Design/Model Changes.

Following are the changes that are made:

- Concept of Inheritance is used properly; all the operations (e.g. Filtering, Color Transformation, Color Reduce, Image Chunking and Image Pattern Generation) drive from common abstract AImageOperation.

- Create enums for:

  - Filtering type i.e. EFilterType (Blur Filter, Sharp Filter), in future if any new filter is introduce we can easily update our enum.

  - Color Transformation i.e. EColorTransformation (Grey Scale, Sepia Tone)

  - Chunking Image i.e. EChunkingStrategy (Mosaic, Pixelation)

  - Pattern Generation i.e. EPatternGenStrategy (Closest Color, Text Pattern)

- Created a separate interface for the functionality of controller i.e. IEnhancedImageBuilder..

  - EnhancedImageBuilder drives from IEnhancedImageBuilder.

  - MockEnhancedImageBuilder drives from IEnhancedImageBuilder.

  - EnhancedImageController composed from IEnhancedImageBuilder.

- Make the entire field private final.

- Created a new Abstract class ACommand; all the commands e.g:

  - BlurCommand

  - PatternCommand

  - PixelateCommand

  - MosaicCommand

  - SaveCommand

  - LoadCommand

  - SharpCommand

  - DitherCommand

  - GreyCommand

  - SepiaCommand

extends the common abstract ACommand.

- Use open close principle to avoid any modification in future and encourage extension.

Testing Plan

Mock model MockEnhancedImageBuilder drives from the interface i.e IEnhancedImageBuilder

The batch controller i.e. EnhancedImageController composed from IEnhancedImageBuilder Interface that performs image operations:

1- Image Filter
2- Color Transformation
3- Color Density
4- Image Mosaic
5- Image Pixelation
6- Pattern Generation

Following are the testing plan of the batch Controller

Filters are applied on two images:

Image 1: Lenna.png
Image 2: birds.png

Example Run:

```
upload Lenna.png
blur
save Lenna_blur.png
q
```

Note: q is 'Quit'

Input command	**Operations command**	**Output command**	**Quit command**
upload Lenna.png	blur	save Lenna_blur.png	q
upload Lenna.png	sharp	save Lenna_sharp.png	q
upload Lenna.png	sepia	save Lenna_sepia.png	q
upload Lenna.png	dither 2 essence	save Lenna_dither_2_essence.png	q
upload Lenna.png	mosaic 6600	save Lenna_mosaic_6600.png	q

upload birds.png	sepia sharp	save birds__sepia_sharp.png	q
upload birds.png	Mosaic 570	save birds_mosaic_570.png	q
upload birds.png	Mosaic 1650	save birds_mosaic_1650.png	q
upload birds.png	dither 2 essence	save birds_dither_2_essence.png	q
upload birds.png	blur	save birds_blur.png	q

Note:

Description of **Operations commands.**

- Filtering Images: A filter has a "kernel", which is a 2D array of numbers, having odd dimensions (3x3, 5x5, etc.). Given a pixel in the image and a channel, the result of the filter can be computed for that pixel and channel. My implementation design for filtering is flexible in a way, given that any predefine filter (as given below), application will compute corresponding pixel values.

  - Image 'blur'

  - Image 'sharp'

- Color transformations: Color transformation modifies the color of a pixel based on its own color. Color operations mentioned below are implemented as required.

  - Grey

  - Sepia

- Reduce color density: One of the ways in which we want to transform the colors in an image is to reduce the number of colors in the image. Currently, application provides easy way to generate output image either with or without essence for each 2 and 8 pixels. For the purpose of color reduction, I have implemented 'Floyd–Steinberg dithering' algorithm as it was discussed in pdf:

Dither 2 essence

Dither 8 essence

- Image Mosaic: The approach "chunking" that effect image into chunks, gives a mosaic effect.

Mosaic 1650

Mosaic 570

Mosaic 6600