Regular expressions

Martin Pola

1 Checking for valid strings

Come up with regular expressions for the use cases below. Test them against at least the given test strings, but try to come up with some test cases on your own, and see if they also work with your regular expressions.

To experiment with regular expressions, regex101.com is a good playground.

Description	Should match	Should fail
Car registration plates	ABC 123 ABC 73K	ABC123 abc 123 ABCD 1234
CSS color codes	#aaaaaa #C849A8 #847290 red green	482cd3 #813na9 #8472906 #123xyz
E-mail addresses	martin@pola.org martin.pola@academy.se martin@canvas.academy.se	martin..pola@netlight.com .martin@academy.se martin@example

2 From HTML to plain text

Define a String in a Java class with some HTML code, such as in this example:

```
<p>Hello everyone!  This is some text,
and here is an image.</p><img src="cat.jpg" alt="Cat" />
```

Use the `replaceAll` method in the `String` class to remove all HTML tags, using a regular expression, leaving you with only the plain text. This is the expected result:
Hello everyone! This is some text, and here is an image.

Make sure that the regular expression works for more complex HTML structures, such as this one:

```
<p>Hello everyone!  All animals are equal, but
<span style="font-style:  italic;">some animals</span> are
more equal than others.
<a href="https://en.wikipedia.org/wiki/Animal_Farm">Read more
at Wikipedia!</a></p>
```

The plain text version of the HTML snippet above should be:
Hello everyone! All animals are equal, but some animals are more equal than others. Read more at Wikipedia!

3 Extracting data

Download neuromancer.txt and save it on your computer. In a Java program, load the contents of the file to a String, and then use regular expressions to extract data from the text file.

When extracting data, the part that you would like to save must be defined as a *group*. That is accomplished by surrounding it by parenthesis. An expression that previously was used to match words with only capital letters, such as `[A-ZÅÄÖ]+`, can be used as a group by changing it to:
`([A-ZÅÄÖ]+)`

Can you, for instance, find any double names? Here is an outline of what the code could look like – make sure to have parenthesis in your regular expression to capture groups:

```java
Pattern pattern = Pattern.compile("...your regex here...");
Matcher matcher = pattern.matcher(neuromancer);

while (matcher.find()) {
    System.out.println(matcher.group());
}
```

Ideaas of things can you can look for:

- Double names (e.g. *Anna-Klara*)
- Words with an initial capital letter followed by at least ten lowercase letters (e.g. *Intercontinental*)
- Words that contain the letter `s` at least four times (e.g. *consciousness*)

Stretch task Given results from the findings above, can you see which word that is the most common? Can you build a highscore of words, perhaps by using a `HashMap` with the word as the key and the frequency count as value?