So my project, which is yet to be named, is essentially a image processor. Like most image processors my project accepts an input, performs processes based on inputed parameters, and generates an output.

The inputs and outputs in this case are image files. In order to communicate with Java, some classes had to be imported. Some classes of critical nature include the File, ImageIO, and BufferedImage class.

As for the processes previously stated, they are contingent on inputs other than just the image itself. This system accepts multiple inputs. Before explaining the inputs it is important to note their value in terms of their role as arguments supplying the parameters of the processing methods. The stages of image processing fall chronologically in the following order: Greyscale Effects, Sepia Effects, Negative Effects, and Image Corruption. The three prior represent processing which alters the image file, while the last represents additive and subtractive processing with respect to image data.

Before moving to the inputs themselves, the structure and form which they take should be discussed. The input arguments are in the form of an object. This object is an instance of the Profile class. The profile class accepts a multitude of user parameters and creates an object which encompassed these parameters, able to return them when called in the main file. The input parameters are as follows:

- Input Path
- Amount Greyscale
- Max Cluster Size Greyscale
- Amount Sepia
- Max Cluster Size Sepia
- Amount Negative
- Max Cluster Size Negative
- Amount Additive Corruption
- Amount Subtractive Corruption
- Output Path

The object which houses these values also contains methods which return them to the main file. Once the object is created the main file begins its work.

First a new file object is created. This is based on the input path from the Profile object. This file is then read as an image by ImageIO. The image object is created based on the Buffered Image class. This class provides image height and width, as well as the ability to read and write pixel information.

The image object will then be manipulated using the effects, in the order previously established. Take Greyscale Effects for instance. The two arguments are Amount Greyscale and Max Cluster

Size Greyscale. Before moving on it is important to understand the definition of a cluster. Essentially a cluster is the portion of the image file being affected by that particular effect. Say one cluster run is performed on the image object, the only pixels altered would be the ones included between the cluster origin and the cluster length.

Each run is a loop of the following processes: a) creating random cluster origin point and random cluster length based on height and width, and b) looping through pixels using their x, y points, and lastly c) for each pixel in the cluster: read the pixel, process the pixel depending on the effect, and write the new pixel value to the image object.

The cluster size is used to determine cluster side length. This side length is determined by first taking the the difference between 10 and the inputed value, the max. This complement reflects the desired amount when the height is divided by it. For example if the user entered 2 for size, they intended for the cluster size to be rather small, given the height. So rather than dividing the height by 2 to find the maximum cluster length, 10 is subtracted by two. The result, 8 then divides the height. On every run the utilized side length is calculated to a random value between the origin point and the max cluster length. As for the origin point, that is a random value between 0 and the height. These processes are ran, each altering the image object. Upon completion a new file instance is created. The user is prompted to retrieve their file at the path specified as output in the Profile object.

TODO:

*Create a much more complete/accurate UML diagram.

*Create GUI to push arguments to profile constructor.

**Add GUI to UML diagram

**Create intuitive user manual after finalizing GUI and refactoring of processes

*Finish creating corruption methods. This will use a method which takes the image file pathname, change the extension to .txt. Next will add/delete data based on Profile object parameters. Then extension will be change back to original. Still have to figure out processing part (addition/deletion) but I have the extension changing method down.