**CSc 110 Lab 6**

**Objectives:**

1. Processing data from a file: text and bitmap image
2. More methods
3. Group work (~3 students) for reading images and colour testing
4. Conditionals and Loops (While-Loops), if there is time

**Review**

Possible solution for the Guesser program from Lab 5: [Guesser.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/Guesser.java)

**EXERCISES**

**Exercise 1 File Data**

a) This exercise will have you code a program with a single method, "demo", that given a Scanner variable, reads and echoes several pieces of data from the input (keyboard).  To reinforce the practice you are given this program shell: [Ex1.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/Ex1.java) (also pictured below) to get you past declaring the Scanner and invoking the method.  Implement, test, compile and run what you see below first, then write the body of the method as it is described in the documentation.

Test the following input or any string and three doubles:

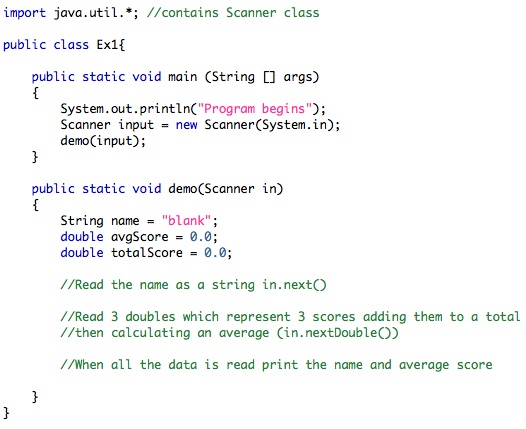
toad

10.1

20.2

30.3

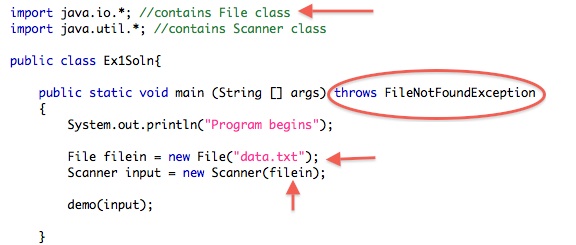
And expect the following to be printed: "toad 20.2"



b) Run your program again but this time enter the data all in one line:

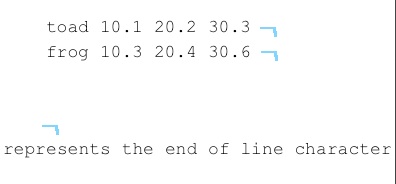
    toad 10.1 20.2 30.3

Your result should be the same.  That is because the keyboard is just a special case of a file.  The file's name is System.in.  
  
c) Make the following additions/changes to your program:



* The java.io library is required for a File definition.  System.in is a default file and did not require the explicit inclusion of the io library.
* Since defining a file handle to reference a data file that may not be present, we must indicate what main should do suppose a FileNotFoundException error is thrown.  The compiler will catch this if you leave it out and tell you that a FileNotFoundException must be caught or thrown, so we throw it.
* We declare filein and we set Scanner's input to be from that file.
* We create the file data.txt that has the line toad 10.1 20.2 30.3 in it.
* Method demo requires no changes.  The program should run as before.

d) Make your program handle more than one line of data by adding a loop.

suppose your data is as follows:  
  
  
  
The first string read ( next() ) gets toad, each read for a double ( nextDouble() ) gets a double. The file pointer moves to the next token each read and is pointing at the string value frog.  
  
Your TA will walk the data reads for you to make it plain what is happening and where the file pointer is each time.

**Exercise 2**

a) Make a copy of this program (Ex1.java), named Ex2.java but instead of one method reading all the data make three methods:

1. The first gets the name (a String), then passes the name and the Scanner variable to a second method
2. The second method gets and totals the scores (doubles). It passes the total to a third method. It also prints the name and average (returned by the third method).
3. The third method takes in the total (a double). It calculates and returns the average to the second method which prints.

Your output is exactly the same as before.  Loop to read in the whole data file (in the main method). Consider using a [Scanner method](http://download.oracle.com/javase/1,5,0/docs/api/java/util/Scanner.html#method_summary) for the condition of your loop.  
  
This is an exercise to make you comfortable passing the Scanner around which maintains its pointer into the data.  Your assignments will be much easier if you are comfortable passing the Scanner.   
  
Also you should be becoming a little more comfortable with the idea that you can create methods in any order that you wish and employ them later as part of a larger program.

**Exercise 3 Writing code in Groups: Colours in Computer Images**

1. The TA will touch on these java objects:
   * [File](http://docs.oracle.com/javase/1.4.2/docs/api/java/io/File.html)
   * [IOException](http://docs.oracle.com/javase/1.4.2/docs/api/java/io/IOException.html)
   * [ImageIO](http://docs.oracle.com/javase/1.4.2/docs/api/javax/imageio/ImageIO.html)
   * [BufferedImage](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/image/BufferedImage.html)
   * [Color](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Color.html)
2. Specifically the following methods:
   * [read in ImageIO](http://docs.oracle.com/javase/1.4.2/docs/api/javax/imageio/ImageIO.html#read%28java.io.File%29)
   * [getRGB in BufferedImage](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/image/BufferedImage.html#getRGB%28int,%20int%29)
   * [getRed, getBlue and getGreen in Color](http://docs.oracle.com/javase/1.4.2/docs/api/java/awt/Color.html#getBlue%28%29)
3. Then several groups will form to write a java class to check for red, green and blue in the bitmap samples: [white.bmp](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/white.bmp), [black.bmp](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/black.bmp), [red.bmp](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/red.bmp), [green.bmp](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/green.bmp),[blue.bmp](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/blue.bmp) and [yellow.bmp](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/yellow.bmp).

**RGB Colour model**

See [wikipedia's explanation](http://en.wikipedia.org/wiki/RGB_color_model) if you're interested.

**Process**

The shell for the class is already written, with the necessary imports: [Colours.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/Colours.java).  
Each group should:

1. Understand what Colours.java must do, what files it will use (where must they be?), and what should be printed
2. Design a solution using the methods above
3. Test with all the suggested inputs, checking for correctness
4. Be prepared to explain their final, working solution to the class (on the white board)

The groups will present their solutions. Be sure to pay attention and ask questions of the other groups so that you understand the different ways to solve the problem, what works, what doesn't work and why.

**Exercise 4- Another File Example: simple webpage formatting**

Webpages use HTML (hyper text mark-up language) for formatting. Save this file: [horoscopes.html](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/horoscopes.html). It may be readable in a text editor, but it will not be a nicely formatted webpage when you open it with a web browser. Not until you put the HTML newline tag after each line: <br/>

For large files, this would be a tedious process by hand, but with a computer program to do the processing, it only takes a few seconds: [HTMLFileProcess.java](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/HTMLFileProcess.java). This program reads in the horoscope file with Scanner, like in Exercise 1, then after each line it prints an HTML line break. Copy the result (right click > Mark or Select All > select the text > press Enter > now you can paste) into a new file: scopes.html, and see the difference.

**Exercise 5- (time permitting) Conditionals and loops.**

a) Given the grid image and the code snippet complete the following instructions.  The TA will get you started and circulate among you to see how you are doing.

|  |  |
| --- | --- |
| http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/images/grid.jpg | Here is an image describing a couple of  points on a grid where the first (x1,y1) is (1,2) and the second  (x2,y2) is point (3,4)  We want to create a method move that when passed x1, y1, x2, y2 will describe the moves necessary to get from the first point to the second.  If we regard x in the positive direction as East and y in the positive direction as North, the method passed x1, y1, x2, y2 should print: Go East Go East Go North Go North  **or any combination of grid moves that describes the movement**. |

You have [this code](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/Navigation.java) to get you started.  Try testing it with different values in the main method, including the values shown above.  
  
Note this method will only deal with one East/West move and one North/South move. These are if-statements and not loops.  
  
There are many ways that you can integrate while-loops to repeat the steps to get from point 1 to point 2.

b) Add in repetition so that all the necessary moves are made until start x,y reaches end x,y.  After succeeding with the first call try other values:  
This is where you can use a paper grid so you can see if your code is correct.

3, 4, 1, 2  
-1, -2, 3, 4  
3, 4,  -1, -2  
  
Don't forget no movement:  
  
1, 2, 1, 2

**Looking back...**

Make sure you had a chance to look over everything in Lab 6 and also last lab. You can always go back over old labs for more practice and examples.

In this lab you'll have practiced and/or seen examples of:

* Reading text input from a file
* More passing Scanner and other parameters
* More return statements in methods
* While-loops (see some different solutions to the Navigation exercise [here](http://webhome.csc.uvic.ca/~csc110l/2012_5/Lab6/code/NavigationDone.java))
* Reading image information from a bitmap file

**The End**