Регуларни изрази

материјал прилагођен за 2012/13 шк. год.

Објекат класе `Scanner` користи регуларне изразе како би у улазном току препознавао и издвајао целе бројеве, реалне бројеве, вредности типа `boolean`, ..., као и неке нестандардне токене које ми сами описујемо регуларним изразима.

Регуларни израз се дефинише објектом класе `String` или класе `Pattern` (`java.util.regex.Pattern`), који представља компајлирану репрезентацију регуларног израза.

Карактери који у регуларном изразу имају специјално значење називају се метакарактерима.
То су:

```
<([{\^-=$!|]})?*+.>
```

Остали карактери унутар регуларног израза представљају сами себе.
Тако нпр. регуларни израз `"oop"` одговара баш стрингу `"oop"` јер ни `o` ни `p` нису метакарактери.

`\\`	backslash \
`\t`	табулатор
`\n`	line feed
`\r`	carriage-return

Ако је потребно упаривање са неким од метакарактера, испред њега се наводи backslash.
Нпр. унутар String-литерала писаћемо \\. на месту где очекујемо баш тачку, односно \\(тамо где очекујемо (, док уколико се регуларни израз учитава нпр. са стандардног улаза или из фајла, тамо се пише \. односно \(.

Карактерске класе

`[abc]`	одговара било ком од карактера наведених унутар заграда [ ] (овде: a, b или c)
`[^abc]`	било који карактер осим a, b или c (негација)
`[a-zA-Z]`	било шта од a до z или од A до Z, укључујући и a, z, A, Z (опсег)

Унутар карактерске класе важи другачији скуп метакарактера.
Нпр. тачка ту губи своје специјално значење, док га - добија (означава опсег).

`[. ,]`	тачка, бланко или запета

Логички оператори

`^`	негација
`|`	или
`&&`	и

`[a-z&&[def]]` – d, e или f, јер: (од a до z) и (d, e или f)
`[a-z&&[^aeiou]]` – сугласник, јер: (од a до z) и (није ниједан од: a, e, i, o, u).

Предефинисане карактерске класе

.	било који карактер (подразумевано осим line separator-a, а са флегом `DOTALL` укључујући line separator)
`\d`	цифра `[0-9]`
`\D`	било који карактер који није цифра `[^0-9]`
`\s`	белина `[ \t\n\x0B\f\r]`
`\S`	било који карактер који није белина `[^\s]`
`\w`	слово, подвлака или цифра `[a-zA-Z_0-9]`
`\W`	било који карактер који није ни слово ни подвлака ни цифра `[^\w]`

Када се у регуларни израз укључују секвенце које почињу обрнутом косом цртом `\` (backslash), како Јава унутар `String`-литерала третира backslash као почетак escape секвенце, backslash унутар регуларног израза који се задаје `String`-литералом мора се писати као `\\`

Границе

`^`	почетак линије (подразумевано улаза, а са флегом `MULTILINE` сваке линије)
`$`	крај линије (подразумевано улаза, а са флегом `MULTILINE` сваке линије и улаза)
`\b`	граница речи
`\B`	није граница речи
`\A`	почетак улаза
`\G`	крај претходног упаривања
`\z`	крај улаза

Када се у регуларни израз укључују секвенце које почињу обрнутом косом цртом `\` (backslash), како Јава унутар `String`-литерала третира backslash као почетак escape секвенце, backslash унутар регуларног израза који се задаје `String`-литералом мора се писати као `\\`

Квантификатори (похлепни покушавају да упаре што дужи, шкрти што краћи део улаза, а посесивни читав улаз)

похлепни	шкрти	посесивни	
`X?`	`X??`	`X?+`	`X` једном или ниједном (опционо `X`)
`X*`	`X*?`	`X*+`	`X` нула или већи број пута
`X+`	`X+?`	`X++`	`X` једном или више пута
`X{n}`	`X{n}?`	`X{n}+`	`X` тачно `n` пута
`X{n,}`	`X{n,}?`	`X{n,}+`	`X` бар `n` пута
`X{n,m}`	`X{n,m}?`	`X{n,m}+`	`X` бар `n`, али не више од `m` пута

Capturing group и back reference

`(X)` подизраз (capturing group). Касније у изразу се може референсирати на њега коришћењем тзв. back reference: `\n` где је `n` индекс групе и одређује се бројањем заграда `(` у регуларном изразу које претходе `X`-у. Броји се од 1. Читав израз је такође једна група, а њен индекс је `0`.
Нпр. регуларни израз `(\d\d)\1` описује произвољне две цифре за којима следе те две исте цифре

Приликом упаривања улаза са регуларним изразом бивају сачуване секвенце са улаза упарене са сваком од група у изразу. Овим секвенцама могуће је приступити након што се операција упаривања успешно заврши.

Line terminator

Подразумевано, било шта од следећег препознаје се као line terminator:

`'\n'`	newline (line feed)
`"\r\n"`	carriage-return за којим непосредно следи line feed
`'\r'`	carriage-return
`'\u0085'`	next-line
`'\u2028'`	line-separator
`'\u2029'`	paragraph-separator

док, уколико се користи флег `UNIX_LINES`, као line terminator препознаје се само `'\n'`.

Употреба флегова

Осим `DOTALL`, `MULTILINE` и `UNIX_LINES`, постоје и други флегови: `CANON_EQ`, `CASE_INSENSITIVE`, `COMMENTS`, `LITERAL`, `UNICODE_CASE`.
Уместо `String`-а са дефиницијом регуларног израза неопходно је користити објекат класе `Pattern` који се добија компајлирањем регуларног израза уз навођење жељеног флега или жељене комбинације флегова. Флегови се могу комбиновати битским или (|) или оператором +.
Нпр.

```
Pattern izraz = Pattern.compile(".{3}",Pattern.DOTALL);
Pattern izraz1 = Pattern.compile("^C", Pattern.CASE_INSENSITIVE+Pattern.MULTILINE);
Pattern izraz2 = Pattern.compile("^C", Pattern.CASE_INSENSITIVE|Pattern.MULTILINE);
Scanner sc = new Scanner("a\nb\ncd");
System.out.println("*" + sc.findWithinHorizon(izraz, 0) + "*");
System.out.println(sc.findWithinHorizon(izraz1, 0));
       // ⇔ System.out.println(sc.findWithinHorizon(izraz2, 0));
```