

Project Four A: Hue Finder

Course Logistics

The previously planned three-week-long group/team Project Four is being replaced by two one-week-long individual projects: Project Four A and Project Four B. Each of these individual projects are **OPTIONAL** but recommended as valuable learning experiences. The Programming Project portion of your final grade in this course will only include contributions from Project Four A and/or Project Four B, if those scores improve your overall Programming Project score for the course.

Work Individually

You must develop the code for this assignment entirely on your own. If you encounter difficulties while working, you are encouraged to discuss general algorithms and debugging strategies with anyone you would like. If you feel that getting assistance will require someone else viewing your code, the course staff are the only people that you are allowed to share or show any part of your code with prior to the late deadline for this assignment. This means that you **CANNOT** share code with your team and group mates for this project. You may not store or share your code in any way that other students can access. And you are not allowed to view or make use of any code that is not either written entirely by you, or provided to you by the course staff. We will be running code similarity checking tools on all submissions, and anyone who submits code that is not their own will be severely sanctioned: including (but not limited to) recording a score of zero for this project which is included in your final grade calculation.

Project Introduction

This project allows a user to pick any color (from the over 16 million possible choices), and then search for named colors with a similar hue. The program will either report a list of all named colors that are found with that same hue, or it will report the named colors that are found with a hue that is just smaller (in the red direction = small hue values) and just larger (in the blue direction = larger hue values) than the search color.

For this project, you are provided with code that implements this functionality as a java console application that makes use of a 2-3-4 tree to store and search for named colors. However there are a few bugs within this implementation that you must find and fix (with the help of some provided failing unit tests). After fixing these bugs, you will write an HTML/JS webpage that allows the user to select a color and then pass that color to a CGI program. This CGI program will then perform the same kind of search as the working console application, but will display its results as an output page of html.

Setup

1. Use ssh to log into your GCP VM, or a department CSL machine. **UPDATE (4/16): It may be preferable to complete this activity on your GCP VM so that you can use Java 11. The CSL web server uses Java 7 which does not support streams and lambda expressions. So if you use these machines, you'll need to re-write this portion of the provided code.**
2. Download and extract the contents of the provided starter code, and then change directories to review this code:

```
wget http://pages.cs.wisc.edu/~cs400/ProjectFourA.tar.gz
tar -xzf ProjectFourA.tar.gz
cd ProjectFourA
```

3. Review the contents of the included files:
 - Tree234.java - a mostly working 234 tree implementation
 - TestTree234.java - unit tests for the provided 234 tree
 - Color.java - a class defining a named color
 - HueFinderConsole.java - a driver application for a console based application
 - color.csv - a data file containing several named colors
 - junit5.jar - archive for running junit tests
 - submit - a script to submit your work when it is done

Debugging the 2-3-4 Tree

4. The provided TestTree234 class includes five junit test methods that should all fail when they are run with the provided Tree234.java implementation. If you notice any warnings when compiling the provided code, that those warnings can be ignored for this assignment and do NOT represent the logic errors that you are meant to resolve in this step. Your job is to find and fix the **FOUR BUGS** that were intentionally left within the provided implementation. These bugs were designed to be found and resolved in the same order that their corresponding test methods are listed within TestTree234. After finding and fixing all four bugs, the final test method should also pass. Note that some of the bugs that early tests are meant to point you toward may also effect the correct functionality of later tests. So if you get stuck, feel free to investigate the later bugs to see whether they help lead you to the problems that are also causing earlier tests to fail.
5. After finding and fixing each of the bugs within the provided Tree234 implementation, try compiling the other provided java files, and running the HueFinderConsole application. You will be prompted to enter a color, in terms of it's additive red, green, and blue components (each an integer between 0 and 255 inclusive). This program will then display either 1) a list of named colors that are found with the same hue, or 2) the closest named colors that are found in the red and blue directions from the hue of the color that you entered. This will be the functionality that

you will be porting to web form in the next two sections, so it will be helpful to be familiar with how this program works.

Implementing an HueFinderPrompt.html

6. Instead of running this program from the console, we'd like for our users to be able access this functionality through their browser. Create a new html page named HueFinderPrompt.html with the following required features:
 - The title of this page should be: Hue Finder Prompt.
 - The page should include a meta tag referencing your cs username as the author.
 - The page should include an input element with type="color" that the user can use to select a search color.
 - A button that the user can click after choosing a color, to retrieve the closest hue results.
 - A script section that includes the definition of at least one javascript function. This function should be used to pass the color selected by the user as an argument to the HueFinder.cgi program that you will write in the next step. For reference, you can navigate to a new URL location from your javascript code by using the [window.open\(\)](https://developer.mozilla.org/en-US/docs/Web/API/Window/open) (<https://developer.mozilla.org/en-US/docs/Web/API/Window/open>) method or by setting [window.location.href](https://developer.mozilla.org/en-US/docs/Web/API/Window/location) (<https://developer.mozilla.org/en-US/docs/Web/API/Window/location>).

It is important to understand that the value of the input type="color" element in your page is a color that is formatted as #rrggbb where rr, gg, and bb are the hexadecimal encodings of the color selected. While your page should pass this color in hexadecimal form to the cgi script, you'll need to remove the # character before doing so. This is because the # character has a special meaning to browsers that we are not interested in using. This character can be removed from the color string with the help of the substring function. For example "#13579b".substring(1) returns the string "13579b" without the # character.

Implementing HueFinderCGI.java and HueFinder.cgi

7. Now that you have a webpage directing search requests to HueFinder.cgi, create a HueFinder.cgi bash script that passes its search color on to the Java program that you are about to write. This Java class will be named HueFinderCGI.
8. Create a Java source file named HueFinderCGI.java. This program should contain a main method that takes the search color passed to it from the cgi bash script as a command line argument. It should then load up the named colors from all csv files in the working directory, and then display the named colors with matching or closest hues by printing html out to standard out. When displaying each color, you should print the name of that color as well as its hue in plain text. You should also display what each of these colors look like, by printing out an input element with the

type color, the value set to the hexadecimal color that you are displaying, and the [disabled attribute](#) [_ \(https://www.w3schools.com/tags/att_input_disabled.asp\)](https://www.w3schools.com/tags/att_input_disabled.asp) set: so that the user cannot change these colors. The functionality that you are implementing here is the same as the provided HueFinderConsole.java, so feel free to reuse as much of that code as you are able. The main differences effect the input and output ends of this program.

9. You can copy these files to either your department CSL account, or to your GCP VM to test the complete process from a browser. Just make sure that any changes that you make are copied back into this ProjectFourA folder that you will be submitting below.

Submit

10. [Use the following form](https://docs.google.com/forms/d/e/1FAIpQLSeyTJxZddpS42I-xop9KEUayHkktpyOpORiuEA-t4ttmil8fQ/viewform?usp=sf_link) [_ \(https://docs.google.com/forms/d/e/1FAIpQLSeyTJxZddpS42I-xop9KEUayHkktpyOpORiuEA-t4ttmil8fQ/viewform?usp=sf_link\)](https://docs.google.com/forms/d/e/1FAIpQLSeyTJxZddpS42I-xop9KEUayHkktpyOpORiuEA-t4ttmil8fQ/viewform?usp=sf_link) to name a single color based on it's red, green, and blue color components. A csv file containing these colors will periodically be updated and available for you to [download here](#) [_ \(http://pages.cs.wisc.edu/~cs400/CS400NamedColors.csv\)](http://pages.cs.wisc.edu/~cs400/CS400NamedColors.csv). You are not required to include this file in your final submission, but it may be fun to see your classmates color names appear in your program's search results.
11. Navigate to your HueFinderConsole folder and run the submission script (which will prompt you for your cs login and password) by typing:

```
./submit
```

12. If you are not already in the habit of doing so, please use the -download option with this submission script to retrieve and check the contents of your final submission.
13. After this project has been graded, you'll be able to use the -feedback option with this script to retrieve more feedback about the points that your submission was able to earn.

Grading Rubric

Criteria	Points
Fixing 2-3-4 Tree Bugs	15
Create Working HTML/JS Prompt Page	10
Create Working CGI Backend	15
Name a Color (using google form)	5
Total Points Possible	45