Regular Expressions: Concept Challenge

Concept Challenge: Procedure

- **Pause** Try to solve the problem yourself
- **Discuss** with other learners (if you can)
- **Watch** the UC San Diego learners video
- **Answer** the question again
- **Confirm** your understanding with our explanation

```
public abstract class Document
{    …
     protected List<String> getTokens(String pattern)
     {
         …
```

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

`d.getTokens("[1-3]");` → **`["1", "2", "3", "3"]`**

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

```
d.getTokens(_________); → ["1", "2", "33"]
```

Which of the following regular expressions can you insert in the blank so that it will give the output shown? Select all that apply.

A. "[1233]"
B. "[1, 2, 33]"
C. "[0-9]+"
D. "[1-3]*"
E. "1|2|33"

Expression	Matches
`"a*"`	Zero or more a's
`"a+"`	1 or more a's
`"[a-f]"`	Any character between a and f
`"[^a-cz]"`	Any character which is not between a-c and not z
`"[abc]+"`	1 or more of the character a, b, or c in a row
`"abc"`	The characters abc in a row
`"a|b"`	The character a or the character b

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

`d.getTokens("[1233]");` →

A. "[1233]"

Assume you have a Document object, d, whose text is

```
"Splitting a string, it's as easy as 1 2 33!  Right?"

d.getTokens("[1233]");→   ["1", "2", "3", "3"]
```

A. "[1233]" ✗

Assume you have a Document object, d, whose text is
"Splitting a string, it's as easy as 1 2 33! Right?"

```
d.getTokens("[1,2,33]");
```

B. "[1,2,33]"

Assume you have a Document object, d, whose text is

```
"Splitting a string, it's as easy as 1 2 33!  Right?"
```

```
d.getTokens("[1,2,33]"); → [",", "1", "2", "3", "3"]
```

B. "[1,2,33]" ❌

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

```
d.getTokens("[0-9]+");    →
```

C. "[0-9]+"

Assume you have a Document object, d, whose text is

```
"Splitting a string, it's as easy as 1 2 33!  Right?"
```

```
d.getTokens("[0-9]+");    →  ["1", "2", "33"]
```

C. "[0-9]+" ✓

Assume you have a Document object, d, whose text is
"Splitting a string, it's as easy as 1 2 33! Right?"

`d.getTokens("[1-3]*");`   →

D. "[1-3]*"

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

`d.getTokens("[1-3]*");` →

D. "[1-3]*"

```
["","","","","","","","","","","","","","","",""
,"","","","","","","","","","","","","","","",""
,"","","","","1","","2","","33","","","","","","
","","","",""]
```

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

`d.getTokens("1|2|33");`   →

E. “1|2|33"

Assume you have a Document object, d, whose text is
`"Splitting a string, it's as easy as 1 2 33!  Right?"`

```
d.getTokens("1|2|33");    →   ["1", "2", "33"]
```

E. “1|2|33" ✓

C. "[0-9]+" ✓

E. "1|2|33" ✓

So… Which is better?

C. "[0-9]+" ✓

E. "1|2|33" ✓

So… Which is better?

C. "[0-9]+" ✓

E. “1|2|33" ✓

So… Which is better?

C. "[0-9]+" ✓

E. “1|2|33" ✓

So… Which is better?

Option C is FAR more versatile. It captures ANY non-negative integer (not just 1, 2, and 33).