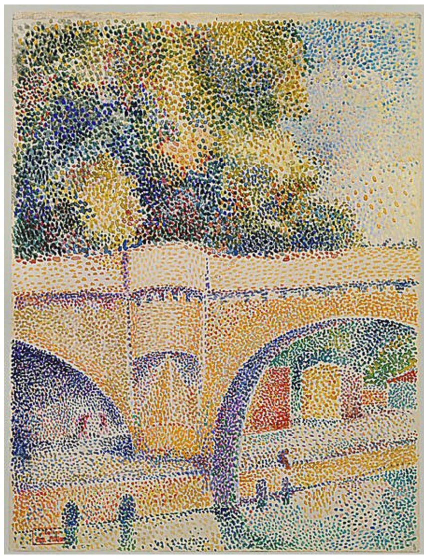
Project 10: Pixel Patterns

Due: Monday, Nov. 4 at 11:59 PM

Description: Electronic displays on computer monitors, televisions, and cell phones consist of grids of tiny, single-color squares called "pixels" (short for "picture elements").[1] Each pixel can be set to a different color, and images are produced by choosing an appropriate pattern. This technique for displaying images is similar to a mosaic or pointillist painting, like the one shown to the right.


Le Pont Neuf (c. 1912) by Hippolyte Petitjean

A common way to represent an image in a computer is with a two-dimensional array. Each element corresponds to a pixel, and the value of the element specifies a color. Three integers are needed to specify a single color in a computer, but if an image is in grayscale, only one integer per pixel is required. This means that a grayscale image can be represented as a two-dimensional integer array.

In this project, we will create four black-and-white images using nested loops and 2D integer arrays! The four images will be a pattern of vertical stripes, horizontal stripes, diagonal stripes, and a checkerboard.

You will write a method for each image that outputs a 2D integer array of image data. Each method will take the width and height of the image as parameters along with the stripe (or checker) width. Each of these lengths will be specified in units of pixels. You will also write a method that takes an array of image data as an argument and writes the data to a file. The resulting file can be opened with an image viewer.

Objectives: Your program will be graded according to the rubric below.

1. (15 points) Write the method saveImage. This method saves an array of image data to a PGM file with a given name.
2. (60 points; 15 per method) Write the methods createVerticalStripes, createHorizontalStripes, createCheckerboard, and createDiagonalStripes. Each method returns a 2D integer array of a given width and height that stores the pixel values for an image with one of the following patterns:
   - createVerticalStripes: a vertically striped image with a given stripe width
   - createHorizontalStripes: a horizontally striped image with a given stripe width
   - createCheckerboard: a checkerboard image with a given square width
   - createDiagonalStripes: a diagonally striped image with a given stripe width

[1] The resolution of a display is usually given as the size of its pixel grid. For example, a 1080p HDTV has a rectangular grid of pixels that is 1080 pixels tall and 1920 pixels wide. (The aspect ratio of most widescreen televisions is 16:9, so only the height needs to be given. The width is found by multiplying the height by 16/9.)

3. (15 points) Write the main method. This method calls the methods from objectives 1 and 2 to create four PGM image files, one for each pattern. Name the files “vertical-stripes.pgm”, “horizontal-stripes.pgm”, “checkerboard.pgm”, and “diagonal-stripes.pgm”. The width and height of each image should be 250 pixels, and the pattern widths should be 10 pixels.
4. (10 points) Use meaningful variable names, consistent indentation, and whitespace (blank lines and spaces) to make your code readable. Add comments where appropriate to explain the overall steps of your program.

Two-Dimensional Arrays: A one-dimensional array can be thought of as a list of elements labeled by an index. Below is an illustration of an array of length 5 with the indices shown as the contents:

0	1	2	3	4

A two-dimensional array can be thought of as a grid of elements labeled by two indices. One index specifies the row of the element; the other specifies the column. Below is an illustration of a 2D array with 3 rows and 5 columns. The indices are shown as the contents with the row index first and the column index second.

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4

The following line of code will construct this array and assign its address to the reference variable image:

```
int[][] image = new int[3][5];
```

Note that the data type of the reference requires two sets of brackets to indicate that it is a two-dimensional array. Similarly, the expression to construct the array has two sets of brackets. The number inside the first set specifies the number of rows; the number inside the second set specifies the number of columns.

When a 2D integer array is constructed, its elements are initialized to 0 (just like a 1D array). We can assign the value 42 to each element using nested for-loops:

```
for (int row = 0; row < image.length; ++row)
{
    for (int col = 0; col < image[0].length; ++col)
    {
        image[row][col] = 42;
    }
}
```

Note the following three things about this code:

1. To access a particular element, use two sets of brackets after the array reference. Put the row index inside the first set and the column index inside the second. For example, to access the element on row 1 and column 3, use the expression image[1][3].

2. The length field of the array gives the number of rows.
3. It is possible to access an entire row of the array by using a single set of brackets after the array reference. For example, image[0] is a reference to the first row. This can be useful for finding the number of columns, which is given by image[0].length.[2]

Portable Graymap (PGM) Files: To create our image files, we will use a format called "portable graymap." PGM files are simple text files with image data. Our files will have the following information:

```
P2
height width
255
image_data
```

The first and third lines specify that the format is ASCII grayscale and the maximum gray value is 255. Each file we create will have exactly this information on these lines. You do not need to change it.

The second line contains the number of rows and columns separated by a space. The remaining lines of the file after the third line contain the pixel values.

Each pixel value is an integer between 0 and 255. The larger the number, the brighter the pixel. A value of 0 is completely black, and a value of 255 is completely white.

Each row of pixel values should be on a single line separated by spaces. The number of rows should equal the height on the second line, and the number of values on each row should equal the width.

As an example, consider a file with the following information:

```
P2
3 8
255
0 0 255 255 0 0 255 255
0 0 255 255 0 0 255 255
0 0 255 255 0 0 255 255
```

This file stores the data for an image with vertical stripes that are two pixels wide. The image has two black stripes and two white stripes. The height of the image is three pixels.

PGM files can be opened with the default image viewer on macOS. If you are running Windows, you will need to download a compatible viewer. We recommend IrfanView, a tiny, fast, and free image viewer that can be downloaded here: https://www.irfanview.com/.

If you are unable to view your images, note that you can always open them as text files and view the data in the format shown above. This should give you enough information to check whether your code if working correctly.

[2] A two-dimensional array can be thought of as an array of arrays. The expression image[0] is a reference to a 1D array, which is why it has a length field.

Method Descriptions

1) `void saveImage(String filename, int[][] image)`

   Given a file name and a 2D array of integers, this method creates a PGM file with the pixel values stored in the array.

   - Use a PrintWriter object to create the file and write the image data. Refer to the Project 9 instructions for information on the related import statements and exceptions.[3]
   - PrintWriter objects have the methods println and print that work like System.out.println and System.out.print. Unlike the System.out methods, the PrintWriter methods write their arguments to a file, rather than the console.
   - Before you write the pixel values, make sure you write the header information described in the previous section (i.e., P2, height width, 255).
   - Remember that PGM files are just text files. When you call saveImage in the main method, you will give it a file name with a .pgm extension, but writing the data is exactly like writing words to the personal dictionary in Project 9.
   - Make sure you close the file after you write the data. Otherwise, the data will be lost.
   - Write and test this method before trying to write the methods that create the image data. It is much easier to debug the latter methods if you can view the output in a file.

2) `int[][] createVerticalStripes(int height, int width, int stripeWidth)`

   Given a height, width, and stripe width (all in units of pixels), this method returns a 2D integer array with the pixel values for a black-and-white vertically striped image.

   - For example, the method call createVerticalStripes(2, 12, 3) returns the following array:

     ```
     [[0, 0, 0, 255, 255, 255, 0, 0, 0, 255, 255, 255],
      [0, 0, 0, 255, 255, 255, 0, 0, 0, 255, 255, 255]]
     ```

   - In order to write this method, think carefully about how you would create a single row with alternating 0 and 255 values of a given width. You will need to use a for-loop to iterate over each element of the array. You will also need a way to keep track of when to change the color.
   - One way to track when to change the color is to use a count variable. Increment this variable on each iteration of the loop, and reset it when it grows larger than some appropriate value (that is related to the stripe width).
   - An alternative to using a count variable is to use the modulo operator. Suppose we have a for-loop with a variable idx that counts from 0 to 11. Note that if we calculate idx % 3 on each iteration, the resulting sequence of values is 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2.

[3] The Project 9 instructions include a line of code that constructs a PrintWriter using a File object as an argument. The PrintWriter class also has a constructor that takes just the file name as an argument:

```
PrintWriter writer = new PrintWriter(filename);
```

3) `int[][] createHorizontalStripes(int height, int width, int stripeWidth)`

   This method works like createVerticalStripes, but it returns an array with pixel values for a black-and-white horizontally striped image.

   - For example, createHorizontalStripes(4, 6, 1) returns the following array:

     ```
     [[0, 0, 0, 0, 0, 0],
      [255, 255, 255, 255, 255, 255],
      [0, 0, 0, 0, 0, 0],
      [255, 255, 255, 255, 255, 255]]
     ```

   - This method can be written by tweaking the code for createVerticalStripes. Rather than change the color after every stripeWidth columns, you need to change it after every stripeWidth rows.

4) `int[][] createCheckerboard(int height, int width, int squareWidth)`

   This method works like createVerticalStripes, but it returns an array with pixel values for a black-and-white checkerboard image.

   - For example, createCheckerboard(4, 4, 2) returns the following array:

     ```
     [[0, 0, 255, 255],
      [0, 0, 255, 255],
      [255, 255, 0, 0],
      [255, 255, 0, 0]]
     ```

   - Think of this method as a combination of createVerticalStripes and createHorizontalStripes. The loops will create a pattern like createVerticalStripes, but after every squareWidth rows, the starting color needs to change.

5) `int[][] createDiagonalStripes(int height, int width, int stripeWidth)`

   This method works like createVerticalStripes, but it returns an array with pixel values for a black-and-white diagonally striped image.

   - For example, createDiagonalStripes(4, 4, 2) returns the following array:

     ```
     [[0, 0, 255, 255],
      [0, 255, 255, 0],
      [255, 255, 0, 0],
      [255, 0, 0, 255]]
     ```

   - The trick to writing this method is noting that the sum of the row and column indices along each diagonal is constant. For instance, consider the following diagram of the previous array, which includes the row and column indices in the contents:

`0,0`	`0,1`	`0,2`	`0,3`
`1,0`	`1,1`	`1,2`	`1,3`
`2,0`	`2,1`	`2,2`	`2,3`
`3,0`	`3,1`	`3,2`	`3,3`

Follow the right half of the white stripe that starts at element (0,3) and moves down and to the left. The next element is (1,2), the next is (2,1), and the last is (3,0). The sum of the indices for each of these elements is 3, and they are all colored black.

- The code for this method will be very similar to createVerticalStripes, but the sum of the row and column indices will appear in the condition that changes the color, rather than just the column index.

6) `void main(String[] args)`

The main method calls the methods above to create four PGM files, one with each pattern. Set the width and height of each image to 250 pixels and the pattern width to 10 pixels. Name the files “vertical-stripes.pgm”, “horizontal-stripes.pgm”, “checkerboard.pgm”, and “diagonal-stripes.pgm”.

- Use smaller image sizes while testing your code. Wait to increase the width and height to 250 until you are confident that your methods are correct.

Submission Instructions: Submit your source code to Canvas in a .zip file. Below are instructions for creating this file in Eclipse.

1. Click “File” in the top-left corner and then “Export…” from the drop-down menu.
2. In the Export window, click the arrow next to “General,” then click “Archive File,” then click the “Next” button.
3. Click the arrow next to your project folder and then check the box next to the src folder.
4. In the “To archive file” field, browse to a convenient location to create the .zip file (e.g., your desktop) and name the file “Project10”. Click the “Finish” button.
5. Upload the .zip file to Canvas.