Műveletek szövegekkel, reguláris kifejezések

Alap reguláris kifejezések

.	Bármely karakter
^kifejezés	A kifejezést a sor/szöveg elején keresi.
kifejezés$	A kifejezést a sor/szöveg végén keresi.
[abc]	Megengedett karakterhalmaz (itt a, b vagy c).
[abc][xy]	Két egymásutáni karakterhalmaz (itt első betű a, b vagy c, a második betű x vagy y).
[^abc]	A ^ jel a halmazdefiníciót negálja, azaz bármilyen karakter jó, ami NINCS a halmazban.
[a-d1-7]	Intervallumokat is lehet használni karakterhalmazban.
kif1|kif2	A két kifejezés közül valamelyik.
kif1kif2	A két egymást követő kifejezés (mindkettőnek teljesülnie kell).

Rövidítések

\d	Bármely számjegy (ugyanaz, mint [0-9])
\D	Bármely nem számjegy (ugyanaz,[^{0} mint [^0-9])
\s	Bármely whitespace karakter (szóköz, tab, sortörés stb.) (ugyanaz, mint a [\t\n\x0b\r\f])
\S	Nem whitespace karakter
\w	Betű, nem whitespace (ugyanaz, mint a [a-zA-Z_0-9])
\W	Nem betű

Mennyiségi módosítók (az előttük levő kifejezés darabszámát módosítják)

*	Nulla, vagy több (ugyanaz, mint a {0,})
+	Egy vagy több (ugyanaz, mint a {1,})
?	Nulla vagy egy (ugyanaz, mint a {0,1})
{x}	Pontosan x darab.
{x,y}	x és y darab közötti (beleértve x-et és y-t).

Escape, vagy \ (backslash)

A \ jellel kikapcsolhatjuk a speciális jeleket. Pl. ha [szögletes zárójelet keresünk a szövegben, ami egyébként a karakterhalmaz jele lenne, \[kifejezést használjunk. Hasonlóan a { és a (zárójelekre. A rövidítések megadása esetén Java környezetben kettős visszaperjelre lesz szükség, pl. ”\\d” jelzi a számot.

Csoportosítás és hivatkozás

A reguláris kifejezésekben használhatunk zárójelet a kifejezések csoportosítására (például ”(Kis|Nagy) (Lajos|Ágnes)” → illik a „Kis Lajos”, „Kis Ágnes”, „Nagy Lajos” és „Nagy Ágnes” szövegekre), viszont a zárójelekkel határolt rész egyben felhasználható is lesz az eredmény feldolgozása során. Ez a cseréknél lesz érdekes, amikor a megtalált részt vissza is akarjuk helyettesíteni. Ilyenkor a $ jelöli a teljes mintát, $1 az első zárójelezett részt, $2 a másodikat és így tovább.

Reguláris kifejezések használata Java-ban

s.matches(”kifejezes”)	Teljes illesztést vizsgál. Eredmény: true/false.
s.split(”kifejezes”)	A bemenetet feldarabolja a kifejezés szerint. Az eredmény egy String tömb lesz a darabokkal. A határoló kifejezés nem szerepel az eredményben.
s.replaceFirst(”kifejezes”, ”mire”)	Első illesztés kicserélése.
s.replaceAll(”kifejezes”, ”mire”)	Összes találat kicserélése.

Példák: .matches()

```
jshell> "abc".matches("abc")
$1 ==> true

jshell> "abc".matches("ab")
$2 ==> false

jshell> "abc".matches(".")
$3 ==> false

jshell> "abc".matches("...")
$4 ==> true

jshell> "abc".matches(".{3}")
$5 ==> true
```

A .matches() mindig teljes egyezést vizsgál, azaz nem elég ha a minta szerepel a szövegben, elejétől a végéig egyeznie kell.

```
jshell> "AD2017".matches("[A-Z]{2}2\\d{3}")
$1 ==> true

jshell> "web@webler.hu".matches("[\\w.]+@[\\w.]+\\.[a-z]{2,3}")
$2 ==> true

jshell> "web_webler.hu".matches("[\\w.]+@[\\w.]+\\.[a-z]{2,3}")
$3 ==> false
```

A teljes mailcím validátor ennél azért összetettebb. Ez csak egy egyszerűsített példa.

Példák: .split()

```
jshell> "Nagy Lajos;Debrecen;1987-02-17".split(";")
$1 ==> String[3] { "Nagy Lajos", "Debrecen", "1987-02-17" }

jshell> "itt, ott, amott, erre,arra".split(", ?")
$2 ==> String[5] { "itt", "ott", "amott", "erre", "arra" }

jshell> "abCdeFghIjk".split("[A-Z]")
$3 ==> String[4] { "ab", "de", "gh", "jk" }
```

Példák: .replaceFirst(), .replaceAll()

```
jshell> "Webler Lajos".replaceFirst("([A-Z])[a-z]+(\\s|$)", "$1.$
2")
$1 ==> "W. Lajos"

jshell> "Webler Lajos".replaceAll("([A-Z])[a-z]+(\\s|$)", "$1.$2"
)
$2 ==> "W. L."

jshell> "x 1  y   2    z     3".replaceFirst("\\s{2,}", " ")
$3 ==> "x 1 y   2    z     3"

jshell> "x 1  y   2    z     3".replaceAll("\\s{2,}", " ")
$4 ==> "x 1 y 2 z 3"
```

Feladatok

1. Döntsd el, hogy egy szöveg tartalmaz-e számjegyet.

2. Döntsd el, hogy egy szöveg tartalmaz-e évszámot (YYYY-MM-DD alakban).

3. Döntsd el, hogy egy szöveg tartalmaz-e legalább két darab évszámot.

4. Döntsd el, hogy egy szöveg pontosan két évszámot tartalmaz-e.

5. Döntsd el, hogy egy szöveg tartalmaz-e olyan zárójelbe tett szövegrészt, amiben van írásjel is (pont, vessző, kérdőjel vagy felkiáltójel).

6. Döntsd el, hogy egy szöveg tartalmaz-e o szögletes zárójelet, amiben egy kerek zárójel van (nem több). A zárójelekben lehet egyéb szöveg is. (Például: jó: ”a + [b * (1 + c)]”, nem jó: ”a + [(-b) + 2 * (c -1)]”)

7. Döntsd el, hogy egy szöveg tartalmaz-e nevet, de nem az elején. A név egy olyan szövegrész, ami nagybetűvel kezdődik és kisbetűvel folytatódik, és szóköz vagy írásjel határolja. (Például: jó: ”Ő Lajos.”, nem jó: ”Ez jQuery.”)

8. Cseréld ki egy szövegben az összes évszámot aposztrófosra (2017 → '17).

9. Cseréld ki egy szövegben az összes whitespace-t sima szóközre.

10. Cseréld ki egy szövegben az összes szóközt és ékezetes betűt alulvonásra. De alulvonásból ne legyen több egymásután.

11. Cseréld fel egy szövegben az összes kéttagú névben a vezeték és keresztnevet. (Például: ”Kovács Lajos, Kis János, Nagy Gizella” → ”Lajos Kovács, János Kis, Gizella Nagy”)

12. Töröld ki egy szövegben az időpontokból a másodpercet (Például: ”2017-11-19 23:43:32” → ”2017-11-19 23:43”).