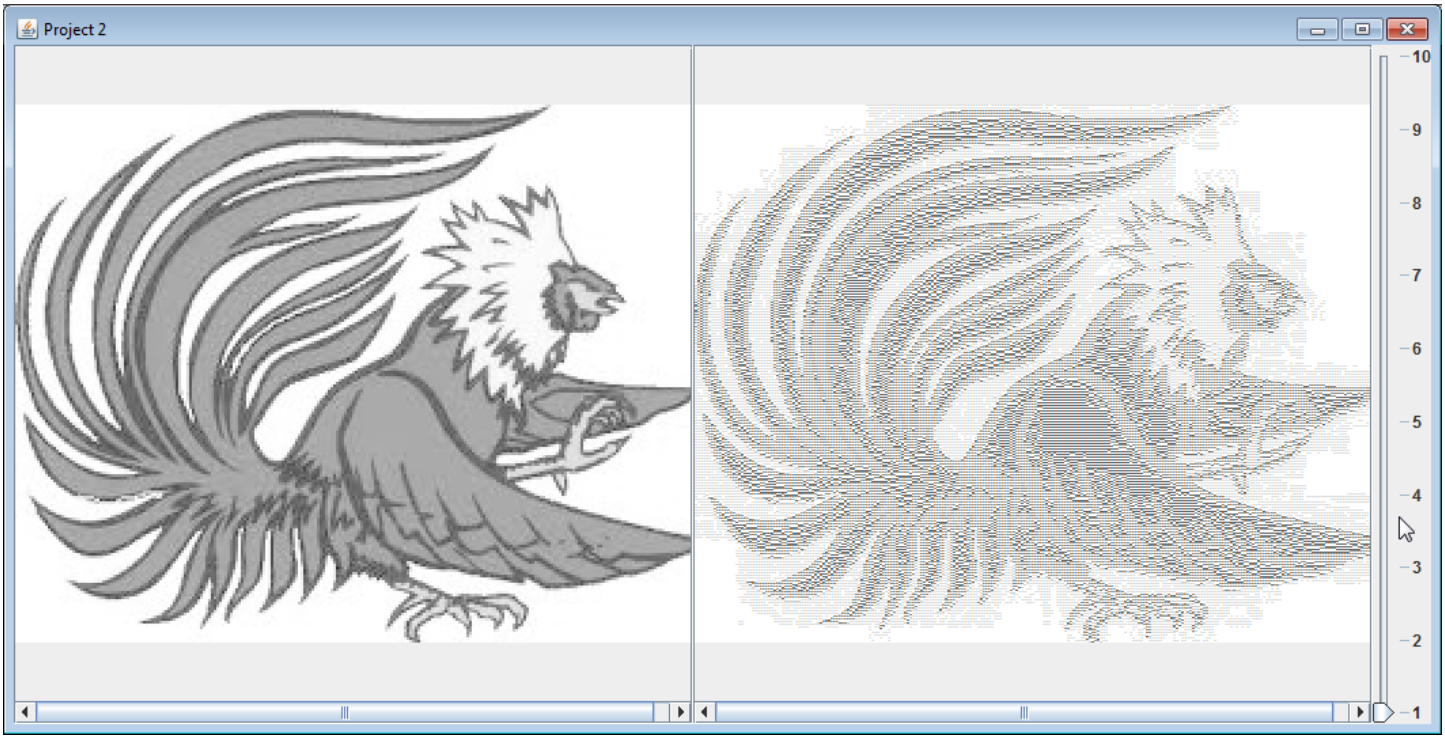
CS 232
Project 2
Image Manipulation

In this project, you must implement a very simple image manipulation program. The program should allow the user to specify an image through a command-line argument. That image should then be loaded into the left panel, and its ASCII version (explained below) should be loaded into the right panel. A vertical slider on the right-hand side should allow zooming on both images. The interface is illustrated below.



An ASCII image (the one on the right) is a rectangular grid of characters that reflect the brightness of the corresponding pixel in the original image. For instance, the “face” of the gamecock in the ASCII image above is actually this:

```
````````````.`.O-````````````````
````````````.``+@.````````````````
````````.+@+.`:@+````````` ````````
``````.+@$@@@-`O@-````````   ``````
``.`..O@$@$$$@O@$@O+.``` ````````````
`````.O$OoOO@@$$@$$#@-``` `````````
```` `-$@OooOO@#$OOO@$@:````````` ```
`````.:@$$OoooOOOooooO@@+.`````` `
````.``o$@OoOooOOOOOOOO@$@`````` `
````.``:$@$OO@$$$$@@OoooO@$O:``` ``
```.``.+$$@O@OOOOO@$$@OoooO@@-`````
``````o@@OooO+:-:-+o$@@OOOoO@$@-```
````.O@$OooOo+:::::-@$$@@OoOo@@`````
````o@$@$oOooOo::---OO@@@Oo@@@$.`````
```+@$@$@oOoO@@-:::::::--++oo@@$@.`````
```O$$$@OOoo@$$o:::::::::-Oo++-o@:`````
```:-@@$@ooo@$@@:::::::::o$@$@@@OO$`````
````-$$@$OoOO@$@-::::-:o$@$O``-+@@o` `   `
```.O@$$@oOO@$$$@--::::o$$@@-`````. ``  ``
```.$$$$@OOo@$$$$o:::::.-oO@$$O+````` `` ``
``..:o@$$$oooO$$@O@@o+::::::o@@O.```   ``
`````@@$#@OOoo$#@O$$$$@Oo+ooO$O.``` `` ``
```..@$$@$$OOO$$@O$$@@O@$@+o+:````` `` `
````-@o@@$$OO@$$OO$Oooo@$-`````````` `` `
````...@$@$$O@@$OoOoooo@$.`````````` ` ``
``````-$@$$$@$$$@ooOOoo@O`````` ``
``````-@@$@#$@$$@OO$@o@$o```` ```
``````:o@$@@$$$$$@$@$@$O````   ``
```````.:-.+$$$O@@@O@oo````   `````
```
```

There are four classes needed in this project. First, we need a class to represent an image, which should just be a rectangular grid of integers representing the grayscale pixel value. Next, we need a class to represent an ASCII image constructed from an actual image. Then we need a class for an image viewer (left-hand side of the interface above) and a class for an ASCII image viewer (right-hand side of the interface).

To assist with some of the more tricky parts of this assignment, a class called ImageUtilities will be provided for you that contains the following static methods:

```
public static Image loadJPEG(String filename) throws java.io.IOException
```
Returns an Image object created from the JPEG image specified by the filename.

```
public static void saveJPEG(Image image, String filename) throws java.io.IOException
```
Saves an Image object as a JPEG file with the specified filename.

```
public static javax.swing.ImageIcon createImageIcon(Image image, int zoomLevel)
```
Returns an ImageIcon (for display in a JLabel) based on the Image object at the specified zoom level (1 – 10).

```
public static java.awt.Font createFontForZoomLevel(int zoomLevel)
```
Returns a Font object based on the specified zoom level (1 – 10).

```
public static void resize(javax.swing.JTextArea textArea, AsciiImage ascii, int zoomLevel)
```
Resizes a JTextArea object based on the AsciiImage that it is displaying at the specified zoom level (1 – 10).

The specific requirements for each class are presented below. Each team must distribute these classes one per member to be implemented according to the given specifications. Each member will be scored individually based on his own implemented class, which must, at a minimum, pass all unit tests provided to be considered satisfactory. The individual performance represents 70% of the individual's grade, with the other 30% to be determined from the group submission.

Image

This class represents a grayscale image. It is just a rectangular array of integers representing the pixel values. Three image manipulation methods should be provided—shrink, invert, and mirror.

Public Attributes

`enum Axis {HORIZONTAL, VERTICAL}`
an enumeration representing the axis by which the image should be mirrored

Private Attributes

`int[][] pixel`
a two-dimensional array of integers representing the grayscale pixel values (0 – 255)

`int width`
`int height`
the width (number of columns) and height (number of rows) of the image

Public Methods

`Image(int width, int height)`
constructor that creates a fully black (all 0s) image of the specified width and height

`Image(Image image)`
constructor that creates a deep copy of the image passed to it

`int getWidth()`
`int getHeight()`
returns the width and height, respectively

`int getPixel(int row, int col)`
`void setPixel(int row, int col, int value)`
gets and sets the pixel at the specified row and column

`void shrink()`
halves the size of the image in both width and height by taking the average of the current pixel and its immediate east, southeast, and south neighbors and using that value as the pixel value for the smaller image

`void invert()`
creates the "negative" of the image by replacing each pixel value with its inverted value. For example a pixel value of 0 would become 255, 100 would become 155, and 200 would become 55.

`void mirror(Axis axis)`
"flips" the image about the specified axis. For instance, if the axis were vertical, then the first column and last column would be swapped, the second and next-to-last columns would be swapped, etc.

AsciiImage

This class represents an ASCII image, which is just a rectangular array of ASCII characters, each representing a different pixel intensity.

Private Attributes

`static final char[] shades = {'#', '$', '@', 'O', 'o', '+', '-', ':', '.', '`', ' '}`
static array of "pixel" values ranging from darkest to lightest

`char[][] pixel`
`int width`
`int height`
the array of character pixels, along with the width and height

Public Methods

`AsciiImage(int width, int height)`
constructor creates a new AsciiImage of the specified width and height filled with the darkest "shade" available

`AsciiImage(Image image, int maxDimension)`
constructor creates an AsciiImage object based on the specified Image, where the original image is scaled down to the maxDimension argument. For instance, if the image is 400x300 and the maxDimension is 350, then the image will be shrunk repeatedly until the largest dimension is less than or equal to 350. The AsciiImage is created from the Image by replacing each pixel of the image with the character in the shades array that is closest to its intensity.

`AsciiImage(Image image)`
constructor creates an AsciiImage object based on the specified Image. This constructor operates just like the one above except that resizing the image is unnecessary.

`int getWidth()`
`int getHeight()`
`char getPixel(int row, int col)`
`void setPixel(int row, int col, char value)`
getters and setters

`Image getImage()`
recreates an Image object based on the ASCII image. Note that this Image object will be lossy because of the very few pixel values available to an ASCII image.

`String toString()`
method that returns the ASCII image as a string, complete with new-line characters at the end of each line. If this string were printed to an output file, the resulting text would be the ASCII image. The StringBuilder class should be used here, rather than doing successive concatenations, for efficiency and speed.

ImageViewer (extends JPanel)

This class represents an image viewer that can be zoomed to a specified level. The viewer should be scrollable with a preferred size of 500-by-500 pixels.

Private Attributes

`Image image`
the image to be viewed

`JLabel label`
the label that will hold the ImageIcon created from the Image

`int zoomLevel`
the current zoom level (must be between 1 and 10, inclusive)

Public Methods

`ImageViewer(Image image)`
constructor that creates an ImageViewer from the specified image with an initial zoom level of 1

`void setImage(Image image)`
sets the current image of the viewer at the existing zoom level

`void setZoomLevel(int zoomLevel)`
sets the zoom level and zooms in on the image. The zoom level must be between 1 and 10, inclusive.

AsciiViewer (extends JPanel)

This class represents an ASCII image viewer that can be zoomed to a specified level. The viewer should be scrollable with a preferred size of 500-by-500 pixels. It should also be uneditable with a selection color of white. It will also need to make use of the GridBagLayout in order to center it correctly.

Private Attributes

`AsciiImage image`
the image to be viewed

`JTextArea textArea`
the text area that will hold the string representation of this AsciiImage

`int zoomLevel`
the current zoom level (must be between 1 and 10, inclusive)

Public Methods

`AsciiViewer(AsciiImage image)`
constructor that creates an AsciiViewer from the specified image with an initial zoom level of 1

`void setImage(AsciiImage image)`
sets the current image of the viewer at the existing zoom level

`void setZoomLevel(int zoomLevel)`
sets the zoom level and zooms in on the image. The zoom level must be between 1 and 10, inclusive.