**Cloud Assignment**

**Requirements**

Create a File Sharing “Cloud” Application

**Server Side**

* + Allow any number of computers to connect
    - Be able to serve multiple clients concurrently.
  + When connected via the network your App must:
    - Respond with the name of the Developer
    - Wait for commands from client
  + When requested by client for corresponding command:
    - Respond with a list of available Files in the share directory on the server
    - Transfer any requested File
  + All files that are available (binary image files) to be shared must be stored in a single directory
    - i.e. C:/Users/bondstoneXX/workspace/Cloud/store/
  + Should have an XML configuration file with information of
    - Developer name
    - Port
    - Shared directory path
    - History file path

Name of the file could be e.g.; serverConfig.xml

* + Any data that needs to be stored should be in XML format. Name of the file could be e.g.; serverUploadHistory.xml
    - Required Data to be Stored
      * Name of the developer (the one listed in the server configuration file)
      * History of client IP Addresses that downloaded from the server (socket.getInetAddress().getHostAddress).

**Client Side**

* + Should have an XML configuration file with information of
    - Server Port and hostnames
      * Port to listen on/Connect to (These are the same port)
    - Download directory path
      * Example path could be c:\downolad on client machine.
    - Download History file path
      * Name of the file could be e.g.; clientDownloadHistory.xml
    - To connect to an available host computer from a drop down – Use IP addresses in the drop down list.
  + Client functionality
    - Get the list of files from the connected server.
    - Download any File – User can and double click a file from the list of file names or select a file and hit download button to download a file from server.
    - View the files being shared by the User’s computer
      * `This will be the shared folder on the client machine. Keep in mind that the client machine will also have a shared folder as it can run the server program to serve clients from other machines. To display these files, the client GUI application should not have to use socket programming but just use File object’s listFiles() method.
    - All downloaded files should be stored in a separate directory (also stored in the XML file) on client machine.
      * Save the list of downloaded file names in the XML file. Name of the file could be e.g.; clientDownloadHistory.xml

**Flow of Client / Server Communication**:

***Step1: Connect***

**Client**

Once the GUI application is loaded, the user will be able to select an IP address of a server in the classroom from the drop down and hit the Connect button to connect to that server.

**Server**

After the client connects to the server, the server will send the developer name to the client followed by *END* command.

***Step2: List Files***

**Client**

After receiving the developer name, client will send *LIST\_FILES* command to server for a list of files from the server’s shared directory.

**Server**

Upon receiving the file list request, the server will read all the image files (extensions could be, .gif, .jpg, .tif, .tiff, .bmp, etc.) from its shared directory. It will send the file names delimited by *END* command. The file list will be followed by *END\_OF\_LIST* command.

***Step3: Download File***

**Client**

Client will populate the JList on “Remote File List” panel with file names. Once the JList is loaded and the user double clicks a file name in the JList, client will send *SEND\_FILE* command to server. Then it will send the file name followed by *END* command. Client will check whether the server sends *FILE\_DOES\_NOT\_EXIST* or *FILE\_EXISTS* command. If server sends *FILE\_DOES\_NOT\_EXIST* command, then client will throw an exception. If server sends *FILE\_EXISTS* command then client will read the length of the file followed by *END* command and then read all the bytes sent by server based on the specified file length. Client will then save the file binaries with the same file name in the download directory.

**Server**

Upon receiving *SEND\_FILE* command, the server will read the file name and then check if the specified file exists in the shared directory. If it does not exist the server will send *FILE\_DOES\_NOT\_EXIST* command. But, if it exists then the server will send *FILE\_EXISTS* command and write length of file followed by *END* command. And then write the file content in byte arrays to the socket output stream.

**Commands interface:**

Commands class contains all the command constants.

**public** **interface** Commands

{

**public** **static** **final** **byte** *END\_OF\_LIST* = '\r';

**public** **static** **final** **byte** *LIST\_FILES* = 'L';

**public** **static** **final** **byte** *SEND\_FILE* = 'F';

**public** **static** **final** **byte** *END* = '\n';

**public** **static** **final** **byte** *FILE\_EXISTS* = 'Y';

**public** **static** **final** **byte** *FILE\_DOES\_NOT\_EXIST* = 'N';

}

**GUI Interface:**

The client GUI window will have 3 major panels for displaying file list from server, download folder and local folder (shared server folder on client machine – a server program can be run on a client machine as well, and thus having a shared folder on client machine.).

|  |  |
| --- | --- |
| Remote File List Panel | Downloaded File List Panel |
| Local File List Panel | |

Each panel will automatically populate file list as soon as the information is available. Each panel may also include a refresh button so that the user can see the updated file list. For example the “Remote File List Panel” will populate the file list when file list information is available from server for the first time. But, a few minutes later if the user clicks the Refresh button, client will again send *LIST\_FILES* command to server to get latest file list.

**Swing Examples Reference URL(s):**

<http://www.java2s.com/Code/Java/Swing-JFC/CatalogSwing-JFC.htm>