ShortHands

The list of shorthands

There are several pre-defined shorthands for the commonly used character sets:

- \d is any digit, short for [0-9];
- \s is a whitespace character (including tab and newline), short for [\t\n\x0B\f\r];
- \w is an alphanumeric character (word), short for [a-zA-Z_0-9];
- \b is a word boundary. it doesn't match any specific character, it rather matches a boundary between an alphanumeric character and a non-alphanumeric character (for example, a whitespace character) or a boundary of the string (the end or the start of it). This way, "\ba" matches any sequence of alphanumeric characters that starts with "a", "a\b" matches any sequence of alphanumeric characters that ends with "a", and "\ba\b" matches a separate "a" preceded and followed by non-alphanumeric characters.
- \D is a non-digit, short for [^0-9];
- \S is a non-whitespace character, short for [^ \t\n\x0B\f\r];
- \W is a non-alphanumeric character, short for [^a-zA-Z_0-9].
- \B is a non-word boundary. It matches the situation opposite to that one of the \b shorthand: it finds its match every time whenever there is no "gap" between alphanumeric characters. For example, "a\B" matches a part of a string that starts with "a" followed by any alphanumeric character which, in its turn, is not followed by a word or a string boundary.

Each shorthand has the same first letter as its representation (**d**igit, **s**pace, **w**ord, **b**oundary). The uppercase characters are used to designate the restrictive shorthands.

Example

Let's consider an example with the listed shorthand. Remember, that in Java we use additional backslash \ character for escaping.

```
String regex = "\\s\\w\\d\\s";

" A5 ".matches(regex); // true
" 33 ".matches(regex); // true
"\tA4\t".matches(regex); // true, because tabs are whitespace as well

"q18q".matches(regex); // false, 'q' is not a space
" AB ".matches(regex); // false, 'B' is not a digit
" -1 ".matches(regex); // false, '-' is not an alphanumeric character, but '1' is
```

OK.

String startRegex = "\\bcat"; // matches the part of the word that starts with "cat"
String endRegex = "cat\\b"; // matches the part of the word that ends with "cat"
String wholeRegex = "\\bcat\\b"; // matches the whole word "cat"

If you do not want to use shorthands, we can write the same regex as below:
String regex = "[\\t\\n\\x0B\\f\\r][a-zA-Z_0-9][0-9][\\t\\n\\x0B\\f\\r]";

Find a word within a sentence with at least 5 characters, but no more than 11
Which regex matches any word within a sentence that has at least 5 characters, but no more than 11?

String regex = "\\b\\S{5,11}\\b";

\\d
\\s
\\w
\\S
matches all digits
matches spaces
matches alpha-numeric characters including '_'
matches non-spaces

To understand more about \b and \B -> https://www.freecodecamp.org/news/what-does-b-in-regex-mean-word-boundary-and-non-word-boundary-metacharacters/