README.md

Updates for HW 7:

- Successfully implemented new mosaicking functionality
- Successfully supported a script command for mosaicking
- Successfully implemented mosaicking in the GUI
- Created a new Coord class to simplify dealing with positions of pixels
- Created a new Position class to represent the position of a pixel

We implemented a new Coord class which took our provider's implementation of a pixel array (which was represented as a list of RGB values similar to PPM's) and converted it into a 2D array list of pixel values. We did this to make calculating distances between pixels and seeds using the Euclidean distance formula much simpler and neater.

We also added a Position class which represented the positions of pixels in the Cartesian plane, which allowed us to quickly compare distances between pixels and the equality of pixels based on their location.

We believe that these classes greatly simplified the mosaicking implementation by allowing us to avoid a bunch of brute-force logic, and will also benefit the implementation of future features.

As far as implementing our actual mosaicking goes, we created a method that accepts a value that represents the amount of mosaic seeds and two random values. The random values will be used to find random x and y values within the bounds of our image for setting our seeds. Before we begin to create the given number of seeds, we first check that the requested value of seeds is not negative, and does not exceed the maximum number of pixels within our image. While creating the given number of seeds with random x and y values, we make sure to check that no two seeds share the same x and y positional values so no seeds overlap each other. Then, we loop through each pixel within our image and find its closest seed using a method that calculates the distance between two given positions. Then, we add that pixel to the arraylist of closest pixels to that seed. Next, we average out the red, green, and blue values for each pixel within that array list, and set each pixel within the arraylist of closest pixels to that seed to those averaged rgb values. Finally, we return the mutated image.

To test the new mosaicking functionality, we seeded two random values to pass in to the mosaicking command. Then, we assertEqual compared the individual red green blue values for each pixel in the mosaiced image to ensure that the mosaicking worked successfully. Next, we tested the exception cases where the user would try to set a negative amount of seeds or an amount of seeds that exceeds the number of pixels within the picture. Unfortunately, we were unable to test inputs for mosaicking command within the GUI because the GUI accepts user

inputs via a popup, which we could not manually set values for within a test class. However, we were able to test for errors with the mosaicking command within the terminal by referencing the String appendable, splitting it on its newlines, and then using assertEquals on the correct output which we found within the String array that came from splitting the String appendable of terminal outputs. We used this strategy to test whether the correct error statements were given in their respective situations.