R·I·T	Rochester Institute of Technology Golisano College of Computing and Information Sciences Department of Information Sciences & Technology

ISTE-121 Project Description
Image Processing Client/Server

Objectives:

Write a Multi-Threaded, Client/Server GUI that allows a client to send a color image file to the server and requests that the server return either a grayscale, negative, or sepia version of the image. See the images below for an example of each type of image (full size images are also included in the Project downloads on myCourses).

Original Color Taj Mahal image

Grayscale Taj Mahal image

Negative Taj Mahal image

Sepia Taj Mahal image

Client Requirements

The client GUI should be similar to the Homework 5 GUI, except that you will only need a "Send Image" button. When the button is clicked, a FileChooser should appear so that the user can select an image to be processed by the server. The user will also need a way to input whether they want the color image to be returned as grayscale, negative, or sepia. Once the user selects their image processing preference, the client will transfer the image to the server.

The server is only able to process 3 images at a time, so if the server is busy, the client should wait until the server is done processing an image before sending the image to the server.

Once the image is transferred to the server, the client will wait for the server to process the image, receive the image data once the server sends it, and save it to disk using the new image name that the server provides.

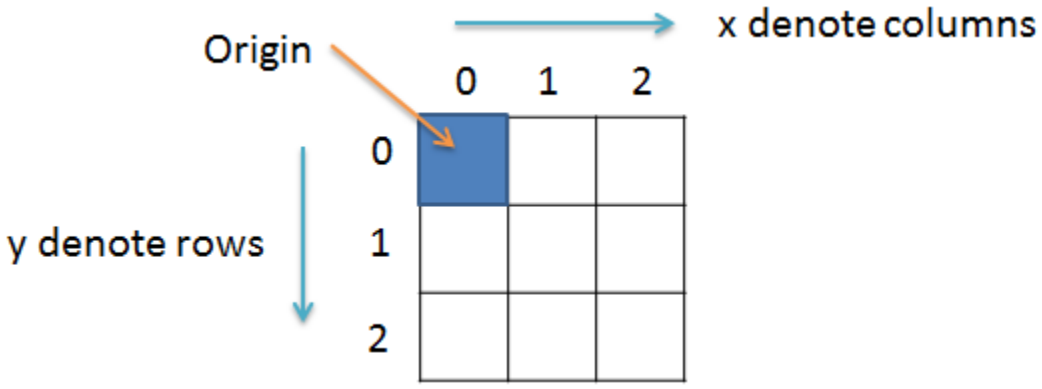
Server Requirements

The server will be multi-threaded, just like in Homework 5. It is suggested that you reuse as much of that code as possible as a starting point. When a client connects to the server, the client will send the name of the image to process, the type of image processing requested (grayscale, negative, or sepia), and transfer the image to the server.

Note that the server can only process 3 images at once, so if more clients are requesting image processing, they should be told to wait until server-side resources become available.

Server Image Processing

Upon receiving the image from the client, the server will determine the width (number of columns) and height (number of rows) of the image.



Next, the server will divide the image into six sectors, as shown below.

Sector 1	Sector 2	Sector 3
Sector 4	Sector 5	Sector 6

Each sector represents a portion of the pixels of the image and will be assigned to a thread for processing. Each of the six image processing threads will process their sector of the image and return it to the calling thread to be combined into a new image.

Since the width and height of the image may not be evenly divisible, Sectors 3 and 6 should be used to process any excess column pixels and Sectors 4, 5, and 6 should be used to process any excess row pixels.

When the processed image is ready for transfer, it should be renamed, depending upon the type of processing. For example, image.jpg should be named image_grayscale.jpg, image_negative.jpg, or image_sepia.jpg, depending upon if grayscale, negative, or sepia processing was done.

Communications Requirements

For client/server sockets, you may use either DataInputStream/DataOutputStream or ObjectInputStream/ObjectOutputStream. Regardless of the streams you use, make sure that the client and server logs have plenty of information in them for demonstration purposes!

Image Processing Requirements and Hints

You are required to use the ImageIO and BufferedImage Java libraries for your image processing tasks. ImageIO is used for reading and writing image files. You can assume that all input images will be JPG and you may output all images as JPG.

The BufferedImage library has methods that allow you to determine image width and height as well as getting and setting pixel values.

As shown above, images are divided into pixels, where the upper left corner is (0,0) and the lower right corner is (width-1, height-1). In general, a pixel is located at (x,y) and consists of 4 values: a, r, g, b where a is alpha (transparency), r is red level, g is green level, and b is blue level.

When getRGB is called from the BufferedImage library, it returns a 32-bit integer, where a, r, g, and b are the first, second, third, and fourth bytes, respectively, of the integer. To read each quantity, use the following code as a guideline.

```
int p = img.getRGB(x,y);
int a = (p >> 24) & 0xff;
int r = (p >> 16) & 0xff;
int g = (p >> 8) & 0xff;
int b = p & 0xff;
```

This code does right bit shifts and logical ANDs to mask out the rest of the 32 bits, so that only the 8 bits you care about are saved.

After processing the pixel values, to store the new alpha, red, green, and blue values back as a new pixel value in the image, use the following code as a guideline.

```
p_new = (a_new << 24) | (r_new << 16) | (g_new <<8) | b_new;
img.setRGB(x, y, p_new);
```

The first line left shifts the new values of alpha, red, green, and blue and logical ORs them together into a single integer that represents the new pixel value. This new pixel value is then written to the image.

Grayscale Image Processing

To convert a color image to grayscale, the red, green, and blue values of the pixel are averaged together and replaced by that average value. The alpha value is kept the same.

```
int avg = (r + g + b) / 3;
a_new = a;
r_new = avg;
```

```
g_new = avg;
b_new = avg;
```

Negative Image Processing

To get the negative of a color image, subtract the red, green, and blue values from 255. The alpha value is kept the same.

```
a_new = a;
r_new = 255 - r;
g_new = 255 - g;
b_new = 255 - b;
```

Sepia Image Processing

To convert a color image to sepia, use the following equations. The alpha value is kept the same.

```
a_new = a;
r_new = (int)(0.393*r + 0.769*g + 0.189*b);
g_new = (int)(0.349*r + 0.686*g + 0.168*b);
b_new = (int)(0.272*r + 0.534*g + 0.131*b);
```

It's important to note that if r_new, g_new, or b_new exceed 255, you must set them equal to 255 or there will be major problems with the pixel.

Project Deliverables

There are several deliverables throughout the project to both keep you on track and to give you experience with typical software practices in industry.

Project Plan/Design Review – presented in class on Friday 11/16

Your team will present the major tasks for the project and who you have assigned to complete these tasks as well as a class-level design that shows class member variables, methods, and interactions between the classes. This can be done in any way you want as long as the message gets across. Anything from PowerPoint slides to drawing on the board is fine. Everyone must participate in this presentation.

Code Review – presented in class on Wednesday 11/28

Your team will present a major component of your project code in terms of its design and intended functionality. You don't necessarily need to run the code, but you need to describe how and why it works. Everyone must participate in this presentation.

Project Presentation and Demo – during our final exam period on Thursday 12/13 (4:15-7:15 PM)

In this presentation, you will discuss the final design of your code, similar to the Design Review, and you will give a demonstration of your project. The demonstration will consist of showing that each test case is working properly. I will provide a set of test cases within two weeks of this deliverable.

Grading

The project is worth 25% of the course grade and will be graded as follows.

Grade Item	Weight
Project Plan/Design Review	20%
Code Review	10%
Presentation/Demo	10%
Project Performance	50%
Peer Review	10%

Besides the deliverables described above, the project performance and peer reviews will also be considered as part of the grade. The Project Performance is completely based upon your team's ability to successfully demonstrate each of the test cases I provide, while the Peer Review is used to ensure that each member has contributed to the success of the group. Extra credit on top of the value of the Peer Review may be awarded for exemplary contributions and additional reductions in the project grade beyond the 10% may be applied if a student demonstrates poor teamwork and/or effort.