Wyrażenia regularne

Wyrażenia regularne (regular expression) określają **wzorce** dla ciągów znaków. Stosujemy je wtedy gdy szukamy ciągów znaków pasujących do określonego wzorca.

Stosowanie

Znaki specjalne w regexach to

. * + ? { | ( ) [\ ^ $

.	oznacza **dowolny znak**
*	poprzedzające konstrukcje mogą być powtórzone 0 lub więcej razy
+	poprzedzające konstrukcje mogą być powtórzone 1 lub więcej razy
?	konstrukcja jest opcjonalna (0 lub 1 raz)
{a,b}	co najmniej a i maksymalnie b wystąpień
{a,}	co najmniej a wystąpień
{,b}	maksymalnie b wystąpień
{a}	dokładnie a wystąpień
//	to są kwantyfikatory
[abc]	jedna liter a,b,c
[a-zA-Z]	jedna z liter
[abc\[\]] (patrz wyżej)	litera a,b,c lub jeden z nawiasów kwadratowych. Znak \ jest znakiem specjalnym
[.]	dowolna litera
\d	cyfra 0..9
\D	znak nie będący cyfrą czyli też [^0-9]
\s	biały znak
\S	znak nie będacy białym znakiem czyli też [^\s]
\w	mała lub duża litera lub cyfra czyli też [a-zA-Z0-9]
\W	znak nie będący cyfrą lub literą [^\w]

Z dokumentacji \p{Upper} wielka litera POSIX

Grupy czyli **()** służą do łączenia skomplikowanych struktur np. mamy wyrażenie regularne, które dopasowuje fragment, który powtarza się kilka razy. Grupa traktowana jest jako atom.

a(bcd)*	litera a i ciąg bcd 0 lub więcej razy
a(b(cd)?)+	litera a i 1 raz lub więcej b(cd)? Czyli b oraz cd albo wystąpi albo nie. (opcjonalne)

czyli np. ab, abbbbbbb, abcdbcd czyli a i 1 lub wiecej powtórzeń b lub bcd

{[0-9]{1,3}\.}{3}[0-9]{1,3} – wzorzec dla maila, niekoniecznie tylko ten sposób

Klasa Pattern

Reprezentuje samo wyrażenie regularne bez odwoływania się do tekstu. Jego metoda **matcher(String)** zwraca obiekt klasy **Matcher**, który reprezentuje wynik wyszukiwania na konkretnym tekście.

Użycie

1 sposób

```
Pattern pattern= Pattern.compile("[0-9]");
Matcher matcher=pattern.matcher("121212212");
System.out.println(matcher.matches());
```

2 sposób - najszybsze wywołanie statyczne

```
Pattern.matches("[0-9]", "32323")
```

UWAGA: W klasie String jest **matches()**!

Przykłady

https://github.com/idzikpro/JavaBasics/blob/master/src/main/java/pl/idzikpro/regex/RegexMain.java

```
//Czy pierwsze trzy znaki są dowolne, zaś następne 3 znaki tylko cyframi?
System.out.println(Pattern.matches(".{3}[0-9]{3}","abc345"));

//Czy string zawiera tylko cyfry?
System.out.println(Pattern.matches("[0-9]+","545"));

//Czy string zawiera tylko litery?
System.out.println(Pattern.matches("[a-zA-z]+","a"));

//Czy string zawiera tylko cyfry i litery?
System.out.println(Pattern.matches("\\w+","3443dfdf"));

//Czy string zawiera tylko cyfry i ma dokładnie 5 znaków?
System.out.println(Pattern.matches("[0-9]{5}","12345"));

//Czy string rozpoczyna się od "J"?
System.out.println(Pattern.matches("J.*","J"));

//Czy string rozpoczyna się od "J" i kończy na "U"?
System.out.println(Pattern.matches("J.*U","J5UUU"));

//Czy string rozpoczyna się od wielkiej litery?
System.out.println(Pattern.matches("[A-Z].*","Ala"));

//Czy string zawiera spację? (biały znak)?
System.out.println(Pattern.matches(".*\\p{Space}.*","g  g"));

//Czy string jest numerem telefonu zaczynającym się od 7 lub 8 lub 9?
System.out.println(Pattern.matches("[7-9][0-9]{8}","777777777"));

//Czy string jest numerem bankowym? (Zakładając, że zaczyna się od dwóch liter, następnie 4 grupy po 4 cyfry)
System.out.println(Pattern.matches("[a-zA-Z]{2}([0-9]{4}){4}","ab1234123412341234"));

//Czy string jest kodem pocztowym?
System.out.println(Pattern.matches("[0-9]{2}-[0-9]{3}","20-133"));
```