File Content of the Project

Within the project folder you will find the following files:

1. DirectoryServer.java
2. PeerClient.java
3. PeerServer.java
4. Three jpeg files: d.jpeg, l.jpeg and t.jpeg for demo

Compile and Setup Instructions

Directory Server(s)

Computers that are designated as the Directory Server(s) will require a copy the file DirectoryServer.java.

To compile this file, run the following command a terminal that is opened to the folder which contains said files:

```
>javac DirectoryServer.java
```

To start the server, run the following command a terminal that is opened to the folder which contains said files:

```
>java DirectoryServer [port] [serverID] [successorPort] [successorIP]

port: sets the port number of the current Directory Server
serverID: sets the current ID for the Directory Server
```

```
successorPort: sets the TCP port number of the next Directory Server
successorIP: sets the IP address of the next Directory Server

Example:
>java DirectoryServer 20510 1 20510 192.168.1.10
```

Peer to Peer Client and Server

Computers that are designated as the Client Computers(s) will require a copy of the files PeerClient.java and PeerServer.java.

To compile this file, run the following command a terminal that is opened to the folder which contains said files:

```
>javac PeerClient.java PeerServer.java
```

To start the client, run the following command a terminal that is opened to the folder which contains said files:

```
>java PeerClient [clientPort] [directoryServerIP] [directoryServerPort]

clientPort: sets the TCP port number of the Peer Client
directoryServerIP: sets the IP address of the Directory Server
directoryServerPort: sets the UDP port number of the Directory Server

Example:
>java PeerClient 20510 192.168.1.10 20510
```

Usage Instructions

When the Peer Client successfully connects to a Directory Server the user will be able to perform actions via command line terminal. **OMIT THE .JPEG EXTENSION WHEN ENTERING FILE NAMES. (So instead of d.jpeg just enter d)**

```
List of Peer Client Commands:
U - Update/Inform and Update
Q - Query for Content
E - Exit
```

```
Example:
>java PeerClient 20510 192.168.1.10 20510
>Client is starting...
>MESSAGE FROM SERVER -> Client Connected To Server
>Please select a function: Enter U to Upload, Q to Query and E to Exit
>Please Enter: U
>Please enter File Name: d
>MESSAGE FROM SERVER -> File Succesfully Added To DHT

>Please select a function: Enter U to Upload, Q to Query and E to Exit
>Please Enter: Q
>Please enter File Name: d

>MESSAGE FROM SERVER -> Content is available, IP released
MESSAGE FROM PEER SERVER -> New Connection Opened On The Port 20511
**HTTP Request Sent to Server** BEGIN
GET /d.jpeg HTTP/1.1
Host: Gizmo
Connection: Close
Accept: image
Accept-Language: en-ca

**HTTP Request Sent to Server**END

**HTTP Response Got From Server** START
HTTP/1.1 200 OK
Connection: Close
Date: Wed, 10 Apr 2019 16:15:25 GMT
Last-Modified: Wed, 10 Apr 2019 16:14:42 GMT
Accept-Ranges: bytes
Content-Length: 11247
Content-Type: image

**HTTP Response Got From Server**END
```

System Documentation

Data Structures

Directory Server

Our Directory Server(s) has it own class that holds all the entirety of the server's data (ID, IP, Port numbers, error codes, records, etc…)
The record table is stored as a **HashTable** containing tuples of strings <String, String> corresponding to the Content Name and IP Address.

```
DirectoryServer.Java

public static class Server {
    final int statusCode_200 = 200; // OK Request
    final int statusCode_400 = 400; // Bad Request
    final int statusCode_404 = 404; // Not Found
    final int statusCode_505 = 505; // HTTP Version Not Available

    Thread mainTCPThread;        // TCP Thread
    Thread mainUDPThread;        // UDP Thread

    ServerSocket server_TCPSocket;  // Main TCP socket
    DatagramSocket server_UDPSocket; // Main UDP socket

    int initialPortNumber;       // Initial port for UDP
    int PortNumber;              // Port Number of the first DHT Server
    int ServerID;                // Unique ServerID
    String IP;                   // IP Address of the server
    int SuccessorPortNumber;     // Successor server's first active port number
    String SuccessorIP;          // Successor server's IP
    int SuccessorServerID;       // Successor server's server ID

    // ArrayList for UniqueUDP
    ArrayList<UniqueUDP> client_UDPList = new ArrayList<UniqueUDP>();

    // HashTable ArrayList
    // Key = File Name
    // Value = IP Address
    public static Hashtable<String, String> cList = new Hashtable<String, String>();

    ...
}
```

Each client’s UDP connection is stored in a **ArrayList** of it own class (UniqueUDP); this class stores information such as the Client’s IP, Socket, states codes, etc...

```java
// Unique UDP for PeerClients
public static class UniqueUDP {
    final int statusCode_200 = 200;   // OK Request
    final int statusCode_400 = 400;   // Bad Request
    final int statusCode_404 = 404;   // Not Found
    final int statusCode_505 = 505;   // HTTP Version Not Available
    int unique_Port;                  // Unique UDP port
    String client_IP;                 // IP address of the PeerClient
    Thread unique_Thread;             // Unique thread.
    DatagramSocket unique_UDPSocket;  // UDP socket.
    String SuccessorIP;               // IP Address of the successor server.
    int SuccessorPortNumber;          // Port number of the successor server.

    ...
}
```

Client

The Peer Client is much simpler than the Directory Servers. Along with keeping track of it’s IP, Port, Socket information, it also holds the filename as a string.

```java
PeerClient.java

public static class Client {
    int peer_ServerPort;                         // Client server port.
    String[] fileName;                           // Filename.
    int[] server_PortNumbers = new int[4];       // Opening ports on DHT servers
    String[] serverIPs = new String[4];          // IPs of DHT servers
    DatagramSocket clientUDPSocket;              // UDP

    ...
}
```

Design Choices

We tried, to the best of our abilities, to conform with the provided design provided in the project instructions.

The entirety of this project was achieved using vanilla Java through the use of java.net and java.io packages; we did not find any need to use 3rd party packages (if that was even allowed).

The messages/commands as passed through the terminal and are shortened, for instance, “Inform and Update” -> U, as each command had an intuitive and unique letter.

Important Algorithms

For each record in the Directory Server, a **Hash Function** is used to generate the Server ID. This value is calculated by summing the decimal values of ASCII characters of the content and modding it by 4.

```
PeerClient.java

// Make a ServerID using mod4 for one of the four DHT Servers
int calculate_ServerID = 0;
for (int i = 0; i < userInput.length(); i++) {
    calculate_ServerID += (int) userInput.charAt(i);
}
calculate_ServerID = calculate_ServerID % 4;
```