**Regular Expressions**

**(?<=\.) {2,}(?=[A-Z]) :**

1. (?<=\.) - Before searched match, we need to have a ‘ . ‘ ;
2. **‘ {2,}’ -**  After the point, we need to have two or more spaces
3. **(?=[A-Z])** – After spaces, we need a uppercase letter

**Pattern Matching :**

* **Search patterns describe what should be matched**

**\+359[0-9]{9} : 1. Starts with ‘+’**

**2. Matched ‘359’ like literal**

**3. [0-9] – class witch searching for numbers between 0 and 9.**

**4. {9} – Number in previous class must be repeated 9 times.**

**Compact dis[ck] : 1. Starts with ‘compact dis;**

**2. [ck] – finishes with ‘C/K’**

**Using Regex In Java :**

* Java library supports regular expressions

**Pattern pattern = Pattern.compile("a");**

**Matcher matcher = pattern.matcher("aaaab");**

**while (matcher.find()) {**

**System.out.println(matcher.group());**

**}**

**[aeiouy] – matches a lowercase vowel**

**[0-9] – matches all numbers**

**[a-z] - matches all letters**

**. – matches all symbols**

**[^aeiouy] – matches anything except a lowercase vowel**

**[^0123456789] - Мatches anyting except a digit frm 0 to 9**

**[^0-9] - Negating a character range**

**\d – Shorthand for [0-9] (all numbers)**

**\w – Shorthand for [a-zA-Z0-9\_] (all words)**

**\s – Matches any white-space character (space, tab, line break)**

**Negated Shorthand Character Classes**

**\D – Shorthand for [^0-9]**

**\W – Shorthand for [^a-zA-Z0-9\_]**

**\S – Matches any non white-space character**

**Quantifiers**

**+ - Matches the previous element one or more times**

**\+[0-9]+ - It matches the whole number ‘ +3591233311 ’ ( first ‘+’ is declared like literal )**

**\* - Matches the previous element zero or more times**

**\+[0-9]\* - It matches ‘+359412441’, but also ‘+’**

**? - Matches the previous element zero or one time**

**\+[0-9]? – It matches ‘+35942144411’ and ‘+’**

**{min length, max length} - Exact quantifiers**

**\+[0-9]{10,12} – it matches ‘ 359885976002 ‘ and ‘ 3598859760‘**

**\+[0-9]{3,} – it matches more than 3 numbers**

**Lazy Quantifiers**

**Quantifiers are greedy by default**

**“\.+” - Text "with" some "quotations". -🡪 It matches "with some quotations". GREEDY**

* **Make a quantifier lazy with ‘ ? ‘**

**“\.+?” - Text "with" some "quotations". -🡪 It matches "with” , “quotations" LAZY**

* **If we want to match ‘ . ’, we must escape it. -> ‘ \. ‘**
* **If we want to match ‘( )’, we must escape it ­­-> ‘\(\)’**

**. - Dot matches any character**

**\+.+ - it matches ‘ +359 885/97-60-02 ‘**

**| - Pipe is a logical OR**

**\+359( |-).+ - it matches +359 885/97-60-02 and +359-885/97-60-02 except +359/885/97-60-02**

**^ - The match must start at the beginning of the string or line**

**$ - The match must occur at the end of the string or before \n**

**^\w{6,12}$**

**short**

**too\_long\_username**

**!lleg@l\_ch@rs**

**jeff\_butt**

**johnny**

**LOOK AHEAD and LOOK BEHIND**

**\w{3,16}(?=@) -> Positive look ahead. Matches only word with 3 to 16 letters ending with @**

**\{2,}(?=A-Z]) -> After our match, we`re searching for big letters.**

**Johnnyy Johny@**

**(?<=\.) -> POSTIVE look behind. Before match, we search for ‘.’ .**

**Grouping Constructs**

* **(\d{2})-(\w{3})-(\d{4})**

**22-Jan-2015**

**Group 0 = 22-Jan-2015 Group 1 = 22 Group 2 = Jan Group 3 = 2015**

**(?<name>subexpression) - Captures a named group**

**( ) ( ) ( )**

**\d{2}-(?<month>\w{3})-\d{4}**

**22-Jan-2015**

**Group 0 = 22-Jan-2015 Group "month" = Jan**

* **Non-capturing group**

**(?:AM|PM)**

**Backreference Constructs**

**\number – matches the value of a numbered group**

**\d{2}(-|\/)\d{2}\1\d{4} - > 22-12-2015**

**05/08/2016**

**\k<name> – matches the value of a named group**

**\d{2}(?<del>-|\/)\d{2}\k<del>\d{4} -> 22-12-2015 05/08/2016**