CSC 110: Fundamentals of Programming I
Assignment #6: 2D arrays

Due date: Friday Nov. 21, 8pm

How to hand in your work

Submit ImageProcessor.java through the Assignment #6 link on the CSC 110 conneX site. Please make sure you follow all the required steps for submission (including confirming your submission).

Learning outcomes

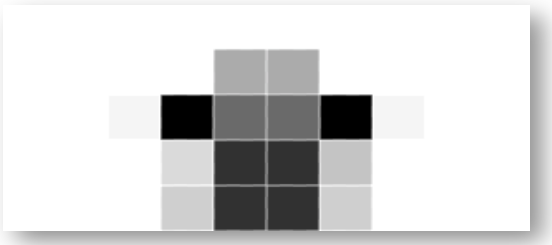
When you have completed this assignment, you should understand:

- How to create and work with *two-dimensional (i.e., 2D) arrays*.
- How to design code that is tolerant of *error cases*.

Problem description

One well-known set of tasks for computers today is the manipulation of images. Your work for this assignment is to create `ImageProcessor.java`. This will perform one image transformation (ASCII-art version of image) and several simple image manipulations (i.e., reflect across vertical axis; reflect across horizontal axis; tile image; brighten/darken image).

Images are often represented in programs as 2D arrays. If the image itself is a grayscale image (i.e., each image pixel represents some shade of gray, plus the possibility of white and black pixels), then we can encode shades as integer values in some 2D array. Consider the very small image below that is 10 pixels wide by five pixels high (`robbie_robot.jpg`):



This could be represented within a Java program (after loading in the JPEG image appropriately) as something equivalent to the following 2D integer array:

```
int[][] image = {
    {255, 255, 255, 255, 255, 254, 255, 255, 255, 255},
    {255, 255, 255, 255, 156, 156, 255, 255, 255, 255},
    {255, 254, 244, 0, 88, 88, 0, 244, 255, 255}.
    {255, 255, 255, 208, 39, 39, 184, 255, 255, 255},
    {254, 255, 254, 197, 40, 36, 197, 255, 255, 255}
};
```

where 0 is black, 255 is white, and grays range in-between. (For example, the top-most two gray squares have values 156.)

Required functionality of ImageProcessor

At runtime your program will be provided with command-line arguments indicating the image manipulation requested by the user.

Two methods are already provided to you: `readGrayscaleImage` takes a filename of a JPEG image as a parameter, and returns a 2D array; `writeGrayscaleImage` takes a filename and 2D array and saves a JPEG image. (The methods do not check if the image is a true grayscale; rather they read the blue-channel value for a pixel and use this as the luminance value for the pixel.) All files are JPEGs. ***Your solution must use two-dimensional integer arrays; you must not use instances of BufferedImage within your own methods. Do not modify the code provided to you in readGrayscaleImage() or writeGrayscaleImage().***

Functionality to be supported – your program must accept the following command line arguments (i.e. you get at the user input via the String[] args array):

- `-ascii <infile> <outfile>`

  The contents of the image in <infile> will be converted into an ASCII image and printed to a text file named <outfile>. Use the following grayscale-to-character mappings:
  - 0 to 25: 'M'
  - 26 to 50: '$'
  - 51 to 76: 'o'
  - 77 to 102: '|'
  - 103 to 127: '*'
  - 128 to 152: ':'
  - 153 to 178: '='
  - 179 to 204: '\'' (note the escape character before the single---quote)
  - 205 to 230: '.'
  - 231 to 255: ' ' (i.e., a single space)

- `-reflectV <infile> <outfile>`
  The contents of the image within <infile> will be reflected across the vertical

axis, and the results stored into <outfile>.

- `-reflectH <infile> <outfile>`

  The contents of the image within <infile> will be reflected across the horizontal axis, and the results stored into <outfile>.

- `-tile <num hz> <num vt> <infile> <outfile>`

  A new image called <outfile> will be created where the <infile> image is repeated <num hz> times (from left to right) and <num vt> times (from top to bottom). hz and vt are integers.

- `-adjustBrightness <amount> <infile> <outfile>`
  A new image called <outfile> will be created with the image file in <infile> either brightened or darkened. A brightened image will have each pixel value increased by <amount>. If <amount> is negative, the image will be darkened.
  Notes:
    - <amount> is an integer
    - grayscale values can only be between 0-255. If your changes result in values outside this range, you will need to replace those with the max (255) or min (0) value.

For example, if the user wants to read an image file named donkey.jpg and produce an ascii version of it (to be stored in a text file named newDonkey.txt), the user would run your program as follows:

```
java ImageProcessor –ascii donkey.jpg newDonkey.txt
```

More examples of what the user types to run the program (plus examples of the input and output) are given in the .zip file associated with this assignment.

ALSO: Your program must check for errors in the user input. I.e. if the command given by the user is not valid or any of the specific arguments given by the user don't match the expected format, you must print an appropriate error message.

What the marker will look for

- Compiles & runs on the lab machines.
- Implements the required functionality for image manipulation.
- Accepts the expected command line arguments.
- Handles input errors and prints a helpful error message.
- Is well-structured through the use of meaningful methods.
- Is appropriately documented and matches the style guidelines.

Files to submit: ImageProcessor.java via the Assignment #6 link on conneX along with all other files necessary to run your code.

Bonus (worth up to 10% if your basic assignment mark is at least 9/10)
For bonus points, enhance your ImageProcessor by adding one or more additional image manipulations. Examples:

- A blur function that blurs an image by replacing each pixel value with a weighted average of nearby values. Size of the blur kernel (i.e. how many levels of nearby pixels to consider) could be adjusted by the user.
- An addNoise function that adds noise to the image by randomly permuting the base image values (slightly).

Document your additions to ImageProcessor in a text file named **`bonus.txt`.** Explain what you have done and give instructions and examples of how to use your new functions. Be creative!