sal.util.RE

A More Readable REGEX library

This only includes the features that may be useful for the course.

Constant Expressions (1) Escaped Versions of of Characters			
BS	Match a backslash ‘\’	LPAR	Matches ‘(‘
NL	Match a new line ‘\n’	RPAR	Matches ‘)’
DOT	Match ‘.’	LSET	Matches ‘{‘
PLUS	Match ‘+’	RSET	Matches ‘}’
MINUS	Match ‘-’	LSQ	Matches ‘[‘
STAR	Match ‘*’	RSQ	Matches ‘]’
CARET	Match ‘^’	SQUOTE	Matches ‘\’’
DOLLAR	Match ‘$’	DQUOTE	Matches ‘\”’
BAR	Match ‘|’	LPAR	Matches ‘(‘
Constant Expressions (2) Character Sets			
WILD	Any character	WORD	a ‘word’ character
DIGIT	0-9	BOUNDARY	a word boundary
DEC	0-9 (same as DIGIT)	LOWER	a-z
HEX	0-9a-fA-F	UPPER	A-Z
BIN	0-1	ALPHA	UPPER+LOWER
OCT	0-7	ALNUM	ALPHA+DEC

Methods – All return a REGEX		
comment(String start)	Match a single line comment beginning with `start`	comment("//") will match from // to the end of the line.
esc(String)	Prefix string with a BS	esc("ABC") is "\\\\ABC"
esc(char)	Return an escaped unicode sequence	
in(String...)	Concatenate arguments and enclose in [ ]	in(DEC) – a decimal digit
notIn(String ...)	As in but match values not in set	notIn(DEC) – not a decimal digit
From here trailing string arguments are alternates. e.g any("a", "b") means (a|b).*		
any(String ...)	Any number of occurrences (including 0) of one of the strings	any("a", "b") would match e.g. "aababbab". It would also match "".
some(String ...)	Any non-zero number of occurrences of one of the strings	some("a", "b") would match e.g. "aababbab". It would not match "".
maybe(String…)	0 or 1 occurrence.	maybe("a", "b") would match "a", "b" or "".
exactly(int n, String...)	match n occurrences.	exactly(2,"a", "b") will match "ab".
only(int n, String...)	match n occurrences but only if there is no longer match. it is shorthand for exactly(n, *pattern*)+notBefore(*pattern).*	This is different from exactly. Given "abab" exactly(2,"a,b") would match "ab", only(2, "a", "b") would not because a longer match is possible.
one(String …)	equivalent to only(1, String ...)	
two(String ...)	equivalent to only(2, String ...)	
three(String ...)	equivalent to only(3, String ...)	
oneOf(String...)	match one of the alternatives. The data matched is 'captured' and can be accessed later.	oneOf("cat","dog") will match either "cat" or "dog".
notBefore(String ...)	Negative look ahead. The `before(pattern)` is matched but not 'consumed'. X+notBefore("Y") will not match X if the next pattern matches Y.	"="+notBefore("=") will match an "=" which is not followed by a second one.
before(String ...)	Look ahead. if X is any pattern then X+before(Y) means pattern X will fail unless it is immediately before pattern Y.	"=" + before("!") with match "=" in "=!" but not in "=x".