CS 117: Introduction to Computer Science	**Winter 2006**
Handout 8: Problem Set #5 (Due: Monday, 27 February 2006, 11:59pm)	*22 February 2006*

Black Knight: *'Tis but a scratch.*
Arthur: *A scratch? Your arm's off!*
— from "Monty Python and the Holy Grail" (1975)

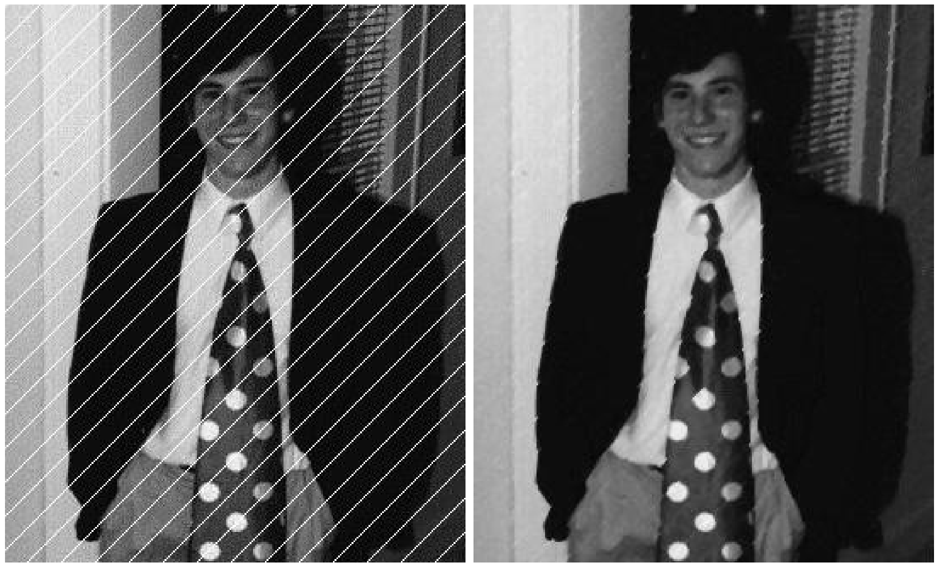
This is a group assignment, to be done in your newly assigned group. *Remember: I expect that all work will be done with both partners present!*

Photographs are often a mess. Sometimes there's a hair or a scratch on the negative when you print a photograph. Still photographs taken from television broadcasts or from worn video tapes are often filled with static. Even photos taken with digital cameras might show signs of scratches or smudges on the lens. There are many ways an image can accumulate pixels that show "noise" instead of the intended subject. In this assignment, you'll add functionality to your PhotoLab class to remove some of this noise.

One way to eliminate some kinds of noise is a *median filter*. The idea behind a median filter is that most pixels are surrounded by pixels of the same or similar color. If one pixel has an unexpected color (white for a scratch on the lens or black for dust on the negative, for example), the surrounding pixels are probably still okay. If we somehow let a pixel's neighbors "vote" on the pixel's proper color, we might be able to get rid of the noise. Here's one way to arrange such a vote. Start with a grayscale image. Cycle through all the pixels in the image with the usual double loop over the rows and columns of the image. For each pixel p:

- Collect the gray values for some the pixels in a neighborhood near p. The neighborhood could be a square of a particular radius, for example, where the radius-1 neighborhood would consist of p itself plus the eight surrounding pixels.
- Sort the gray values you have collected, smallest gray level to largest.
- Pick the middle gray value of your sorted list. The gray level of this pixel is the median gray in the neighborhood. Use that median value as the new color for p.

The left image below is a gray version of a now-familiar image, with some artificial scratching added to it. Using the median filter described above, with radius-1 square neighborhoods, we get the image on the right. At the cost of a bit of blurring and a bit of lost contrast, we have nearly eliminated the severe damage of the diagonal scratches.



Your task is to add two new methods to your PhotoLab class. If you have a new partner, you can start from the PhotoLab that either of you produced for the previous assignment, or you can start again from scratch. Here are the methods that your PhotoLab class should have:

- `public PhotoLab(EzImage image)`

  The constructor; the input parameter is the source image.

- `public EzImage scratch()`

  This method should create and return a new image with some artificial scratching of some kind performed on the original image. Diagonal white lines are absolutely fine, but feel free to be creative.

- `public EzImage medianFilter(int radius)`

  This method should create and return a new image that is the result of applying the median filter described above, using square neighborhoods of the specified radius, to the original image. A radius-1 neighborhood will be a 3-by-3 square, a radius-2 neighborhood will be 5-by-5, etc.

A few tips:

- Remember that you want to do this only for gray images, so convert any images that you use to gray before calling these methods.
- As in the `blur()` method in the last assignment, you'll have to be careful near the edges of the image. The neighborhoods of pixels near the borders of the image will not be squares. If you aren't careful, you'll end up with `ArrayOutOfBoundsException` errors.
- Some images respond better to the median filter than others. If you have a chance, play around with a variety of images, and try to figure out what kinds of images respond well or not so well to the median filter. Describe your conclusions in `ps5.txt`.
- Start early, ask questions if you have them, and have fun.