

Review #12 Strings and Regular Expressions

Special Characters

\\	backslash character
\t	tab character
\n	newLine character
\r	carriage return character
\f	form feed character
\a	alert (bell) character
\cx	control character <i>x</i>

Predefined character classes

.	Any character (may or may not match a line terminator)
\d	A digit: [0-9]
\D	A non-digit: [^0-9]
\s	A whitespace character: [\t\n\x0B\f\r]
\S	A non-whitespace character: [^\s]
\w	A word character: [a-zA-Z_0-9]
\W	A non-word character: [^\w]

```
String s = "ab cd"; // tab between b and c
String[] t = s.split("\t"); // splits on tab
for(int i=0;i<t.length;i++)
    System.out.println(t[i] + " ");
```

Output:

```
ab
cd
```

```
String s = "ab cd"; // spaces or tab between b and c
String[] t = s.split("\\s"); // splits on any whitespace
for(int i=0;i<t.length;i++)
    System.out.println(t[i] + " ");
```

Output:

```
ab
cd
```

```
String s = "Now I have seen it all";
String[] t = s.split("\\s");
for(int i=0; i<t.length; i++)
    System.out.println("x" + t[i] + "x");
```

Output:

```
xNowx
xIx
xx
xhavex
xx
xx
xseenx
xx
xitx
xallx
```

```
String s = "and I believe!";
String[] t = s.split("\\S"); // a, n, d give empty Strings
for(int i=0; i<t.length; i++)
    System.out.println("x" + t[i] + "x");
```

Output:

```
xx
xx
xx
x x
x x
```

Leading matches generate empty strings
Contiguous matches generate empty strings
Trailing matches DO NOT generate empty strings

```
String s = "Now I have seen it all. Really!";
String[] t = s.split("\\s+"); // + sign treats contiguous spaces as one space
for(int i=0; i<t.length; i++)
    System.out.println("x" + t[i] + "x");
```

Output:

```
xNowx
xIx
xhavex
xseenx
xitx
xall.x
xReally!x
```

```
String s = "alb2c3";
String[] t = s.split("\\d"); // splits on digits 1, 2 and 3
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:

```
xax
xbx
xcx
```

```
String s = "alb2c3";
String[] t = s.split("\\D"); // splits on chars a, b and c
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:

```
xx
x1x
x2x
x3x
```

```
String s = "abc123";
String[] t = s.split("\\d"); // digits at end do not generate empty
strings for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xabcx

```
String s = "ab123c";
String[] t = s.split("\\d"); // digits 2 and 3 generate empty strings
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xabx
xx
xx
xcx

```
String s = "ab123c";
String[] t = s.split("\\d+"); // digits 123 are treated as one
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xabx
xcx

```
String s = "abc%123*";
String[] t = s.split("\\w");// splits on a, b, c,1,2,3 - empty strings for a, b, c, 2, and 3
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xx
xx
xx
x%x
xx
xx
x*x

```
String s = "abc%123*";
String[] t = s.split("\\W");// splits on %, and *
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xabcx
x123x

Character Classes

[abc]	a, b, or c (simple class)
[^abc]	Any character except a, b, or c (negation)
[a-zA-Z]	a through z or A through Z, inclusive (range)
+	count as one if contiguous
*	split once as empty string and then count as one if contiguous

```
String s = "abcdef";
String[] t = s.split("[be]"); // splits on b and e
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xax
xcdx
xfx

```
String s = "abc def";
String[] t = s.split("[b e]"); // splits on b, space, and e
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

Output:
xax
xcx
xdx
xfx

```
String s = "abcdef";
String[] t = s.split("[ad]"); // splits on a and d
for(int i=0;i<t.length;i++) // empty string before a
    System.out.println("x" + t[i] + "x");
```

split on a: empty string is before a and bcdef is after the a so **t[0] = ""**
 Then split on d: bc before d and ef is after the d so **t[1] = bc**
 Nothing left to split on so **t[2] = ef**

Output:
xx
xabcx
xefx

```
String s = "abc def";
String[] t = s.split("[a d]"); // splits on a, space, and d
for(int i=0;i<t.length;i++) // empty string before a and d
    System.out.println("x" + t[i] + "x");
```

```
Output:
xx
xbcx
xx
xefx
```

```
String s = "abcdbcdef";
String[] t = s.split("[bd]"); // splits on each b and each d
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xax
xcx
xx
xcx
xefx
```

```
String s = "abcdbcdef";
String[] t = s.split("[bd]+"); // treats db as one character
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xax
xcx
xcx
xefx
```

```
String s = "abaabbadbda";
String[] t = s.split("[bd]+"); // treats contiguous b's and d's as one character
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xax
xaax
xax
xax
```

```
String s = "abcdbcdef";
String[] t = s.split("[^bd]"); // treats bcdcd as one character
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xx
xbcdcdx
```

```
String s = "babcdrcedf";
String[] t = s.split("[b-d]+"); // splits on b, c or d
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xx
xax
xrx
xex
xfx
```

```
String s = "akbkckdkek";
String[] t = s.split("k",3); // splits into an array of length 3 on the first 2 k's
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xax
xbx
xckdkekx
```

```
String s = "abcdbcdef";
String[] t = s.split("[cd]",3); // contiguous cd gives empty string
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xabx
xx
xbcdefx
```

```
String s = "abcdbcdefmmnncnabcr";
String[] t = s.split("[b-d]+",3); // splits on first 3 non-contiguous b, c or d
for(int i=0;i<t.length;i++)
    System.out.println("x" + t[i] + "x");
```

```
Output:
xax
xefmmnx
xnabcrx
```

```
String s, regex;
String[] t = s.split(regex);
for(int i=0; i<t.length; i++)
    System.out.println("x" + t[i] + "x");
```

s = "baa2aacab"	s = "baa2aacab"	s = "baa2aacab"
regex="[a]"	regex="[a]+"	regex="[a]*"
xbx xx x2x xx xcx xbx	xbx x2x xcx xbx	xx xbx xx x2x xx xcx xx xbx

s = "mnmogbnoo2omnnggmomn"	s = "mnmogbnoo2omnnggmomn"	s = "mnmogbnoo2omnnggmomn"
regex = "[m-o]"	regex = "[m-o]+"	regex = "[m-o]*"
xx xx xx xx xgbx xx xx x2x xx xx xx xggx	xx xgbx x2x xggx	xx xx xgx xbx xx x2x xx xgx xgx

s = "Java is fun"	s = "Java is fun"	s = "Java is fun"	s = "Java is fun"
regex = "[aeiou]"	regex = "[aeiou]+"	regex = "[aeiou]*"	regex = "[aeiou]+"
xJx xvx x x xs fx xnx	xJx xvx x x xs fx xnx	xx xJx xx xvx xx x x xx xsx x x xfx xx xnx	xJx xvx xsx xfx xnx

s = "COMPUTER"	s = "COMPUTER"	s = "COMPUTER"	s = "COMPUTER"
regex = "[AEIOU]+"	regex = "[^AEIOU]+"	regex = "[AEIOU]*"	regex = "[AEIOU]+"
xCx xMPx xTx xRx	xx xOx xUx xEx	xx xCx xx xMx xPx xx xTx xx xRx	xCx xMPx xTx xRx

Review #12 Strings and Regular Expressions

Consider the code:

```
String s, regex;
String[] t = s.split(regex);
```

or

```
String s, regex;
String[] t = s.split(regex, num);
```

Place the value of each element of the array `t` in the corresponding blank. Use "" for the empty string, _ for a space, and leave undefined cells blank.

	s	regex or regex,num	elements in t
1.	"1 2 3"	" "	_____
2.	"a b c"	"a"	_____
3.	"a b c"	"\\s"	_____
4.	"a b c"	" b"	_____
5.	"broom"	"o+"	_____
6.	"b12cd"	"\\d"	_____
7.	"ac dc"	"\\S"	_____
8.	"3x^5"	"[x^]"	_____
9.	"diligent"	"[aeiou]"	_____
10.	"av ce de jk"	"\\w"	_____
11.	"at#no@time"	"\\W"	_____
12.	"to be or not"	"o",2	_____
13.	"a 12 abc 1234"	"\\d"	_____
14.	"a 12 abc 1234"	"\\d+"	_____
15.	"a 12 abc 1234"	"\\D+"	_____
16.	"abcbdbeb"	"b",3	_____
17.	"carpet"	"[^p-r]"	_____
18.	"carpet"	"[p-r]"	_____
19.	"carpet"	"[p-r]+"	_____
20.	"java"	"^a",2	_____
21.	"weekday"	"[^ae]"	_____
22.	"1a2b3c"	"\\D"	_____
23.	"review #12"	"[e] \\d+"	_____
24.	"a aa aaa aaaa"	"a+",3	_____
25.	"banana split"	"[an]+"	_____

Consider the code:

```
String s, regex;  
String[] t = s.split(regex);
```

or

```
String s, regex;  
String[] t = s.split(regex, num);
```

Place the value of each element of the array `t` in the corresponding blank. Use `""` for the empty string, `_` for a space, and leave undefined cells blank.

	s	regex or regex,num	elements in t
26.	"graduation"	"[aeiou]+"	_____
27.	"graduation"	"[aeiou]*"	_____
28.	"pp qq rr ss t"	"[pr]+"	_____
29.	"convex"	"/*"	_____
30.	"convex"	"/.*"	_____
31.	"abc@aol.com"	"."	_____
32.	"abc@aol.com"	"[.]"	_____
33.	"abc@aol.com"	"a+"	_____
34.	"aaa@aaa.com"	"a*"	_____
35.	"analyze"	"a-o", 3	_____

Review #12 Strings and Regular Expressions - Answers

"" is an empty string; _ is a space

	t[0]	t[1]	t[2]	t[3]	t[4]	t[5]	t[6]	t[7]	t[8]
1.	1	2	3						
2.	""	_b_c							
3.	a	b	c						
4.	a	_c							
5.	br	m							
6.	b	""	cd						
7.	""	""	_						
8.	3	""	5						
9.	d	l	g	nt					
10.	""	""	_	""	_	""	_		
11.	at	no	time						
12.	t	_be_or_not							
13.	a_	""	_abc_						
14.	a_	_abc_							
15.	""	12	1234						
16.	a	c	dbeb						
17.	""	""	rp						
18.	ca	""	et						
19.	ca	et							
20.	""	ava							
21.	""	ee	""	a					
22.	1	2	3						
23.	r	vi	w	#					
24.	""	_	_aaa_aaaa						
25.	b	split							
26.	gr	d	t	n					
27.	""	g	r	""	d	""	t	""	n
28.	""	""	_qq_	""	_ss_t				
29.	""	c	o	n	v	e	x		
30.	convex								
31.	no output								
32.	abc@aol	com							
33.	""	bc@	ol.com						
34.	""	""	@	""	.	c	o	m	
35.	""	""	ythe						