School of
Engineering and Computer Science
Te Kura Mātai Pūkaha, Pūrorohiko

# Lab Exercise for Assignment 4

## Intro to Programming: Lab Exercise for Assignment 4

- Due Nov 24 23:30

### Goals

This assignment will give you experience in writing programs that get input from files and write output to files. It will also give you more practice using conditionals and loops.

### Resources and links

- Copy the files to your USB stick or Download Zip file. Please note that the zip file also contains code and data for the assignment.
- Description of the Assignment

## Summary

Programs with files, loops, and conditions
- Test Practice:
  ➡ a test question from a previous year
- Image exercise:
  ➡ Two little programs to draw grid of random color squares and a gradient.
- Files exercise:
  ➡ Series of programs to read data from files and print/draw/summarise the results.

### Hints about Colours:

You can use `UI.setColor(new Color(r, g, b))` to set the color of the UI to a new colour if `r`, `g`, and `b` are integers containing the red, green, and blue component values. `r`, `g`, and `b` must be between 0 and 255 (inclusive).

To obtain a random colour, you can use the `Math.random` method:

```java
int r = (int)(Math.random()*256);
int g = (int)(Math.random()*256);
int b = (int)(Math.random()*256);
Color color = new Color(r, g, b);
```

## Test Practice

Suppose the file called `SomeText.txt` contains the following:

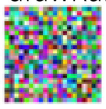
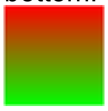
```
1003 laptop 1400
1009 disk 279
1021 laser 1899
1017 printer 630
```

What will the following `filePrint` method print out? Hint: keep track of the values of all the variables

```
public void filePrint (){
    try{
        Scanner scan = new Scanner (new File("SomeText.txt"));
        int max = 100;
        while ( scan.hasNext() ) {
            int x = scan.nextInt ();
            String token = scan.next ();
            int y = scan.nextInt ();
            if (x > y ) {
                max = y;
            }
            UI. println (max + " for " + token + " : " + x);
        }
        UI. println ("done: " + max);
        scan.close ();
    }
    catch(IOException e){UI.println("Fail: " + e);}
}
```

## Exercise Series 1: Drawing images

The first two exercises are in the ImageExercise class. You are to complete two methods for drawing images. Both methods involve a nested loop.

- drawRandomImage draws an image consisting of squares of random colours.
  
- drawGradient draws a square image that is a smooth gradient from red at the top to green at the bottom:
  

## Exercise Series 2: Reading files

The second series of exercises is in the FilesExercise class. It contain seven methods that each involve reading some data from a file and doing something with the data. You should complete the first method (printFile) and then complete methods until you have written at least **two** by yourself. Because each exercise addresses different kinds of file structure, you may find it helpful to do all the methods.

- sumNumbers should read numbers from a file called "numbers.txt" which contains only numbers. It should add all the numbers up and print the total at the end. (The total should be 2174.)

- drawCircles should read a file called "circle-positions.txt" containing pairs of numbers. It should draw a circle of radius 20 at each position that it reads from the file. For example, if a line of the file has 120 350 on it, then it should draw a circle centered at (120, 350). You can make them all the same colour or random colours.

- drawNames should read a file of names called "names.txt" and draw them on the graphics pane. Each line of the file has two numbers (x and y), a one word name, followed by three integers specifying the color (red, green, blue). For each line, it sets the color of the UI, and then draws the name at the position. Note, it is important to read the colour components as integers.

- firstWords should read a file called "text.txt" and print out just the first token (word) on each line. You should use next() to read the first word, and nextLine() to read the rest of the characters on the line.

- longestWord should read all the words in a file called "text.txt". At the end, it should print out the longest word. You will need to use the length() method which can be called on Strings to find the length of each word that you read.

- largeCourses should read data from a file called "course-counts.txt" that has a coursecode and a course size (the number of students) on each line. The method should count (and print out) how many of the courses have more than 100 students. (The answer should be 10.)

- `totalOrder` should read data from a file called "order.txt" that has an item, a count and a unit-price on each line. The method should calculate and print the total cost of the order.