

NAME _____

DATE _____

PERIOD _____

UNIT 1 · TOPIC 1.12 · WORKSHEET

Blueprints and Instances

Follow the references, climb the class family tree, and catch the claim that forgot about Object.

About 30 minutes

9 questions

Cumulative through 1.12

TOPIC

1.12

**DIRECTIONS**

Work in pencil. For code, write exactly what prints, one line per `println`. A call like `word.length()` asks a `String` object for its number of characters; calling methods on objects is taught fully in Topic 1.14. Some puzzles include pieces you won't need.

**THINK FIRST**

After `String b = a;` runs, the program gives a a different string. Does b change too? Guess in the margin; question 1 settles it.

PART A · 3 QUESTIONS

A Warm up your brain

1 PENCIL CHECK

Trace the code one statement at a time. After each line, write the `String` object each variable refers to.

```
1 String first = "maple";
2 String second = "cedar";
3 first = second;
4 second = "birch";
```

After line	first refers to	second refers to
1		(not declared yet)
2		
3		
4		

2 YOU'RE THE JVM

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

```
String a = "otter";
String b = "sea otter";
String c = a;
a = b;
System.out.println(c.length());
```

A 9

B 8

C 5

D otter



FAIR QUESTION

If a String variable holds only a reference, why does println show the text?

Because println follows the reference to the object and displays the object's characters. The reference itself, which you can picture as a memory address, stays hidden. That's handy, but it can fool you into thinking the variable *contains* the letters. It contains only the way to find them.

3 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 city = "Boston"; System.out.println(city);		
2 String city = "Boston"; int size = city;		
3 String code = "214"; System.out.println(code);		
4 String code = 214; System.out.println(code);		
5 String a = "owl"; String b = a; System.out.println(b);		
6 int n = 5; String s = "n"; System.out.println(s);		

PART B · 4 PUZZLES

B

Puzzles



KEY IDEA

A streaming app's class family tree

Questions 4 and 5 use this hierarchy. Each indented class is a subclass of the class above it. `MediaItem` names no superclass of its own.

```
Object
  MediaItem  attributes: title, length  behavior: play
    Song      attribute: artist
    Podcast   attribute: host           behavior: skip intro
```

4

PENCIL CHECK

For each kind of object, write *Yes* if it has the attribute or behavior, or *No* if it doesn't.

Object	title	artist	host	play	skip intro
A Song object					
A Podcast object					
A <code>MediaItem</code> object					



WATCH OUT

Inheritance runs one way: **up**. In a school app where `Student` and `Teacher` both extend `Person`, a student has everything a person has, but a person doesn't get a student's grade level, and a student never gets a teacher's room number. Always start at the class you're asked about and climb.

5 TWO TRUTHS AND A LIE

These claims are about the streaming app's family tree. Two are true and one is a lie. Circle the lie, then rewrite it so it's true.

- 1 `MediaItem` names no superclass, so Java makes `Object` its superclass.
- 2 A `Song` object can use the behaviors that the `Object` class defines.
- 3 `Song` is not a subclass of `Object`, because its superclass is already `MediaItem`.

THE LIE, REWRITTEN

6 FACE-OFF

Tonight: the `String` class and the object "Ada" argue about who's more important. Fill in the blanks.

STRING CLASS I'm the _____. I describe what every piece of text knows and what it can do.

"ADA" And I'm one specific _____ of you: an _____ with my own characters.

STRING CLASS Every object made from me has the same kinds of _____ and the same _____.

"ADA" Same behaviors, different answers. Ask me for my `Length()` and I say _____. Ask "Grace Hopper" and you get _____.

STRING CLASS Different values, same plan. There's only _____ of me, but there can be any number of you.

"ADA" Just don't let anyone call you an object. Ask "which `String`?" and the answer is always one of us.

7 CODE TILES

Two shakers got their labels mixed up, so the program must make `left` and `right` trade objects. The first two lines and the last two are stuck in place. Choose three tiles and put them in order so the program prints the output below. Two tiles are decoys.

```
String left = "salt";
String right = "pepper";
// your three tiles go here
System.out.println(left);
System.out.println(right);
```

<code>left = right;</code>	<code>right = left;</code>	<code>String temp = left;</code>
<code>right = temp;</code>	<code>left = temp;</code>	

Target output

pepper

salt

YOUR THREE TILES, IN ORDER

PART C · 2 QUESTIONS

C Catch the mistake

8 ONE-TOKEN FIX

A game should save the old level in `before`, move `level` up, and print `Level 1` and then `Level 2`. It prints `Level 2` and then `Level 1` instead. Change exactly one token to fix it.

```
1 String level = "Level 1";
2 String before = level;
3 before = "Level 2";
4 System.out.println(before);
5 System.out.println(level);
```

LINE, OLD TOKEN, NEW TOKEN

9 WHO AM I?

Some ideas about objects are playing a party game. Each gives a clue about itself. Name the idea.

Clue	I am...
I hold the attributes and behaviors that related classes share.	
I extend another class and get its attributes and behaviors without rewriting them.	
I'm the relationship that runs from a subclass up to its superclass.	
I'm what a <code>String</code> variable really holds. Think of me as a memory address.	
My variables keep their value right in their own box, like <code>int</code> and <code>double</code> .	