DHT.java to change the variable "FILE_PATH" as per your system. There is only one place where it needs to be changed (Line #21)
3. For each Peer a new folder has to be created in "FILE_PATH" with name as the Server name which should be same as server name given at command line arguments. Files which needs to be sent and receive will be in this folder. At the receiving side, if the folder path doesn't exist it will automatically create one but on the sending side a file needs to be present.
4. Place the config file also in "FILE_PATH"
5. Add guava-18.0.jar in the CLASSPATH of environment variable (Download Link: Google Code - GUAVA 18.0)

Note: All servers need to be started within 60 seconds else the program will throw Exception. I am putting thread to sleep for 60sec before connecting to other servers to buy us some time to run rest of the 7 servers since we cannot do that simultaneously.

Steps to follow to run this code:

1. Open Command Prompt
2. Compile all .java files in the extracted folder using

```
javac *.java
```

3. Run the DHT.java file passing server name as argument.

```
Java DHT Server0
```

4. Now you will be able to see a Menu. Don't give any input until STEP 7
5. Repeat STEP 3 again for 7 times to start the remaining servers within 60 seconds

```
Java DHT Server1
Java DHT Server2
Java DHT Server3
Java DHT Server4
Java DHT Server5
Java DHT Server6
Java DHT Server7
```

6. After 60 seconds all 8 Servers will be connected to each other.
7. You can now use the Menu displayed on each server. Choose Register File(s) on one or multiple Servers.
8. Enter File Name with Extension and press Enter
9. It will show Success or Failure. It will display the Distributed Hash Table on the server where Registry happened.
10. It will again Display a Menu. Choose Search File.
11. Enter File name with Extension.
12. It will Display the location of the file.
13. Now choose Obtain
14. Enter file Name and the desired file location displayed earlier
15. File Downloaded message will be displayed
16. Choose Exit.