Sets, ranges, alternations

The dot character allows us to write common patterns for matching.
The dot, though, matches almost every possible character, and sometimes we want to be more specific in our regex.
Then, the sets come to our rescue.

The set of characters

Each set corresponds to a single character in the string, but what character exactly it can be is defined by the content of the set. The set is written in square brackets, [].
For example, the set "[abc]" means that a single character "a", "b" or "c" can match it.

String pattern = "[bcr]at"; // it matches strings "bat", "cat", "rat", but not "fat"

"rat".matches(pattern); // true
"fat".matches(pattern); // false

You can use as many sets as you want and combine them with usual characters in a line. There are two sets in the following example:

String pattern = "[ab]x[12]"; // can match a or b followed by x followed by either 1 or 2

his pattern can be successfully matched by the following strings:

"ax1", "ax2", "bx1", "bx2"

But the following strings do not match this pattern:

"xa1", "aax1", "bx"

As you can see, the order of sets in regular expressions is important.

the order you put the characters within the set does not matter.

The range of characters

Sometimes we want to make our character sets quite large. In this case, we don't have to write them all down: we can specify a **range** designated by the dash symbol - instead.

The character that precedes the dash denotes the starting point of the range, the character that follows it is the last character that falls into the range.

If the needed characters immediately follow each other in the ASCII/Unicode encoding table we can put them into a set as a range of characters.

This includes alphabetically ordered letters and numeric values. For example, we can write a set that matches every digit:

String anyDigitPattern = "[0-9]"; // matches any digit from 0 to 9

 String anyDigitPattern = "[9-0]";
 System.out.println("9".matches(anyDigitPattern)); at this point we got error…

Exception in thread "main" java.util.regex.PatternSyntaxException: Illegal character range near index 3
[9-0]

The same works for letter ranges such as "[a-z]" or "[A-Z]".
These ranges match all Latin lowercase and uppercase letters respectively. Since patterns are case sensitive, in case we want to match any letter ignoring the case, we can write the following regex:

String anyLetterPattern = "[a-zA-Z]"; // matches any letter "a", "b", ..., "A", "B", ...

Note, that although the range [A-z] is technically valid, it includes additional symbols that are placed between uppercase and lowercase letters in the ASCII table.

As you can see, you can easily put several ranges in one set and mix them with separate characters in any order:
String anyLetterPattern = "[a-z!?.A-Z]"; // matches any letter and "!", "?", "."

Inside set, ? And . Acts like ordinary characters

Excluding characters

We can write a set that will match everything *except* for the characters mentioned in it. To do that, we write the hat character ^ as the first one in the set.

String regex = "[^abc]"; // matches everything except for "a", "b", "c"

"a".matches(regex); // false
"b".matches(regex); // false
"c".matches(regex); // false
"d".matches(regex); // true

The same works for ranges:
String regex = "[^1-6]";

"1".matches(regex); // false
"2".matches(regex); // false
"0".matches(regex); // true
"9".matches(regex); // true

Escaping characters in sets

The general rule is that you do not need to escape special characters within sets if they are used in their literal meaning.
For example, the set [.?!] will match a single period, a question mark, an exclamation mark, and nothing else.
However, the characters that form a set or a range should be escaped or put in a neutral position in case we look for their literal symbols:

- to match the dash character itself, we should put it in the first or in the last position in the set: "[-a-z]" matches lowercase letters and the dash, and "[A-Z-]" matches uppercase letters and the dash;
- hat ^ does not need to be escaped if placed anywhere but the beginning. This way, the set "[^a-z^]" matches everything except for lowercase letters and the hat character;
- square brackets should always be escaped:

String pattern = "[\\[\\]]"; // matches "[" and "]"

Alternations

To this point, we were talking about single characters. There's another way to match one of the listed items, which includes longer sequences to choose from. The vertical bar | is used to match character sequences either before or after the symbol.

String pattern = "yes|no|maybe"; // matches "yes", "no", or "maybe", but not "y" or "e"

"no".matches(pattern); // true

The vertical bar can be used together with parentheses, which designate the boundaries of alternating substrings: everything within the parentheses is an optional substring that can match the alternation block:

String pattern = "(b|r|go)at"; // matches "bat", "rat" or "goat"
String answer = "The answer is definitely (yes|no|maybe)";