COMP2396B (OOP & Java) Assignment 4

Introduction

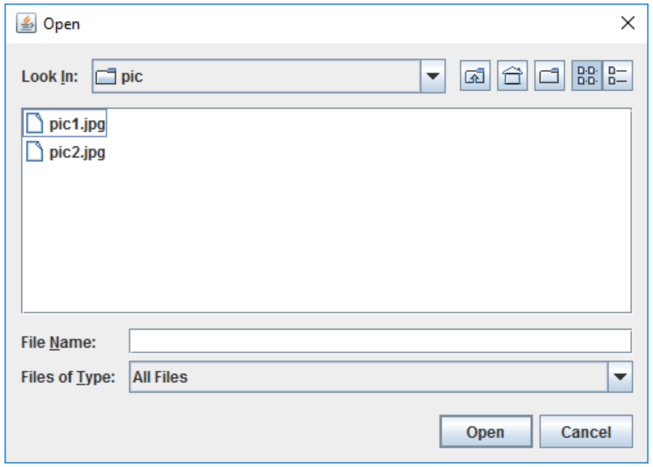
You are required to write an **image sharing system** using P2P protocol

USE CASE:
Can be used in an teacher-students environment, where the teacher shares an image with students of a class in a classroom

Server (the Teacher)

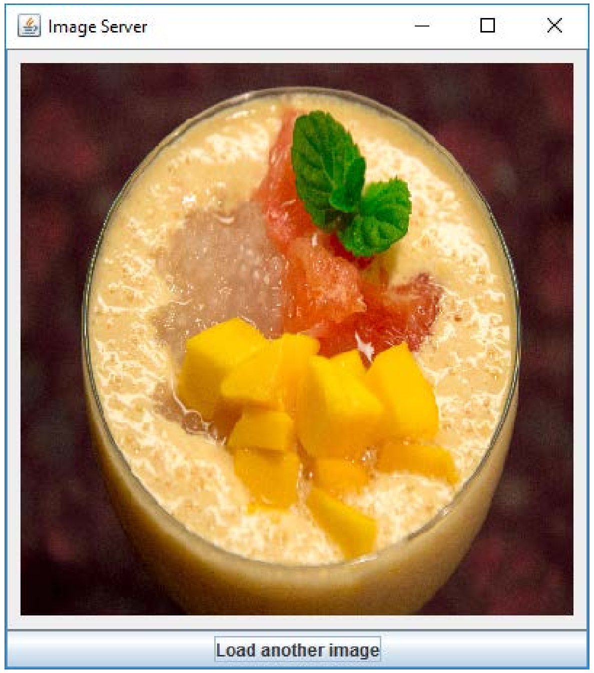
When ImageServer.class is executed, a file chooser should be presented to ask for an image file.
The command should take an argument which specifies max. no. of peers to be returned to a login request (default is 5), e.g.

```
C:> java ImageServer -20
```



The image is then loaded and displayed as shown in figure. If the image file fails to load, the program terminates.

The image should be **resized** and **scaled** to fit into a canvas of **600*600 pixels** in size. The button below can be used to allow user to change the current image. If the new image fails to load, the old image is retained.



Update image

- If the Server switches the image, all peers will be notified to start the download again.

- The Server should send an “image update” message to all peers.

Server Process

- The server should be listened to port no. 8000
- The Server should return a list of peers upon a Peer login, and the maximum no. of peers is according to the command option (default is 5)
- The Server should maintain the list of current active peers (IP address and port no.), and should display the list (by polling the list of peers regularly, may need a “ACTIVE_CHECK” message for every 60 seconds)

- The image is separated into blocks of 30*30 pixels in size and be shared

- The Server should display the status of the list of current active peers, which include user name, last login time, number of active peers, current image file name, no. of blocks, etc.

Server Process

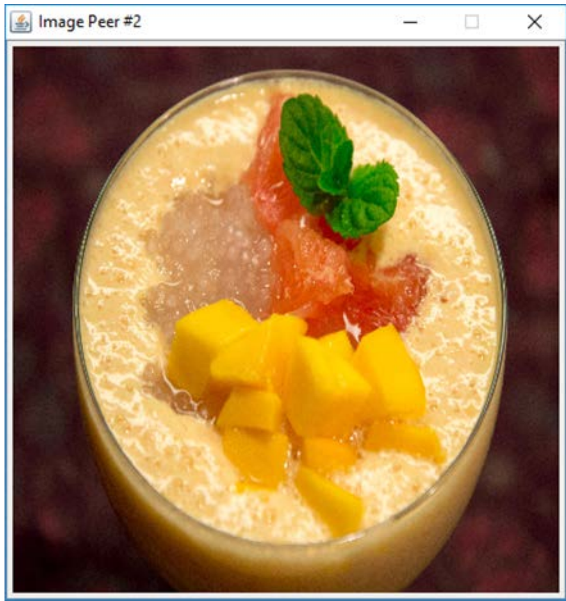
1. Create the GUI to handle image selection and loading, and button to load new image
2. Create a thread to do the “active peer check” and “display peer status”
3. Create a thread to do the “image sharing”
4. Create server socket and listen to port 8000
5. When a peer connects to the server, add the peer to the peer list, returns the list of peers to the peer

Server Image Sharing Thread

1. Find a port that the thread can be listened to for incoming peer
2. Create a server socket listen to the port
3. When receiving an incoming request, start a new thread to process the request

Peers (Students)

- When ImagePeer.class is executed, it should ask the peer to login, and wait for a reply message from the Server which contains list of peers.
- Immediately after the GUI is shown, the peer should connect to the server and start to download the image from the server and any other peers. Program should be terminated if it fails to connect to any server.



Peer Process

- The Peer should keep the list of known active peers
- Try to download blocks of image from the list of peers in parallel.
- The Peer should store the list of blocks that have received and where they come from
- For each known active peers, the Peer should assign a random block no. to be requested from the peer

- Accept connection from other peers and send out blocks of images the other peers needed.

- The Peer should respond to an active check from server

Peer Process

1. Find an available port to be used for incoming request
2. Start a thread listening for incoming request at port no.
3. Connect to the server with port no. 8000
4. Login to the server and retrieve the list of peers
5. For each peer in the “list of peers”, create a connection to the peer, find a block no., and request the block from the peer

Peer-to-Peer Process

```
1. Create the Peer-List
2. Initialize the Image-Block-List
3. While (true)
4. If all blocks in Image-Block-List are filled; break out of while
5. For each peer P in the Peer-List that is free
6.      Start a thread to download a block from P
7. Wait for a random-period
8. EndWhile
```

Peer-to-Peer-Download Thread

1. Mark Peer P as busy
2. Select a block-no that is no yet filled in the Image-Block-List
3. Mark the block-no waiting
4. Connect to P and request the block-no
5. Close the connection
6. If block-no returns success, insert the return block to Image-Block-List, mark block-no filled
7. If block-no returns fail, mark block-no not-filled
8. Mark Peer P as free

Communication Message

The messages between the Server and the Peer, and between 2 peers should be defined as a class of the following structure (suggested):

Me: sender’s user name
You: receiver’s user name
Command: String
Data_image_name: name of image file
Data_block_no: for image, the block no.
Data_content: data content (depends on command)
Peer_list: list of peers

Login Messages

Peer to Server (request)

- Me: Login user name
- You: “Teacher”
- Command: “LOGIN”
- Data_image_name: NULL
- Data_block_no: NULL
- Data_content: password
- Peer_list: NULL

Server to Peer (respond)

- Me: “Teacher”
- You: user name
- Command: “LOGIN_OK” / “LOGIN_FAILED”
- Data_image_name: image file name
- Data_block_no: no. of blocks
- Data_content: NULL
- Peer_list: list of peers

Login Messages

```
Server to Peer
    Me: “Teacher”
    Command: “LOGIN_FAILED”
```

P2P Get Image Block Messages

Peer to Peer (request)
Me: Sender user name You: Receiver user name Command: “GET_IMG_BLOCK” Data_image_name: file name Data_block_no: the block number Data_content: NULL Peer_list: NULL
Peer to Peer (reply)
Me: Sender user name You: Receiver user name Command: “SEND_IMG_BLOCK” Data_image_name: image file name Data_block_no: the block number Data_content: the data Peer_list: list of peers

P2P Get Image Block Messages

```
Peer to Peer (reply)
```
Me: Sender user name
You: Receiver user name
Command: "BLOCK_NOT_AVAILABLE"
Peer_list: list of peers

```
Peer to Peer (reply)
```
Me: Sender user name
You: Receiver user name
Command: "NAME_MISMATCH"
Peer_list: list of peers

Image Update Messages

Server to Peer

Me: “Teacher”

You: Receiver user name

Command: “IMAGE_UPDATE”

Data_image_name: file name

Data_block_no: total no. of blocks

Data_content: NULL

Peer_list: NULL

Peer to Server (reply)

Me: Sender user name

You: “Teacher”

Command: “IMAGE_UPDATE_OK”

Data_image_name: image file name

Data_block_no: NULL

Data_content: NULL

Peer_list: NULL

Peer Active Check Messages

Server to Peer

Me: “Teacher”

You: Receiver user name

Command: “ACTIVE_CHECK”

Peer to Server (reply)

Me: Sender user name

You: “Teacher”

Command: “ACTIVE_CHECK_OK”

Data_image_name: image file name

Data_block_no: no. of blocks received