**Regex**

Regex (short for **reg**ular **ex**pressions) are special text strings for describing a search pattern. Regex is extremely useful for isolating expressions in long strings of text. Think of a regular expression as a pattern that describes a particular series of characters.

1. Work through the fantastic regex tutorial / exercises here: [regexone.com](http://regexone.com/). Make sure to do the "Additional Problems" at the end of the tutorial, where you solve some real-world problems!

Note that *this tutorial uses a language-agnostic regex implementation*, though it's very close to how Java's regex works. To use what you learn with Java, you'll need to review actual Java syntax.

1. After you work through the tutorial, it might be helpful to experiment with regex, to see if the expressions you generate match what you think they should match. The site [regexr.com](http://regexr.com/) is really nice for testing regular expressions (and has a nice regex cheat sheet too).
2. Using your knowledge of regex and the String class' replaceAll() method (which has a regular expression parameter), solve the following [CodingBat](http://www.codingbat.com) problems in one statement (one line ending in a semi-colon):
   1. starOut (String-2) //the backslash character must be escaped in Strings
   2. withoutX (String-1)
   3. xyzThere (String-2)
   4. countCode (String-2)
   5. catDog (String-2)
   6. hasBad (String-1)
   7. bobThere (String-2)
   8. **(Advanced)** withoutX2 (String-1)
   9. **(Advanced)** getSandwich (String-2)\*

\*The replaceAll() and replaceFirst() methods *are* capable of capturing groups (the syntax for this is "$1" where 1 represents the group number). Check the String API for more info.

1. (Riddle) Two girls went to a café and both ordered iced tea. One girl drank five iced teas in the time it took the other girl to drink one. All the drinks were poisoned. The girl that drank five survived, while the girl that drank one died. How is this possible?
2. Write the Java code to remove all the HTML tags from the String below (the replaceAll() method will be useful again):

<html><i>This is italic</i><p><b>This is bold</b></html>

1. Write a regular expression that will capture all the valid hex color codes from a String. A color code is valid if it has six characters, and each character is a valid hex value (digits 0-9 and letters a-f). Note that Java's "capture" construct is not quite like the tutorial's implementation. You'll need a Pattern object and a Matcher object (and their methods) for this problem. The regexone site has some info on Java syntax - check the "References & More" section in the top-right.

#aaaef0 color at line 4, #12e4bc at line 9 >>> aaaef0 12e4bc

1. Did you know that MS Word supports regular expressions? Open the **methods.docx** file and open the Find/Replace dialogue box (Crtl + H on Windows and Mac). Click the **More >>>** button, and check **Use wildcards.**

Write a regular expression that will match all of the method calls in the document (note that Word's regex syntax is very similar to but slightly different than what's in the tutorial, use [this](https://support.office.com/en-us/article/Use-wildcard-characters-to-find-or-replace-text-610e37dc-bb2f-4a8b-8fa5-aa991160eafb?ui=en-US&rs=en-US&ad=US) for the syntax specific to Word).

Using the **Format > Font** option at the bottom, change the font of method calls to a monospace font like Courier New or Consolas (this is something I do frequently when writing a new lab).

1. Do you remember the WordCloud project from AP CS? You needed to remove beginning and trailing punctuation from a word to count its frequency (you didn't remove "interior" punctuation). Complete this operation in two lines with replaceAll. Examples:

"something?!" >>> something

"don't" >>> don't

""some-thing?"" >>> some-thing

1. It is very easy to accidentally double words when typing a sentence. Write the Java code, using a regular expression, that will remove all the doubled words in a String, regardless of case.

This is is a sentence >>> This is a sentence

The \b meta-character will help with this problem (a "word boundary", essentially allowing you to specify whole words only). In addition, you will need to google the syntax for referencing captured groups (e.g. \1 and $1 for group one).

**(Advanced) CodingBat String-3**

*Credit for the idea for the Advanced and Over 9000 challenges goes to Liberty student Ben Wyatt in 2017.*

Similar to how you solved some of the problems in CodingBat's String-2 section, solve the following problems in **String-3** in one statement (one semi-colon inside the method body):

* withoutString
* countXYZ
* gHappy

These may or may not involve regex, recursion, and [ternary operators](http://alvinalexander.com/java/edu/pj/pj010018).

**(Over 9000) CodingBat String-3, hard-mode**

Solve the following problems in String-3 in one statement (one semi-colon inside the method body):

* sumNumbers
* maxBlock

These *definitely* involve regex, significant recursion, and [ternary operators](http://alvinalexander.com/java/edu/pj/pj010018). These problems are *really* hard.