HW3

Please complete the coding exercise (just one!) below, and submit your source code on D2L as usual.

Q. Color strips creation (15 points)

Let's start with the result - your program, made up of two .java files, should produce an output like so [your actual colors will vary, compared to the ones below]:

Ignoring the white regions at the top and bottom (they are not part of what we create), what you are looking at are color strips (256 thin, colored rectangles) one below another, each of dimension 100x4 (width 100 pixels, height 4 pixels).

What is the point of learning to create this? Well, you'd be using such a color strips array in your upcoming (soon) HW4, where you'd create a Mandelbrot fractal (like in HW2), except that instead of coloring pixels using gray values from 0 to 255, you'd use your 0..255 gray values (from HW2) as ARRAY INDICES into a 0..255 color strips list you are creating in this HW (in other words, 'integer in, color out'), and use the resulting colors for your fractal's pixels - that would make your fractal look gorgeously colorful.

To create and display the color strips array, you need to code two Java source files: ColorList.java, and ColorStrips.java. The ColorList program would create the colors, not display them; conversely the ColorStrips program would display the colors, not create them. So we need the contents of

both the files. It is good design practice to keep content ("model", ie. data) SEPARATE from presentation ("view", ie. UI), that is why we are creating two .java files instead of just one.

Make sure both .java files are in the same directory. drjava can compile them both, and when you select ColorStrips.java source file, and click on 'Run', drjava will display the graphical result. FYI - the file you select (ColorStrips.java) is the one that is expected to contain the 'psvm' entry point [main() function]!

In ColorList.java, you'd code would create an ArrayList of java.awt.Color values, and, using a loop, fill the color list with colors created using random r,g,b values. The ColorList class would not return anything, and the constructor would do the color list creation. Use this as a starting point:

```
import java.awt.Color;
import java.util.ArrayList;
import java.util.Random;

class ColorList {

  // 'colList' is the color list we are going to fill and keep. Because it is
  // public, if someone wants a color list (of 256 random colors), they just do:
  // ColorList colors = new ColorList(); // 'colors' will now contain 256 random colors!
  // To access a specific color (eg. the color at index 12), the user just does:
  // Color aColor = colors.colList.get(12); // .get() is a method of ArrayList
  // Search for 'Java ArrayList class' to get more info on using the templated ArrayList class.
  public ArrayList <java.awt.Color> colList = new ArrayList <java.awt.Color> ();


  ColorList() {
    Random rng = new Random(); // rand num generator

    /*
      loop 256 times:
        create r,g,b as random values between 0..255
        create a color using Color c = new ...
        add the color 'c' to the list
      next
   */
  }// constructor
}// ColorList
```

As you can see, ColorList is very simple (you don't need any more code than what is indicated above)! It's also very useful - in just one line (eg. ColorList prettyColors = new ColorList();), we cause the creation of 256 random colors which we can subsequently access, given a 0..255 index.

Next, you need to write ColorStrips.java, which simply has a 'psvm'. In main(), we are going to create a window (of type JFrame - that is what most Java windows are), and add 256 JPanel elements into it one below another (JPanel is called a Panel component (a 'subwindow' of sorts), and is itself capable of holding sliders, buttons, labels, text fields etc, but we won't create any of those here). Conceptually:

```
JFrame - outer window that we drag around
  JPanel - a thin rectangle, colored
  JPanel - another colored rectangle
  JPanel
  ...
  JPanel - 256th colored JPanel, ie the bottommost color in our strip
```

Start with the following outline for ColorStrips.java. I'm showing how you can add two colored panels "by hand" - you need to modify the code to add the 256 colors that will be in your color strip.

```
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.Dimension;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class ColorStrips {

  // "psvm" - our static "launcher" function (starting point for execution)
  public static void main(String[] args) {

    ColorList cols = new ColorList(); // generate colors, to use for populating strips below

    JFrame frame = new JFrame(); // our outer window
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // make it possible to add TWO panels one below another, in a 2x1 grid (2 rows by 1 column)
    frame.setLayout(new GridLayout(2,1));

    // create TWO colored panels, add them to the frame
    // [you need to modify the code, to create 256 colored panels instead]
    JPanel p;
    Color c;
    // #1
    p = new JPanel();
    // 5th color (no good reason to pick this, just arbitrary!)
    c = cols.colList.get(4);
    p.setBackground(c); // color our panel
    p.setPreferredSize(new Dimension(100,4)); // set the dimensions of our panel
    frame.add(p); // pop the panel into the frame
    // #2
    p = new JPanel(); // we can reuse p!
    // again, just arbitrarily picking color #67
    c = cols.colList.get(66);
    p.setBackground(c);
    p.setPreferredSize(new Dimension(100,4));
    frame.add(p);

    // all done
    frame.pack(); // arrange all the panels
    frame.setVisible(true); // show to the world :)
  }// main()
}// ColorStrips
```

As you can see from the above, ColorStrips is also simple! If you compile and run the above without modifications (after creating and compiling ColorList.java), you should see something like:

We added two panels (after setting their size, and background color from our color strips list), and that is what we see above. After your modification, you'd see a taller version that shows all 256 colors in your list.

That's all (drjava.jpg) there is to it! Here are the two steps, summarized:

* complete ColorList.java first - you might need to look up our class notes examples for how to create java.awt.Color() and how to use ArrayList (or you can just Google/Bing them).

* complete ColorStrips.java, make sure it displays a vertical list of pretty color swatches.

Be sure to submit both source files.

Later on, for extra coding practice, you can decorate each little 100x4 JPanel in your program: you have access to each panel's individual pixels in a double for() loop, and can use a paint() method to color each pixel! This lets you create all sorts of fancy patterns to make pretty art, like so. (https://www.google.com/search?q=color+strips&num=100&source=lnms&tbm=isch&sa=X&ei=g9OFVca0MdLHogSj0KOwCg&ved=0(
Look at this program (SinglePanel.java) to learn how you can color pixels of a JPanel (it involves adding a JLabel child to it, where the JLabel is composed of an ImageIcon, which in turn holds a BufferedImage whose pixels we can write into :) :)). Note that there is a thin border around our JLabel - that is ok for now - later, you can read up on how you'd eliminate it.

Enjoy. This should be quick and easy and fun!