Geometry, Dither, and Frequency

1. ## Most Common

   Write a class named `MostCommonOp` that filters an image by selecting the mode of each sample.

   ```
   <<implements BufferedImageOp, pixeljelly.ops.PluggableImageOp>>
                             MostCommonOp
   + MostCommonOp( int m, int n )
   + getDefault( BufferedImage src ) : BufferedImageOp
   + getAuthorName() : String
   + getM( ) : int
   + getN( ) : int
   ```

   a. `filter( BufferedImage src , BufferedImage dst)` . This method mode-filters the src.
   b. `getDefault()` : Returns a MostCommonOp using a 9x9 region.

2. ## DitherOp

   Write a class named `DitherOp` that color dithers a source image.

   ```
   <<implements BufferedImageOp, pixeljelly.ops.PluggableImageOp>>
                               DitherOp
   + enum Type { STUCKI, JARVIS, FLOYD-STEINBURG, SIERRA, SIERRA-2-4A }

   + DitherOp( type : DitherOp.Type, paletteSize : int )
   + DitherOp( type : DitherOp.Type, Color[] palette )
   + getDefault( BufferedImage src ) : BufferedImageOp
   + getAuthorName() : String
   + getType( ) : DitherOp.Type
   ```

   a. `filter( BufferedImage src , BufferedImage dst)` . This method applies dithering to a source image. The destination must use an `IndexedColorModel` . There are two constructors; one that accepts a palette and one that does not. If the palette is provided, the destination colors will be only those contained in the palette. The palette-based constructor will throw an exception if the palette is null or if any element is null. The non-palette constructor will generate an optimal palette (of length 256) for the source image using the median cut algorithm as a pre-processing step when filtering.
   b. `getDefault()` : Returns a DitherOp of JARVIS type with a palette size of 16.

3. ## FishLensOp

   Write a class named `FishLensOp` that creates a fisheye-lens effect.

   ```
   <<implements BufferedImageOp, pixeljelly.ops.PluggableImageOp>>
                   FishLensOp : GeometricTransformOp
   + FishLensOp( weight : double, isInverted : boolean )
   + getDefault( BufferedImage src ) : BufferedImageOp
   + getAuthorName() : String
   + getWeight() : double
   + getIsInverted() : boolean
   + setIsInverted( isInveted : boolean )
   + setWeight( weight : double )
   ```

   a. `getDefault()` : returns a `FishLensOp` using a weight of 5 and is not inverted.
   b. `filter` : Given a destination sample at location (x', y') we convert to polar coordinates (r', t'), given with respect to the image center (not the upper-left) of the destination image. We then compute the source location as (r, t) where t = t' and r is computed as shown below.

      - focalLength = max(width(src), height(src))
      - scale = focalLength / Log (weight * focalLength + 1)
      - r = r' if r >= focalLength
      - r = scale * Log(weight * r' + 1)   *if r < focalLength and isInverted*
      - r = $(e^{(r' / scale)} - 1)$ / weight   *otherwise*

4. ## DCT Compression

   Write a command-line **program** named `DCTCompressor` that is able to compress and un-compress an image. This program operates in one of two modes: *encode* and *decode*. When encoding, the program accepts a filename (along with other parameters) and encodes it. When decoding, the program accepts the name of an encoded image file (along with other parameters) and decodes it.

   ### DCT-File

   A compressed file is known as a **DCT-file**. This file is generated using the following process.

   a. Divide the image into 8x8 tiles. Some tiles (at the right and bottom) might require zero-padding.
   b. Zero-center each sample by subtracting 128.
   c. Compute the DCT coefficients for each band of each 8x8 tile using the shifted samples.
   d. Quantize using a reasonable quantization matrix.
   e. If N is is given, then retain only N of the DCT coefficients on each band of each tile. The N coefficients that are retained are the first N coefficients in a zig-zag scan of the tile. When saving coefficients, ensure that you don't store trailing zeros in the DCT coefficient stream.
   f. **Note** that the format of the file is not specified. You will, however, be graded based on file size. If you create a plain-text file encoding, you will recieve almost no credit for this problem.

   ### Command-Line Arguments

   DCTCompressor <mode> <input> [<N>] <output>

   - <mode> : either *encode* or *decode*.
   - <input> : the URL of an image if mode is encode or a filename if mode is decode.
   - <N> : an integer number between 1 and 64. N is only provided when in encode mode.
   - <output> : the name of a file. In encode mode, this will name the compressed file. In decode mode, this will name the output image file. The output image must be in PNG format.

Additional Requirements

1. You must submit all your work using GitLab using a project named `cs454`.
2. You must place all code into a folder named "hw4".
3. You must follow good SE practices