Joanna Klukowska
joannakl@cs.nyu.edu

Project 1: HTML/CSS Colors

Due date: September 18, 11:55PM EST.

You may discuss any of the assignments with your classmates and tutors (or anyone else) but all work for all assignments must be entirely your own . Any sharing or copying of assignments will be considered cheating (this includes posting of partial or complete solutions on Piazza, GitHub or any other public forum). If you get significant help from anyone, you should acknowledge it in your submission (and your grade will be proportional to the part that you completed on your own). You are responsible for every line in your program: you need to know what it does and why. You should not use any data structures and features of Java that have not been covered in class (or the prerequisite class). If you have doubts whether or not you are allowed to use certain structures, just ask your instructor.

In this project you will provide a tool for converting color values from their hexadecimal representation to the RGB and name equivalents. You can think of your tool as a slightly simplified version of the color converted provided by the W3School: https://www.w3schools.com/colors/colors_converter.asp.

There are many ways of representing colors. You will be working with three of them.

- RGB (red, green, blue) values, see https://www.w3schools.com/colors/colors_rgb.asp. Each parameter (red, green, and blue) defines the intensity of the color as an integer between 0 and 255. For example, `rgb(0, 0, 255)` is rendered as blue, because the blue parameter is set to its highest value (255) and the others are set to 0. There are more than 16 million colors ($255^3 = 16,581,375$) that can be represented using the RGB notation.
- Hexadecimal values, see https://www.w3schools.com/colors/colors_hexadecimal.asp. A hexadecimal color is specified with: `#RRGGBB`. `RR` (red), `GG` (green) and `BB` (blue) are hexadecimal integers between `00` and `FF` specifying the intensity of the color. For example, `#0000FF` is displayed as blue, because the blue component is set to its highest value (`FF`) and the others are set to `00`.
- English language names (these names exist only for a subset of the colors represented by RGB and hexadecimal notations). Web developers need to be able to specify the colors when they are designing web pages. CSS (cascading style sheets) is a language that describes the style of an HTML document. Most modern browsers support a large set of color names that can be used in the CSS specification. Here is a list of color names accepted by CSS together with their hexadecimal values: https://www.w3schools.com/cssref/css_colors.asp.

If you need to review the hexadecimal notation, visit the Wikipedia article about it: https://en.wikipedia.org/wiki/Hexadecimal or any other site or resource.

Objectives The goal of this programming project is for you to master (or at least get practice on) the following tasks:

- working with multi-file programs
- reading data from input files
- using and understanding command line arguments
- using the `ArrayList` class
- writing classes
- working with existing code
- extending existing classes (inheritance)
- using and understanding exception handling

All of the skills that you need to complete this project are based on the material covered in cs101. But there might be certain topics for which you did not have to write a program or that you forgot. Make sure to ask questions during recitations, in class and on Piazza way ahead of the due date.

Start early! This project may not seem like much coding, but debugging always takes time.

For the purpose of grading, your project should be in the package called `project1`. This means that each of your submitted source code files should start with a line:

```
package project1;
```

These materials are licensed under CC BY-SA 4.0 license.

Joanna Klukowska
joannakl@cs.nyu.edu

Dataset

In this project you will be working with an input file containing the listing of named CSS colors and their hexadecimal values. Each line in the input file will contain the name of the color, followed by a comma, followed by one or more spaces, followed by a hexadecimal color value. There may be additional white space characters (spaces and tabs) at the beginning and end of each line. Here are a few examples of such lines:

```
AliceBlue, #F0F8FF
AntiqueWhite, #FAEBD7
Aqua, #00FFFF
Aquamarine, #7FFFD4
```

User Interface

Your program has to be a **console based** program (**no graphical interface**, i.e. the program should not be opening any kind of windows to obtain user's input). If you have written only Java FX based programs in cs101, talk to your instructors or the tutors to make sure you know how to develop this program correctly!

Program Usage

This program should use command line arguments. When the user runs the program, he/she will provide the name of the input file containing the list of the named CSS colors as a command line argument. (This way the user can specify an arbitrary list of named colors, not only the ones supported by CSS.)

The user may start the program from the command line or run it within an IDE like Eclipse - **from the point of view of your program this does not matter**.

If the name of the input file provided as a command line argument is incorrect or the file cannot be opened for any reason, the program should display an error message and terminate. It should not prompt the user for an alternative name of the file. If the program is run without any arguments, the program should display an error message and terminate. It should not prompt the user for the name of the file. The error messages should be specific and should describe exactly what happened, for example:

```
Error: the file CSS_colors.txt cannot be opened.
```

or

```
Usage Error: the program expects file name as an argument.
```

Any error messages generated by your code should be written to the `System.err` stream (not the `System.out` stream).[1]

Input and Output

The program should allow the user to enter a hexadecimal color value. It should produce the corresponding RGB value and the color name (if such exists).

The program should run in a loop allowing the user to check as many hexadecimal values as they wish. On each iteration, the user should be prompted to enter either a name (for which the program computes the results) or the word "quit" (any case of letters should work) to indicate the termination of the program.

If the user enters a value that is incorrect (i.e., not a valid hexadecimal value and not the "quit" keyword), an error message should be printed (stating that the entered string is not a valid hexadecimal color value) and the program should allow the user to enter another value.

If the user enters a valid hexadecimal value, that value should be displayed together with the corresponding RGB values and the CSS name (if such exists) - the program should match the formatting of the examples below.

If the user enters a single word "quit" instead of the color value, the program should terminate.

[1] If you are not sure what the difference is, research it or ask questions.

These materials are licensed under CC BY-SA 4.0 license.

Joanna Klukowska
joannakl@cs.nyu.edu

Here is a sample run of the program:

```
Enter the color in HEX format (#RRGGBB) or "quit" to stop:
frog

Error: This is not a valid color specification.


Enter the color in HEX format (#RRGGBB) or "quit" to stop:
#FF0000

Color information:
#FF0000, (255,  0,  0), Red


Enter the color in HEX format (#RRGGBB) or "quit" to stop:
#71052d

Color information:
#71052D, (113,  5, 45),


Enter the color in HEX format (#RRGGBB) or "quit" to stop:
quit
```

Data Storage and Organization

Your need to provide an implementation of several classes that store the data and compute the results when the program is executed.

In particular, your program must implement and use the following classes. You may implement additional classes as well, if you wish.

As you are working on your classes, keep in mind that they should be (and will be) tested separately from the rest of your program.

`Color` Class

The `Color` class stores information about a particular color.

Constructors

This class should provide three different constructors:

```
public Color ( String colorHexValue )
public Color ( String colorHexValue, String colorName )
public Color ( int red, int green, int blue )
```

Valid values for constructor parameters:

- The string representing `colorHexValue` has to be in the format `#XXXXXX` in which the `X` is replaced by a hexadecimal symbol (digits 0-9 and characters A-F in either upper or lower case). Strings formatted differently or containing different characters should be considered invalid.
- The string representing `colorName` can be anything (including `null` or an empty string ””). This means that the following are all valid values for `colorName`: ”baby blue”, ”banana boat”, ”flying saucer”, ”pink”.
- Integer values for `red`, `green` and `blue` have to be in the range from 0 to 255 (inclusive).

All parameters have to be validated according to the rules specified above. If the constructor is called with invalid arguments, then an instance of `IllegalArgumentException` should be thrown carrying an appropriate error message.

These materials are licensed under CC BY-SA 4.0 license.

Joanna Klukowska
joannakl@cs.nyu.edu

There should be no default constructor.[2]

Accessor methods

The class should provide the following accessor methods:

- `int getRed()` returns the value of the red component,
- `int getGreen()` returns the value of the green component,
- `int getBlue()` returns the value of the blue component,
- `String getName()` returns the name of the color, or `null` if there is no name associated with this `Color` object,
- `String getHexValue()` returns the hexadecimal representation of this `Color` object in the format `#XXXXXX` (see above).

Comparisons

This class should override[3] the `equals` methods (see the documentation for the `Object` class for details[4]). The two `Color` objects are equal if the hexadecimal strings representing their values are the same. The color name should not be considered in the comparison for equality.
WARNING: You need to make sure that this function is implemented ignoring the case of letters used in hexadecimal expressions. This means that two colors represented by `#aaaaaa` and `#AAAAAA` should be considered equal. If you use `String` class `equals` method, the return value will be `false`
HINT: Check out the `equalsIgnoreCase()` method in the `String` class.

This class should implement `Comparable<Color>` interface[5]. The comparison should be done by the alphanumeric comparison of the hexadecimal value associated with the color. For example: `#123456` is smaller than `#923456`, which is smaller than `#92A456` (digits are sorted before letters in alphanumeric ordering).
WARNING: You need to make sure that this function is implemented ignoring the case of letters used in hexadecimal expressions. This means that two colors represented by `#aaaaaa` and `#AAAAAA` should be considered equal. If you use `String` class `compareTo` method, the second one will be declared to be smaller.
HINT: Check out the `compareToIgnoreCase()` method in the `String` class.

toString method

The class should override the `toString` method. This method should return a string according to the specified format:

```
#XXXXXX, (RRR, GGG, BBB), NAME
```

or

```
#XXXXXX, (RRR, GGG, BBB)
```

if the `Color` object does not have a name. The uppercase letters in this format are placeholders for:

- `XXXXXX` six hexadecimal symbols, letters should be printed in uppercase
- `RRR` three digits representing the red component. If the number has fewer than three digits, the missing leading digits should be replaced by spaces
- `GGG` three digits representing the green component. If the number has fewer than three digits, the missing leading digits should be replaced by spaces
- `BBB` three digits representing the blue component. If the number has fewer than three digits, the missing leading digits should be replaced by spaces
- `NAME` is a, possibly empty, string representing the color name

For example, if the `Color` object stores the hexadecimal value `#11ab3c` then the string returned by the `toString` method should be `"#11AB3C, ( 60, 17,171),"`. If the `Color` object stores the hexadecimal value `#9ACD32` then the string returned by the `toString` method should be `"#9ACD32, (154,205, 50), YellowGreen"`.

[2] A default constructor is one that can be used without passing any arguments.
[3] see section 2.2 in the textbook if you need to review what *overriding* is
[4] `Object` class documentation: https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html
[5] `Comparable<E>` interface documentation: https://docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

These materials are licensed under CC BY-SA 4.0 license.

Joanna Klukowska
joannakl@cs.nyu.edu

`ColorList` Class

The `ColorList` class should be used to store all the `Color` objects whose hexadecimal value and name are provided in the input file. You can use it in a different context in your assignment as well, but you are required to use it to store the data from the input file.[6]

The class should inherit from `ArrayList<Color>` class. (An alternative implementation would be for this class to contain an instance of `ArrayList<Color>` as a private data field. For the purpose of this project the `ColorList` class **must** inherit from `ArrayList<Color>`.)
HINT: This class should not contain a data field with the type of `ArrayList<Color>`. If you cannot figure out how to implement this class without that data field, you should be talking to the tutors or the instructors.

The class needs to provide the default constructor that creates an empty list.

```
public ColorList ( )
```

The class should implement

- `public Color getColorByName ( String colorName )`
  method that returns the `Color` object in the list whose color name matches the `colorName` specified by the parameter. This method should be case insensitive. If the method is called with a non-existent color name, the return value should be `null`.
- `public Color getColorByHexValue ( String colorHexValue )`
  method that returns the `Color` object in the list whose hexadecimal value matches the `colorHexValue` specified by the parameter. This method should be case insensitive. If the method is called with a non-existent hexadeciamal value, the return value should be `null`.

You may implement other methods, if you wish.

`ColorConverter` Class

The `ColorConverter` class is the actual program. This is the class that should contain the `main` method. It is responsible for opening and reading the data file, obtaining user input, performing some validation and handling all errors that may occur (in particular, it should handle any exceptions thrown by your other classes and terminate gracefully, if need be, with a friendly error message presented to the user).

You may implement other methods in this class to modularize the design.

Programming Rules

You should follow the rules outlined in the document *Code conventions* posted on the course website at https://joannakl.github.io/cs102_f17/notes/CodeConventions.pdf.

The data file should be read only once! Your program needs to store the data in memory resident data structures.

You may not use any of the collection classes that were not covered in cs101 (for this assignment, do not use `LinkedList`, `Stack`, `Queue`. `ColorSet`, `PriorityQueue`. or any classes implementing `Map` interface).

You may use any exception-related classes.

You may use any classes to handle the file I/O, but probably the simplest ones are `File` and `Scanner` classes. You are responsible for knowing how to use the classes that you select.

Working on This Assignment

You should start right away!

[6] "Why should I store the data from the input file in memory rather than reading it from the input file whenever I need to?", you might ask. Well, reading from files stored on disks is MUCH slower than reading from program's memory. How much slower? Take a look at these numbers *Latency Numbers Every Programmer Should know*, https://gist.github.com/jboner/2841832.

These materials are licensed under CC BY-SA 4.0 license.

Joanna Klukowska
joannakl@cs.nyu.edu

You should modularize your design so that you can test it regularly.

Make sure that at all times you have a working program. You can implement methods that perform one task at a time. This way, if you run out of time, at least parts of your program will be functioning properly.

You should make sure that your program's results are consistent with what is described in this specification by running the program on carefully designed test inputs and examining the outputs produced to make sure they are correct. The goal in doing this is to try to find the mistakes you have most likely made in your code.

You should backup your code after each time you spend some time working on it. Save it to a flash drive, email it to yourself, upload it to your Google drive, do anything that gives you a second (or maybe third copy). Computers tend to break just a few days or even a few hours before the due dates - make sure that you have working code if that happens. (A second copy of the files on the same computer is a good idea to keep multiple versions, but it is NOT a good backup since you do not have access to it if there are problems with your computer.)

Grading

If your program does not compile or if it crashes (almost) every time it is run, you will get a zero on the assignment.

If the program does not adhere to the specification, the grade will be low and will depend on how easy it is to figure out what the program is doing.

50 points program and class correctness: the correct values and format of output, correct behavior of methods, handling of invalid arguments

25 points design and implementation of the three required classes and any additional classes

25 points proper documentation, program style and format of submission

How and What to Submit

For the purpose of grading, your project should be in the package called `project1`. This means that each of your submitted source code files should start with a line:

```
package project1;
```

Your should submit all your source code files (the ones with .java extensions only) in a single **zip** file to Gradescope.

You can produce a zip file directly from Eclipse:

- right click on the name of the package (inside the `src` folder) and select Export...
- under General pick Archive File and click Next
- in the window that opens select appropriate files and settings:
  - in the right pane pick ONLY the files that are actually part of the project, but make sure that you select all files that are needed
  - in the left pane, make sure that no other directories are selected
  - click Browse and navigate to a location that you can easily find on your system (Desktop or folder with the course materials or ...)
  - in Options select "Save in zip format", "Compress the contents of the file" and "Create only selected directories"
- click Finish

These materials are licensed under CC BY-SA 4.0 license.