CSIS0396A/COMP2396A - Assignment 5

Due: 29th Nov, 2014 23:30

Introduction

This assignment tests your skills on writing **networking** program in Java.

You are required to write a **peer-to-peer** (P2P) image sharing program. A **hybrid P2P structure** is used in this assignment, where a server is used as the source of an **image** as well as that of the **list of peers** available. Details about the P2P structure will be discussed in the tutorial.

You need to write **two main programs**. `ImageServer.java` will be the server program, and `ImagePeer.java` will be the client (peer) program. Notice that a server will also act as a peer to share the image.

Part I. Server interface

When `ImageServer.java` is executed, a file chooser should be presented to ask for an image file. (Figure 1)

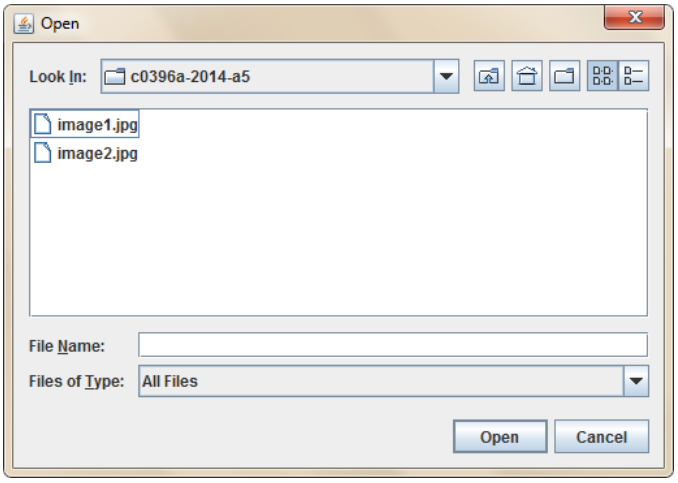


Figure 1: File chooser

Figure 2: Server GUI

The image is then loaded and displayed as shown in figure 2. If the image file fails to load, the program terminates.

The image should be **resized and scaled** to fit into a canvas of 400x400 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.

Part II. Peer interface

When `ImagePeer.java` is executed, it should ask for the server's address to connect to. (Figure 3)

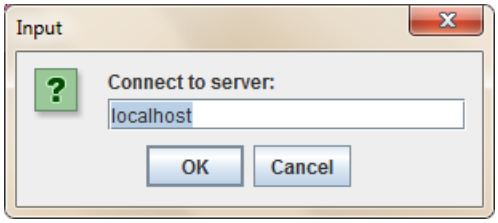


Figure 3: Peer start up screen

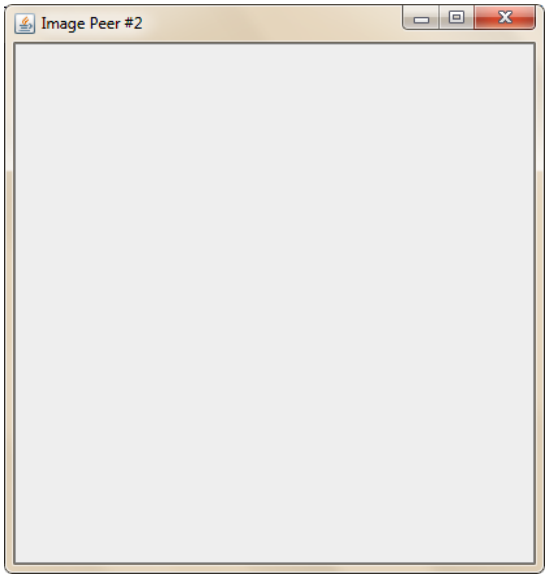


Figure 4: Peer GUI

An empty canvas should then be presented as shown in figure 4. Like the server, the canvas size should be 400x400 pixels.

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. (See details below)

Part III: Peer initialization

The server should be started at port 8000. The server needs to maintain two things:

1. The **image**, separated into blocks of 20x20 pixels in size
2. A list of **active peer** (their address and port number)

When a peer program is started, it should connect to the server through port 8000. The server should than update the list of active peer, and the peer should collect the list of current active peer from the server. This connection can be closed afterwards.

Part IV: P2P operation

A peer should always do two things:

1. Try to **download blocks** of image from the list of peers in **parallel**.
2. **Accept** connection from other peers and **send out blocks** of images the other peers needed.

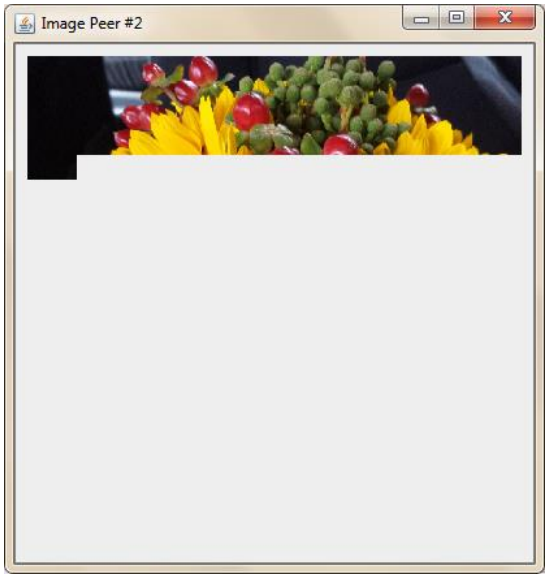
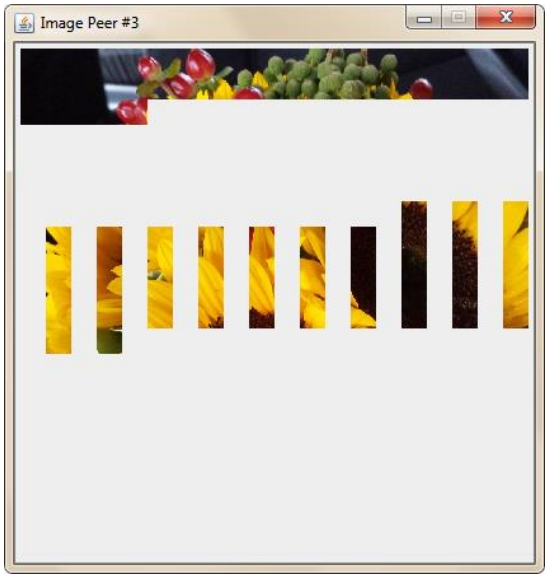
Notice that the server itself is a special peer that is the source of the image to be sent. Details about the suggested P2P operation will be introduced in the tutorial.

Part V: Update image

If the user **switches** the image in the server, all peers will start to download the blocks of the new image and cover the old ones.

Example execution

Execution result depends on how the P2P operation is implemented. Here is an example execution behavior.

When the first peer starts (after the server), it starts downloading blocks of image from the only peer (i.e., the server) available.	
When the second peer starts, it starts downloading from the server peer (the middle part) and the first peer (the top part). Notice that since the first peer is also downloading the blocks, the server is sending out blocks to the two peers in an alternating manner so that the first peer can download the other blocks from the second peer. The first peer will also start download from the second peer.	

When the user switched the image in the server, the peers download new blocks from the server and other peers.	

Marking

- **100% marks** are given to the **functionality**.
  - You may add additional classes, instant variables and methods to the project
  - You will get part of the full marks if you implement some of the features.
  - A program that can run normally without throwing exceptions during runtime gets higher marks.

Submission

Please submit all source files (`*.java`) in a single compressed file (in `.zip` or `.7z`) to Moodle. **Late submission is not allowed**.

Do not submit *.class* file.

Plagiarism

Do not attempt plagiarism. We will check your program with software that checks program structure. Both the source and the copying work will be penalized.

-- END --