# CS 157 Lab 14 November 17, 2009
# Output Text File Processing and
# Introduction to Media Computation

Create a new class called **Lab14**.

1. Create a boolean method called *isPrime* that takes an integer argument and returns true if the argument is prime, false otherwise. Recall that a prime number is one that has only itself and one as factors. (Yes, we've done this problem before). Create enough JUnit test cases so that you're confident that it works. Make sure that it knows that 2 is prime, and 1 is not.

2. Create a File variable for a new output file to be called "primes.txt". Write a main method that uses a PrintStream object to put into this file all of the prime numbers less than 100,000, one per line. When you run your program, you should find the file was created in the same directory where the Netbeans project is located. After you run your program, comment out all of the lines in the main method that constructed this file.

3. Using a Scanner object, open the file for input and find out how many prime numbers are less than 100,000 (the same as the number of lines in the file). To do this, create a method called *countLines* that returns the number of lines in the file. Put all of the file related stuff in this method, and call it from main to get the answer: number of primes less than 100,000: _____.

4. Write a static method named *primesBetween* that takes two integer arguments and returns the number of primes between the two arguments (inclusive). For example, *primesBetween(1, 100)* should return 25. You can also check your answer to number 3 using this method. Put all file related stuff into this method (so the file will be reopened each time the method is called). Rerun your program to determine the number of primes between 25,000 and 25,100. After completing this, make sure all running programs are stopped before continuing.

5. Create a new project called **Media Computation**. (Spaces are allowed in project names) Right click on **Source Packages**, and add a new Java package called **mediacomp**. Copy all of the files found in my **mediacomp** directory to this one. Perhaps the best way to do this is to open a terminal window, navigate down to the **mediacomp** directory you just created using the Unix **cd** command together with **ls**. Then check and make sure you're in the proper directory by typing **pwd**. You should see the name of the current directory, and it should end in **mediacomp** if you're in the right place. Then carefully type the following Unix command: `cp ~jcaristi/mediacomp/* .`

   Make sure you type the period character at the end. After doing that, go back into your Netbeans window and wait for all the error indicators to go away. That may take some time. You may want to ask one of us for help.

6. In the class with a main method create a String variable called **fileName**, and in it put what is returned by the method of the class **FileChooser** called **pickAFile**. You will probably need to import the class from the mediacomp package. Netbeans can do that for you. Try running this file. You will get a window appearing that will ask you to pick a file and allow you to browse for it. Whatever file you pick, its name will be returned. Once you get this to work, find a picture file and put it somewhere where you can find it. If you don't want to download one from the web or use your own, I have a file you can copy from `~jcaristi/xu.jpg` Run your program and select this file with the pickAFile method.

7. Add a Picture object as follows: `Picture p1 = new Picture(fileName);` To see if this all worked, add the following line: `p1.show();` What this does is create a Picture object using the file you picked, and then sending it a message to have it display itself on the screen.

8. Pictures are made up of dots of color called "pixels" (picture elements). Pixels are numbered with two coordinates starting at the upper left corner of the picture, which is location (0, 0). The Pixel at location (4, 7) would be in the $5^{th}$ row down and the $8^{th}$ column to the right. There is a Pixel class in the mediacomp package. You can send a message to a Picture object to ask for any of its Pixels. But first you need to get the height and width of the Picture. Each Picture object has getWidth and getHeight methods that return the values as integers. Store them and print them out just to see what you're dealing with.

9. Create nested loops that run loop variables ***i*** and ***j*** starting at zero and going up to the height and width (but not equal to height and width). Inside the inner loop you can ask the Picture object for the Pixel at location (i, j) by using the **getPixel** method. You can store that Pixel in a Pixel object if you wish. You can ask a Pixel object for its Color with the **getColor** method. You can also set a Pixel's color with the **setColor** method. There is a Color class that contains most of the useful colors as easy to remember constants (for example, Color.YELLOW). Try turning all pixels on the diagonal (where ***i*** is equal to ***j***) to black. After your loop has changed the color of the pixels on the diagonal, tell the Picture object to show itself again. You should be able to see a thin black line from the upper left corner going diagonally down.

10. Try turning the picture upside down. This is a little tricky. To do it, start by creating a second Picture object called **p2** as follows: `Picture p2 = new Picture(width, height);` where width and height are the values obtained from the Picture **p1**. Set up nested loops as before. Get and store the Pixel from **p1** at location ***(i, j)***. Get and store the Pixel from **p2** at location ***(i, height-j-1)***. Tell the Pixel from **p2** to set its Color to the Color of the Pixel from Picture **p1** (the two Pixels are located on opposite sides of the Picture, vertically). At the end of the loops, tell **p2** to show itself.