Partial_01 Report

The program that I developed for my first partial in the class of Multimedia and Computational Graphics was an image editing program developed in Java. This program allows the user to perform three different operations on a JPG image file. The image is first loaded into the program by using the buffered image function and then the main file provides a menu where the user has to choose between cropping a selected rectangular region, inverting the colors of a selected region or rotate a selected region by 90°, 180°, or 270°. The fourth option is when the user has finalized its wanted changes on the image and the modified image is saved as a new file. The user can apply these operations multiple times and in any order before saving the final m modified image which satisfies the requirements that were set for the assignment.

I designed the program using Object-Oriented Programming and not just simple hard coding as I have done other times, this was the requirement for the assignment. My program is divided into two classes which are ImageEditorApp and ImageEditor. The ImageEditorApp class contains the main method and is responsible for handling user interaction. It first creates an object based on the ImageEditor class. It then displays the main menu, reads user input like coordinates angles or names and according to the users input, it calls the corresponding methods from the ImageEditor class. It does not directly manipulate the image pixels.

The ImageEditor class contains the methods used to edit the image such as cropping, inverting colors, rotating and saving the modified image. It stores the current working image in a private BufferedImage variable so that nothing outside of the class can modify the image which is a good practice. This class contains a constructor that receives the path of the image file and

loads it using ImageIO.read(). All modifications are performed directly on this BufferedImage object.

The crop method works by determining the rectangle boundaries using the minimum and maximum values of the coordinates provided by the user. For example (200,200) and (400,400) 400 is the max and 200 is the min for both x and y. It then calculates the width and height of the selected region by subtracting max - min. This creates a new BufferedImage template with those dimensions and copies each pixel from the original image into the new one. This reduces the image to the selected region and updates its resolution.

The invert color method process was personally the most complicated, especially because I needed to research on details of how RGB works. Another rectangle region is selected the same way but this time, it retrieves the pixel's RGB value using getRGB(x, y), which returns a 32-bit integer containing the color information. Then I use a method that extracts the red, green, and blue components, extracting only the bits that correspond to that color. It then converts the bits into integers from 0 to 255 and the inversion formula is applied which is 255 - originalColor. After calculating the inverted values, it reconstructs the pixel to 32-bits integers and writes it back into the image using setRGB(). Since the program works with JPG images, the alpha component is not used. As we saw in class, JPG keeps the brightness of pixels constant.

The rotate method required coordinate transformation. First, it copies the selected region into a temporary image in order to have two pictures it can work with, one where it copies pixels and another one where it pastes rotated pixels. Depending on the selected angle, it creates a new image. For a 180-degree rotation, using a for cycle, each pixel at position (x, y) is moved to (w - 1 - x, h - 1 - y). This mirrors the image horizontally and vertically. For a 90-degree rotation, each

pixel is moved to (h - 1 - y, x) and for a 270-degree rotation, each pixel is moved to (y, w - 1 - x). On our modified image, the original selected region is filled with gray to ensure unused areas are visible. Then, the rotated image is drawn on top so that it gives priority to the rotated pixels.

The creation of this program demonstrates an understanding of pixel manipulation, coordinate systems, and the RGB system. It also represents OO programming because it separates files between user interface and image processing logic. Overall, I think that the program does a decent job for the given assignment. It has three different operations that can modify an image and generate a new image file as output. If I could make some improvements, I would make sure that a message written by me is thrown out in the console when the user inputs an incorrect value.