PSLG Week 04

Ran by Amy and Ben

DLSH

ICTLC Online QR Code:

GitHub

Today’s agenda:

- Image Cropping.
- RGB manipulation.

Image Cropping

Cropping is the idea of taking a set portion of a 2-D grid and only representing that portion.

It’s achieved by creating a new BufferedImage of the size of the desired amount of cropping.

e.g. a 100 x 100 view of a certain part of an image.
You then define what section of the image you wish to crop and add the pixel values of the pixels within the cropping parameters in your BufferedImage

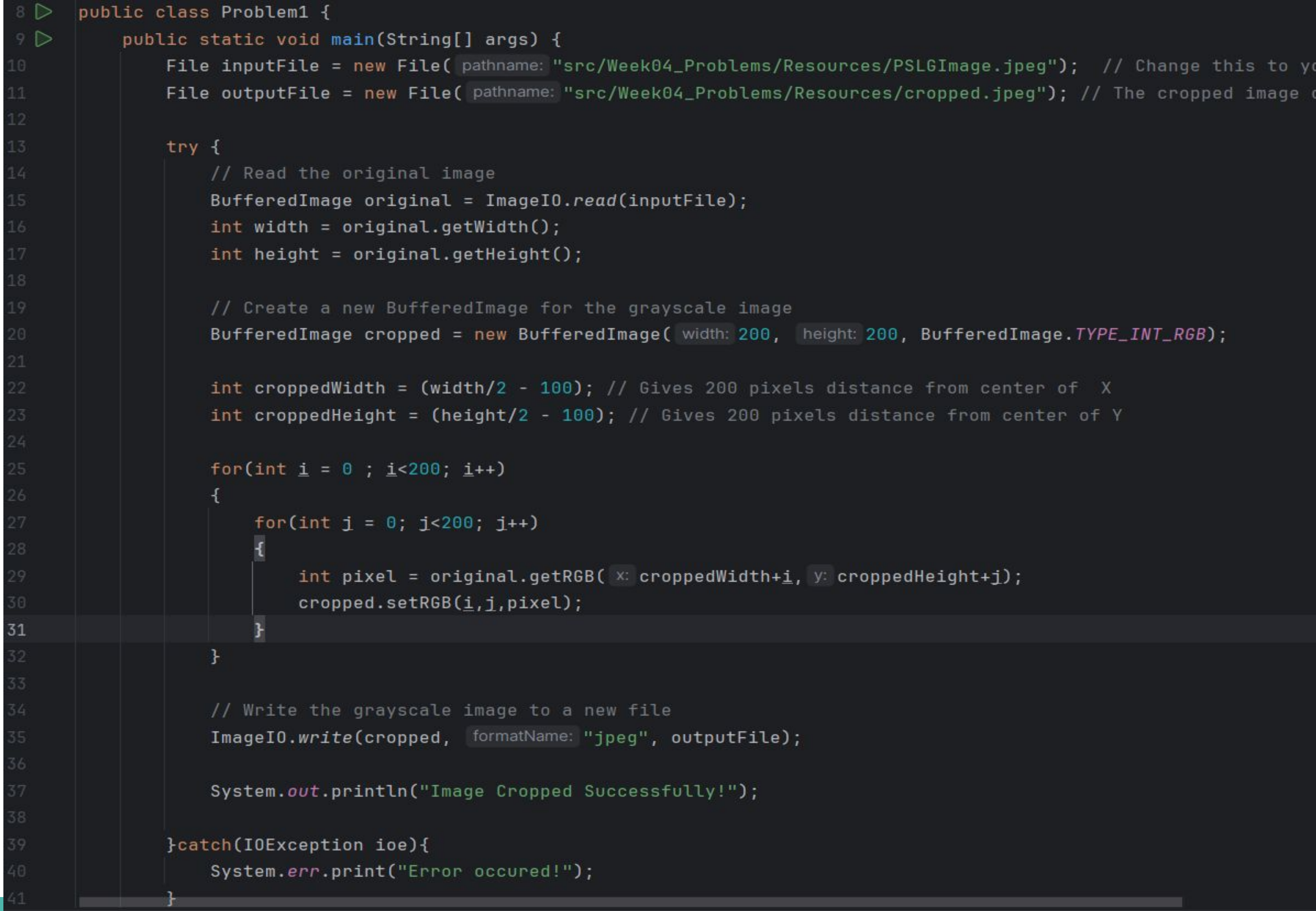
Cropping syntax

```
// Read the original image

BufferedImage original = ImageIO.read(inputFile);

int width = original.getWidth();

int height = original.getHeight();

// Create a new BufferedImage for the cropped image

BufferedImage cropped= new BufferedImage(X,Y,BufferedImage.TYPE_INT_RGB);

int croppedWidth = (width/2 - X/2 ); // Gives 200 pixels distance from center of  X

int croppedHeight = (height/2 - Y/2); // Gives 200 pixels distance from center of Y
```

Cropping problem

Create a java program where you take a desired image and crop it so that the center 200 x 200 pixels are what remain in the new output image

Solution



```
public class Problem1 {
    public static void main(String[] args) {
        File inputFile = new File( pathname: "src/Week04_Problems/Resources/PSLGImage.jpeg");  // Change this to yo
        File outputFile = new File( pathname: "src/Week04_Problems/Resources/cropped.jpeg"); // The cropped image

        try {
            // Read the original image
            BufferedImage original = ImageIO.read(inputFile);
            int width = original.getWidth();
            int height = original.getHeight();

            // Create a new BufferedImage for the grayscale image
            BufferedImage cropped = new BufferedImage( width: 200,  height: 200, BufferedImage.TYPE_INT_RGB);

            int croppedWidth = (width/2 - 100); // Gives 200 pixels distance from center of  X
            int croppedHeight = (height/2 - 100); // Gives 200 pixels distance from center of Y

            for(int i = 0 ; i<200; i++)
            {
                for(int j = 0; j<200; j++)
                {
                    int pixel = original.getRGB( x: croppedWidth+i,  y: croppedHeight+j);
                    cropped.setRGB(i,j,pixel);
                }
            }

            // Write the grayscale image to a new file
            ImageIO.write(cropped,  formatName: "jpeg", outputFile);

            System.out.println("Image Cropped Successfully!");

        }catch(IOException ioe){
            System.err.print("Error occured!");
        }
```

RGB Manipulation

Refresher for RGB values for anyone who might need it. The colour of a specific pixel in an image is determined by its RGB value, an RGB value is composed of 3, 8 bit values that represent the red,green,blue components of the pixel. this means the magnitude of a specific component can be between 0 - 256

For Red for example this would be from no red at all to the brightest red you could imagine. The combination of these components allows you to create most colours on the continuous colour spectrum.

RGB Manipulation (How this is done with code)

```
//In code we first need to extract the RGB value with the following lines

int pixel = image.getRGB();

int red = (pixel >> 16) & 0xFF;

// The >> are bit shifts and isolate each component based on their allocated   // bits e.g. here it shifts the component 16 bits to isolate the red component

int green = (pixel >> 8) & 0xFF;

int blue = pixel & 0xFF;
```

RGB Manipulation (How this is done with code)

```
// Next we manipulate the colour components to fit whatever theme we want

int red += 10; // Increases the red value by 10

int blue += 5; // Increases the blue value by 5

int green -=10 // Decreases the green by 10

// After that we rebuild the original component and add it to our image

pixel = (red >> 16) & 0xFF +(green >>8) & 0xFF + blue & 0xFF;

r.setRGB(pixel);
```

Problem 1

Based on the information given above write a java program that converts an image from a normal colour scheme to grayscale.

The relevant formulas are here:

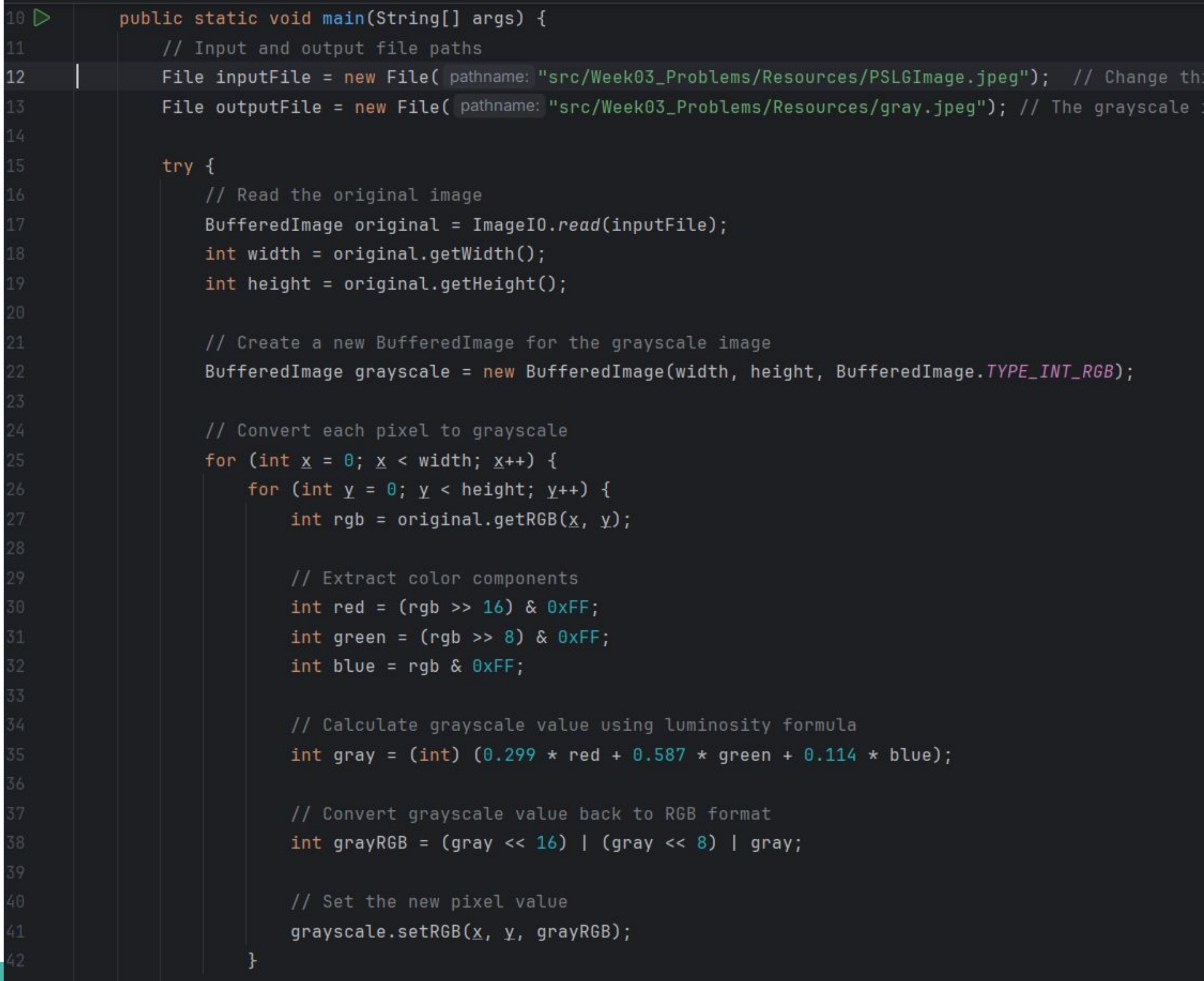
Luminosity = (int) (0.299*red + 0.587*green + 0.114 * blue);

When Rebuilding the pixel : (luminosity >> 16) | (luminosity >> 8) |luminosity;

Also when constructing the BufferedImage for output make sure you use the following declaration:

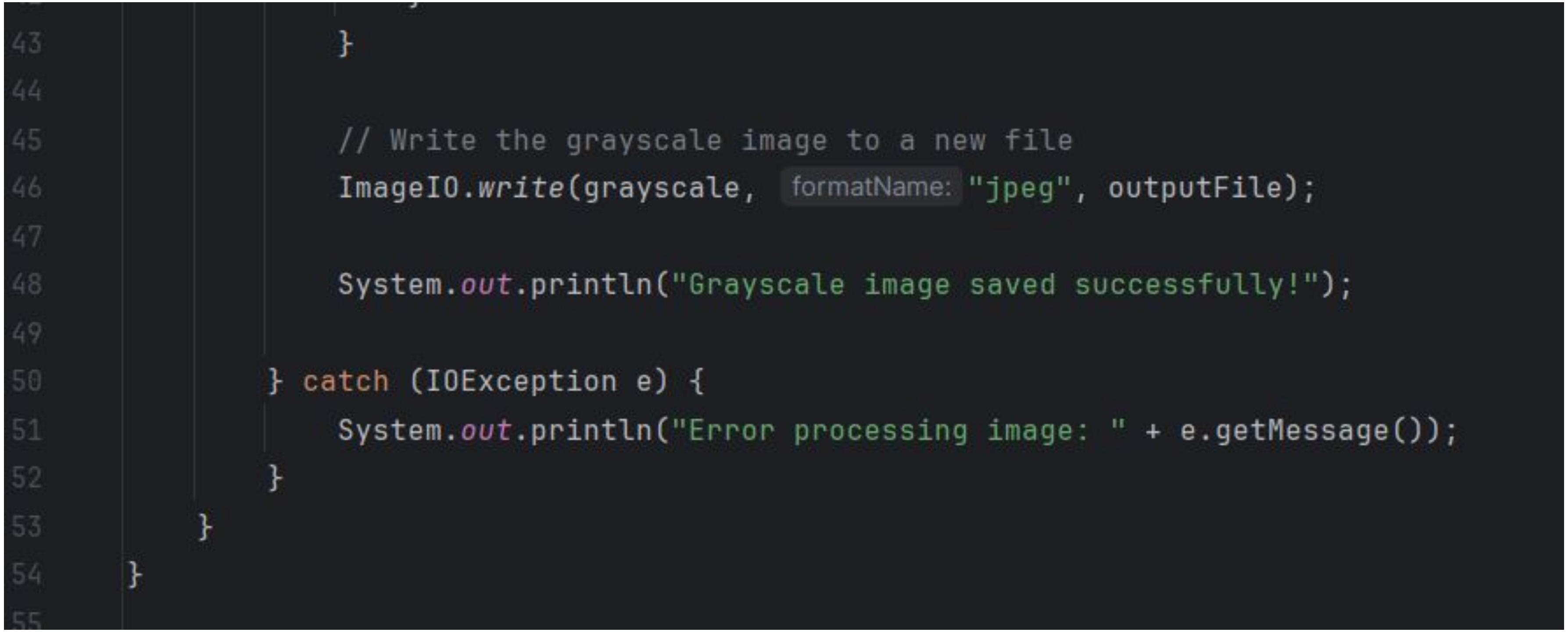
BufferedImage out = new BufferedImage(originalX,originalY,BufferedImage.TYPE_INT_RGB);

Solution



```
    public static void main(String[] args) {
        // Input and output file paths
        File inputFile = new File( pathname: "src/Week03_Problems/Resources/PSLGImage.jpeg");  // Change thi
        File outputFile = new File( pathname: "src/Week03_Problems/Resources/gray.jpeg"); // The grayscale i

        try {
            // Read the original image
            BufferedImage original = ImageIO.read(inputFile);
            int width = original.getWidth();
            int height = original.getHeight();

            // Create a new BufferedImage for the grayscale image
            BufferedImage grayscale = new BufferedImage(width, height, BufferedImage.TYPE_INT_RGB);

            // Convert each pixel to grayscale
            for (int x = 0; x < width; x++) {
                for (int y = 0; y < height; y++) {
                    int rgb = original.getRGB(x, y);

                    // Extract color components
                    int red = (rgb >> 16) & 0xFF;
                    int green = (rgb >> 8) & 0xFF;
                    int blue = rgb & 0xFF;

                    // Calculate grayscale value using luminosity formula
                    int gray = (int) (0.299 * red + 0.587 * green + 0.114 * blue);

                    // Convert grayscale value back to RGB format
                    int grayRGB = (gray << 16) | (gray << 8) | gray;

                    // Set the new pixel value
                    grayscale.setRGB(x, y, grayRGB);
                }
```

Solution



```
                }

                // Write the grayscale image to a new file
                ImageIO.write(grayscale,  formatName: "jpeg", outputFile);

                System.out.println("Grayscale image saved successfully!");

            } catch (IOException e) {
                System.out.println("Error processing image: " + e.getMessage());
            }
        }
    }
```