Program #3 - CoCo

Prof Bill - Feb 2017

Program #3 logistics:

- Emphasis: image analysis and manipulation, interface, abstract class, Skip Lists, generics, Comparator, javaFX gui, algorithm analysis: O(1), O(log n), O(n)
- Due: **Wed Feb 22, 2017** at the beginning of class
- Worth: **8 points** (8% of your grade)

1. Description

CoCo = Color Counter

In Program #3, we'll read an image file and count how many times each color appears in the image.

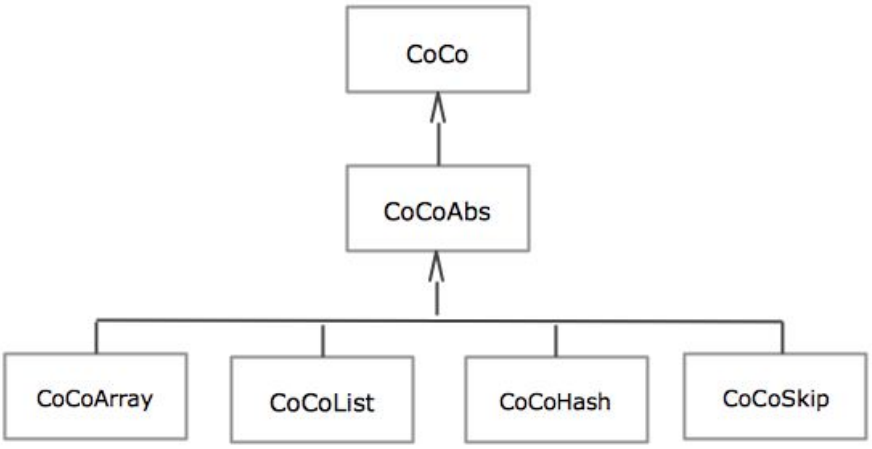
A. **The setup** - Create a class hierarchy for running Coco with an array. Then, add linked list and hash table data structures.
B. **Gui** - Add a simple gui to show an image and its Top 10 colors.
C. **Add O(log N)** - Add a O(log n) structure: array with binary search or Skip List. You code this, not Java Collection Framework.
D. **Results** - Run CoCo, time the results, record and analyze your results.

My goal with P3… 1) Flex your coding muscles (cha!), and 2) Experience the impact of Big-Oh performance of these data structures for real.
CoCo!
thanks… yow, bill

2. Requirements

Here's a more detailed description of the program.

A. The Setup - Here's the class hierarchy for CoCo.



I have code for some of these classes on my k: drive in the **program3 folder**:

- **CoCo** is an interface. It defines the methods to count colors in an image.
- **CoCoAbs** is-a CoCo. It's an abstract class that provides some common code.
- **CoCoArray** is-a CoCoAbs. It's a concrete class that uses ArrayList to count.

More code in program3 folder:

- **ColorEntry** has 2 fields: color and count. It keeps track of how many times a color has been used.
- **CoCoDataStructure** is an enum of the data structures we'll use.

B. Simple gui - Create a simple gui where you show the image, choose a data structure, GO button to run, and then show the results (time, Top 10 colors, etc).
A simple place for your results output is a **javaFX TextArea**.

C. O(log n) structure - code up your own O(log n) data structure: binary search on an array or a skip list. Our text covers both.

Binary search: You can implement binary search on top of ArrayList, which is easiest.
Skip List: I like a Skip List with null pointers, rather than sentinels, like this:
en.wikipedia.org/wiki/Skip_list
For fun, here's the original paper (almost) from ACM Communications, Jun 1990:
homepage.cs.uiowa.edu/~ghosh/skip.pdf

D. Results - run your Program #3 on a test suite of images and record your results.

Question: Are your data structures performing as expected?
Answer: Run Coco!

I'll post a test suite on the k: or somewhere on the web. We'll see. I'll also solicit some images from the class.

One option: store your results in file: coco_results.txt.
Make it a CSV with the following fields, like this:

```
Date, File, DataStructure, NumPixels, NumColors, Time
```

E. Miscellany

Pairs - You can partner with another 210 student for Program #3, if you like. In this case, please do both binary search and skip list data structures. Let me know.

Inspiration - This fun website nudged me in the direction of images and counting colors for P3: mkweb.bcgsc.ca/color-summarizer/

Some coding issues:

- **Piazza** - We'll definitely get a program3 folder going on Piazza. I'll probably do a Helper file too. We'll see.
- **Array** - We'll work on CoCoArray in class.
- **Image gui** - Muganda covers javaFX Image and ImageView classes in section 15.8. Prof Google knows too: "javafx imageview example".
- **More gui** - A simple place for your results output is a javaFX TextArea. Gui is a small component of Program #3.
- **Comparator** - to order your ColorEntry objects (in, for example, a Skip List or array with binary search), you'll want a Comparator that orders these objects based on Color. BTW, I also used a different Comparator to sort my ColorEntry[] array by count.
- **Images** - probably need some chalkboard time on: the x and y axis for images; PixelReader in javaFx; RGB (and RGBA) for javaFX Colors, etc.

Due: You should be done with parts A Setup and B Gui after 1 week, on Wed Feb 15.

Grading

Create a **program3** folder on your k: drive.This folder should contain:

- All your Java source files
- Your program3 executable
- A **README.txt** file where you describe the status of your program and the creative command that you added

All your code must follow our class **Coding Guidelines**. Ugly code will be severely penalized. A program that doesn't even compile is probably worth 0 points.

How to **succeed** in coding: 1. Start early! 2. Don't be shy. Ask a question in class. Email me. Come to office hours. 3. Work in small bites. Divide your program into small, manageable tasks. Knock them down one by one. 4. Always be working. Your program should always compile and run. Never leave your work in disarray.

Remember our **plagiarism** guidelines as well. Getting help from google or stackoverflow or a friend is OK, but:

1. You must acknowledge any help you receive with a comment in your code
2. You must understand any code in your solution
3. Get help on program components, not the primary assignment (the tic tac toe philosophy)
4. If you have any questions in this area, contact me **before** you turn in your work, not after (when it's too late)

thanks… yow, bill