Use of ?

Lookahead and lookbehind assertions are advanced features in regular expressions that allow you to check if a pattern is followed by or preceded by another pattern, respectively, without including that pattern in the match. They are useful for defining complex patterns and performing more sophisticated text processing tasks. Let me break them down for you:

- **Lookahead Assertion**:
  - Syntax: (?=pattern) (positive lookahead), (?!pattern) (negative lookahead)
  - Meaning: Checks if the pattern inside the lookahead assertion matches immediately after the current position in the string (positive lookahead) or doesn't match (negative lookahead), without consuming any characters in the match.
    - Example:
    - foo(?=bar) matches "foo" only if it's followed by "bar", but "bar" is not included in the match.
    - foo(?!bar) matches "foo" only if it's not followed by "bar".
- **Lookbehind Assertion**:
  - Syntax: (?<=pattern) (positive lookbehind), (?<!pattern) (negative lookbehind)
  - Meaning: Checks if the pattern inside the lookbehind assertion matches immediately before the current position in the string (positive lookbehind) or doesn't match (negative lookbehind), without consuming any characters in the match.
    - Example:
    - (?<=foo)bar matches "bar" only if it's preceded by "foo", but "foo" is not included in the match.
    - (?<!foo)bar matches "bar" only if it's not preceded by "foo".

These assertions allow you to define conditions that must be satisfied for a match to occur, without actually including the condition in the matched text. They are powerful tools for writing more precise and efficient regular expressions, especially in cases where you need to match patterns based on their context in the input string.

I think more explanation is needed when it comes to the left part of this statement between brackets @ValueSource(ints = {1, 2, 3, 4, 5}). It seems that it is a shorthand for @ValueSource(type = int.class, values = {1, 2, 3, 4, 5})