

NAME _____

DATE _____

PERIOD _____

UNIT 1 · TOPIC 1.15 · WORKSHEET

String Puzzles

Snap code together, fill holes from the pool, match the joins, and catch the String method that's lying to you.

About 30 minutes

10 questions

Cumulative through 1.15

TOPIC

1.15

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints, one line per `println`. You may use the Java Quick Reference. Some puzzles include pieces you won't need.

**THINK FIRST**

What does `"hopper".substring(1, 4)` give you? Guess in the margin; question 1 settles it.

PART A · 3 QUESTIONS

A

Warm up your brain

1

YOU'RE THE JVM

You are the Java Virtual Machine. Run this code in your head. What prints?

```
String s = "hopper";
String middle = s.substring(1, 4);
System.out.println(middle + s.length());
```

A opp6**B** oppe6**C** hop6**D** opp5

2 YOU'RE THE COMPILER

You are the compiler. For each snippet, decide whether it compiles. If it does, write what it prints. If it doesn't, say why in a few words.

Snippet	Compiles?	Output, or why not
1 String w = "code"; System.out.println(w.length);		
2 String w = "code"; System.out.println(w.indexOf("d") + 1);		
3 String w = "code"; System.out.println(w.substring(1, 3) + w);		
4 String w = "code"; int first = w.substring(0, 1);		
5 String w = new String("code"); System.out.println(w.length());		

3 MATCH-UP

Four print statements look almost the same. Each one changes one thing. Match each statement to what it prints, from the list below. Each output is used once.

Statement	Output
1 System.out.println("Score: " + 4 + 3);	
2 System.out.println(4 + 3 + " points");	
3 System.out.println("Score: " + (4 + 3));	
4 System.out.println(4 + "" + 3);	

- **P** 7 points
- **Q** Score: 43
- **R** 43
- **S** Score: 7



FAIR QUESTION

What if you + an object that isn't a String?

Every object has a `toString` method, which returns a `String` that represents the object.

When you join any object to a `String` with `+`, Java calls that object's `toString` for you and joins the result. A primitive value such as an `int` is turned into a `String` the same way you just saw.

PART B · 3 PUZZLES

B

Puzzles

4

CODE TILES

A word-flipping program fell off the fridge. It moves the second half of "hopper" to the front. The first line is still stuck in place. Put the tiles in order underneath it so the program prints the output below. One tile is a decoy and isn't used.

Flip.java (already in place)

```
String word = "hopper";
// your tiles go here
```

<code>System.out.println(word);</code>	<code>back += word.substring(0, half);</code>
<code>int half = word.length() / 2;</code>	<code>back += word.substring(half);</code>
<code>System.out.println(back);</code>	<code>String back = word.substring(half);</code>

Target output

perhop

hopper

YOUR PROGRAM

5 SNIPPET POOL

Fill each numbered hole with a snippet from the pool so the program prints Hopper , Grace.

LastFirst.java

```
String name = "Grace Hopper";
int space = name.indexOf( (1) );
String first = name.substring( (2) );
String last = name.substring( (3) );
System.out.println(last + ", " + first);
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

" "	"H"	0, space	space
space + 1	1, space	0, space + 1	



FAIR QUESTION

Why does substring stop one short?

So the math stays clean. `substring(from, to)` always has `to - from` characters, and `s.substring(0, s.length())` is the whole string with no index past the end. One rule, no special cases.

6 FACE-OFF

Tonight: `substring` and `indexOf` argue about who's more useful. Fill in the blanks.

SUBSTRING

Give me a start index and I hand you back a brand-new _____. Give me a start and an end, and I stop _____ the end.

INDEXOF

I don't build anything. I tell you *where* something is, as an _____.

SUBSTRING

And when you can't find it?

INDEXOF

I return _____. Hand that to you as a start index and the program throws a _____.

SUBSTRING

At least I never change the original. After I'm done, word is exactly what it was, because Strings are _____.

PART C · 4 QUESTIONS

C Catch the mistake

7 TWO TRUTHS AND A LIE

Two of these claims about the code are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
String s = "banana";
int a = s.indexOf("an");
String t = s.substring(a, a + 3);
```

- 1 a holds 1.
- 2 t holds "ana".
- 3 After the last line, s holds "ana".

THE LIE, REWRITTEN

8 PENCIL CHECK

Fill in the value of each expression. For compareTo, write only *negative*, *zero*, or *positive*.

Expression	Value
"lighthouse".length()	
"lighthouse".indexOf("house")	
"lighthouse".substring(5, 7)	
"lighthouse".substring(4, 5)	
"lighthouse".indexOf("House")	
"light".equals("Light")	
"light".compareTo("lighthouse")	

9 WHO AM I?

A gang of String methods is playing a party game. Each gives a clue about itself. Name the method.

Clue	I am...
I hand back an <code>int</code> , and <code>-1</code> means I came up empty.	
Give me one <code>int</code> or two, and I build you a new <code>String</code> .	
I tell you how many characters you've got.	
I say <code>true</code> only when every character matches.	
I tell you who comes first in the alphabet.	

10 WRAP-UP



REFLECTION

In one sentence: why can't any String method change the String you call it on?

YOUR ANSWER