

String Puzzles

String manipulation



Take a file name apart

PAUSE AND TRY

In plain words, how would you get each piece from the file name?

FILE NAME

"frog.png"

Name: frog

Extension: png

Same name as a .jpg: frog.jpg

By the end of class you can...

1 Create Strings and explain why no String method can change one.

2 Join Strings and numbers with + and +=, and predict the result.

3 Use indexes, length, substring and indexOf to take a String apart.

4 Compare Strings with equals and compareTo.

A large, bold, black number '1' is centered within a white rounded square with a slight drop shadow.

1

PART 1

A String is a row of characters

Two ways to make a String



A **String** object is a sequence of characters. Make one with a **literal** in quotes or with the **String constructor**. No import: String is in java.lang.

LITERAL AND CONSTRUCTOR

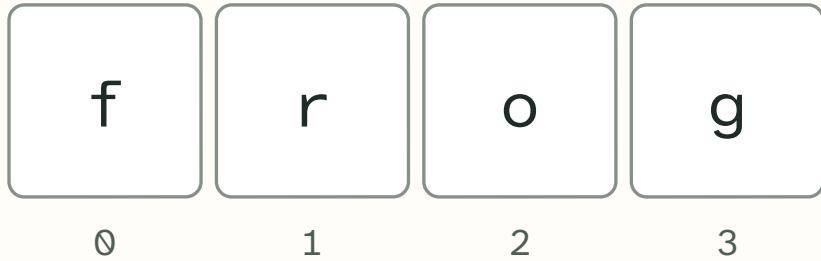
```
String a = "lily";  
String b = new String("pad");  
System.out.println(a.length());  
System.out.println(b.length());
```

OUTPUT

```
4  
3
```

`length()` returns how many characters the String has.

Indexes start at 0



"frog".length() is 4

Indexes: 0 through **length - 1**, so 0 to 3



Using an index outside 0 through length - 1 throws a **StringIndexOutOfBoundsException** while the program runs.

No String method changes the String



Strings are **immutable**. Methods like `substring` return a **new** String and leave the original alone.

WHICH LINE CHANGES POND?

```
String pond = "frog";  
pond.substring(1);  
System.out.println(pond);  
pond = pond.substring(1);  
System.out.println(pond);
```

OUTPUT

```
frog  
rog
```

`pond.substring(1)` returns a new String: the characters from index 1 to the end. To keep it, store it.

A large, white, rounded square containing a bold teal number '2'.

2

PART 2

Joining Strings

+ and += join Strings

JOIN, THEN JOIN AGAIN

```
String greeting = "Hi, " + "Ada";  
greeting += "!";  
System.out.println(greeting);  
String s = new String("pad");  
System.out.println("lily" + s);
```

OUTPUT

```
Hi, Ada!  
lilypad
```



Concatenation: + or += with a String makes a **new** String. Join a number and it becomes text. Join any object and Java calls its `toString` method for you.

Left to right, one + at a time

PREDICT, THEN CHECK

```
System.out.println("Lap " + 2 + 1);
System.out.println(2 + 1 + " laps");
System.out.println("Lap " + (2 + 1));
```

OUTPUT

```
Lap 21
3 laps
Lap 3
```

Expression	First +	Second +
"Lap " + 2 + 1	"Lap 2" (join)	"Lap 21" (join)
2 + 1 + " laps"	3 (add)	"3 laps" (join)
"Lap " + (2 + 1)	3 (in parentheses)	"Lap 3" (join)

Join or add?

PAUSE AND TRY

Write exactly what prints. Say "add" or "join" for every +.

```
int x = 5;  
String s = "x is " + x * 2;  
System.out.println(s);  
System.out.println(1 + 2 + "3" + 4 + 5);
```

Line 3: _____

Line 4: _____

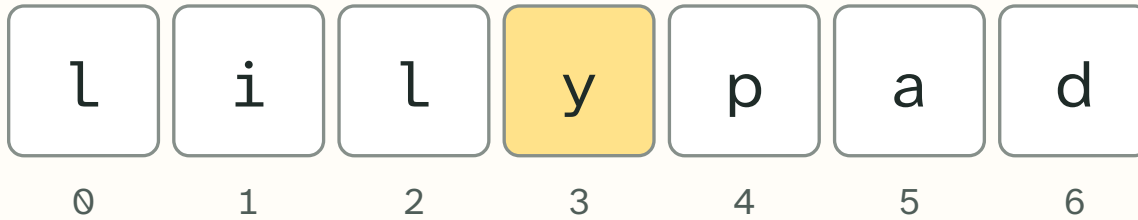


3

PART 3

Taking Strings apart

substring(from, to) stops before to



Call on w = "lilypad"	Result	Rule
w.substring(0, 4)	"lily"	from index 0 up to, not including, 4
w.substring(4)	"pad"	from index 4 to the end
w.substring(3, 4)	"y"	substring(i, i + 1): the one character at i

substring(from, to) always has **to** - **from** characters.

indexOf finds where; substring cuts

WHERE IS IT?

```
String w = "lilypad";  
System.out.println(w.indexOf("pad"));  
System.out.println(w.indexOf("l"));  
System.out.println(w.indexOf("z"));
```

OUTPUT

```
4  
0  
-1
```

FIND, THEN CUT

```
String t = "tadpole";  
int p = t.indexOf("pole");  
System.out.println(t.substring(0, p));
```

OUTPUT

```
tad
```

Cutting past the end



In `substring(from, to)`, **to** can be at most **length()**. Anything past it throws an exception at run time.

COMPILES, THEN CRASHES

```
String w = "frog";  
System.out.println(w.substring(2, 5));
```

WHEN IT RUNS

```
Exception in thread "main"  
java.lang.StringIndexOutOfBoundsException:  
Range [2, 5) out of bounds for length 4
```

Find, cut, and join

PAUSE AND TRY

Write exactly what each println prints.

```
String w = "pollywog";  
int i = w.indexOf("wog");  
String front = w.substring(0, i);  
System.out.println(i);  
System.out.println(front.length());  
System.out.println(w.substring(i, i + 1) + front);
```

1 _____

2 _____

3 _____

A large, white, rounded square containing the number '4' in a bold, teal, sans-serif font.

PART 4

Comparing Strings

equals and compareTo

Call	Result	Why
<code>"frog".equals("frog")</code>	<code>true</code>	every character matches
<code>"frog".equals("Frog")</code>	<code>false</code>	f and F are different characters
<code>"apple".compareTo("banana")</code>	negative	apple comes first in alphabetical order
<code>"pear".compareTo("peach")</code>	positive	r comes after c, so pear comes after peach
<code>"frog".compareTo("frog")</code>	zero	the two Strings are equal

`equals` answers "same?" with true or false. `compareTo` answers "which comes first?" with the sign of an int.

Negative, zero, or positive?

PAUSE AND TRY

Write negative, zero, or positive for A to C, and true or false for D.

A `"cat".compareTo("dog")`

B `"dog".compareTo("do")`

C `"ant".compareTo("ant")`

D `"Ant".equals("ant")`

Words to know

Immutable

Can't be changed once made. String methods return new Strings instead.

Concatenation

Joining Strings with + or +=. Numbers and objects joined to a String become text.

Index

A character's position, from 0 to length - 1.

substring

Returns the characters from one index up to, but not including, another.

indexOf

Returns where the first match starts, or -1 if there is none.

compareTo

Returns a negative, zero, or positive int based on alphabetical order.

Back to frog.png



With your partner, use today's tools to get the name, the extension, and the .jpg name from file = "frog.png".

FILL IN THE BLANKS

```
int dot = file._____;  
String name = file._____;  
String ext = file._____;  
String jpg = _____;
```

Next: the String Puzzles worksheet.