Homework 2

Problems for all students

You must write a number of BufferedImageOp classes. Each class must implement the PluggableImageOp interface and make use of the PixelJelly.jar libraries. (In other words, each operation must load into PixelJelly as a plugin when the plugin folder is properly set).

1. ### FlipOps

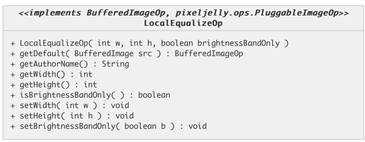
   Write classes named `HorizontalFlipOp`, `VerticalFlipOp` and `DiagonalFlipOp` that implement `BufferedImageOp` The operation will flip an image as described below.

   a. `HorizontalFlipOp` : Flips an image horizontally (around a vertical axis). This must support **in-place** operation.
   b. `VerticalFlipOp` : Flips an image vertically (around a horizontal axis). This must support **in-place** operation.
   c. `DiagonalFlipOp` : Flips an image around a diagonal axis. This could also be called something like a *transpose op* since the rows become columns and the columns become rows.

2. ### LocalEqualizeOp

   Write a class named `LocalEqualizeOp` that implements `BufferedImageOp` The operation performs local histogram equalization on an image.

   a. `LocalEqualizeOp` : Constructs an operation that uses a region of size (w x h) and operates either on the brightness band if `brightnessBandOnly` is true or on each band independently if `brightnessBandOnly` is false.
   b. `getDefault` : Returns an operation that uses a width of 5, height of 5 and operates only on the brightness band.

3. ### BandExtractOp

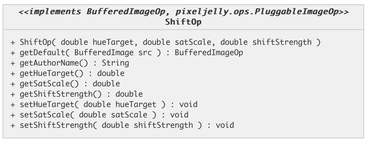
   Write a class named `BandExtractOp` that implements `BufferedImageOp` The operation will return an 8-bit grayscale image that represents a single band of some potentially multibanded image.

   a. `BandExtractOp` : Constructs an operation that extracts a single band from the src image (regardless of whether the input is grayscale, indexed or color). The band must be one of the values in { 'R', 'G', 'B', 'Y', 'I', 'Q', 'H', 'S', 'V' }. If the band is not one of these values, this method must throw an `IllegalArgumentException`.
   b. `getDefault` : Returns an operation that extracts the 'H' band.
   c. `setBand` : The band must be one of the values enumerated in the definition of the constructor. Throw an `IllegalArgumentException` if the band is invalid.

4. ### ShiftOp

   Write a class named `ShiftOp` that implements `BufferedImageOp` The operation will affect both the hue and saturation of any input image as described below.

   a. `ShiftOp` : Constructs an operation that affects both the hue and saturation of every pixel in the image. The value of hueTarget must be in [0, 1] and represents a normalized hue value (an angle). The value of satScale must be in [0, 5]. For every pixel P given as normalized <H,S,V>, the output pixel is given as < hShift(H, hueTarget), Y * satScale, V >. The hShift function accepts two hue values. The absolute angular difference between these hues, *dH*, is computed. This difference is such that two hues differing by 180 degrees are 1 unit apart; hence the difference between any two hues will always be in the interval [0, 1]. The function then returns *H moved closer to hueTarget* by the amount $dH^{shiftStrength}$.
   b. `getDefault` : Returns an operation uses a hueTarget of 0 and a satScale of 1.5.
   c. This operation must support in-place behavior

5. ### ColorHighlightOp

   Write a class named `ColorHighlightOp` that implements `BufferedImageOp` The operation takes a *target* color and *highlights* that color in the image.

   a. `ColorHighlightOp` : For every source pixel P given in normalized HSV coordinates, <H,S,V>, this operation computes the normalized L2 distance D between *targetColor* and P. The output pixel is given as <H, min(1, S * 1.1 * e^{-3*D}), V>
   b. `getDefault` : Returns a ColorHighlightOp with a target color of <220, 50, 50> (8-bit RGB).
   c. This operation must support in-place behavior

6. ### PosterizeOp

   Write a class named `PosterizeOp` that implements `BufferedImageOp` The operation creates an image having only the supported colors red, green, blue, cyan, magenta, yellow, black, and white.

   a. `PosterizeOp` : Constructs an operation that converts every pixel of the source to be one of the 8 supported colors. For every pixel P of the source, the output pixel is the supported color having the smallest L2 distance to P.
   b. `getDefault` : Returns a PosterOp.
   c. This operation must support in-place behavior

Additional Problems for Masters Students

7. ### FalseColorOp

   This operation accepts a palette and converts a grayscale image to the color scheme given by the palette. The resulting BufferedImage must use an indexed color model. The default palette should be of length 256 providing 3-bit coverage of the hue band, 2-bit coverage of the saturation band, and 3-bit coverage of the brightness (v) band. For every pixel P in the source image, the palette color is given by the brightness of P used as an index into the palette. The index must, of course, be normalized to the size of the palette.

   a. `FalseColorOp` : This constructor accepts a palette of colors. The palette must be of length 2 or greater and contain no null values. This method throws an `IllegalArgumentException` if the palette is invalid.
   b. `getDefault` : Returns a `FalseColorOp` that uses the default palette.

Additional Requirements

1. You must submit all your work using GitLab using a project named `cs454`. The written portions of this assignment must be included as a **PDF** document that is also uploaded to the GitLab repository.
2. You must place all code into a package named "hw2".

Test Samples

Expected output examples ☐

Appendix : pixeljelly.ops.PluggableImageOp

`PluggableImageOp` is an interface as shown below. The `getAuthorName` function should return your name. The `getDefault` function should return the default `BufferedImageOp` for the supplied src image.