

NAME _____

DATE _____

PERIOD _____

UNIT 1 · TOPIC 1.14 · WORKSHEET

Ask the Object

Put the dot to work, keep each object's count straight, and catch the call that has no object to ask.

About 30 minutes

10 questions

Cumulative through 1.14

TOPIC

1.14

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints, one line per `println`. Some puzzles include pieces you won't need.

**MEET THE METHODS****Two String methods and a made-up Counter class**

Puzzles use only these, plus the Scanner methods `next()` and `nextInt()` from Topic 1.4. Topic 1.15 covers String methods in full.

```
// String instance methods
int length()           // number of characters
int indexOf(String str) // index where str first appears
                        // (from 0), or -1 if not found

// Counter: a made-up class
Counter()              // a new count starts at 0
void click()           // adds 1 to the count
void add(int amount)   // adds amount to the count
int getCount()        // returns the count
```

PART A · 2 QUESTIONS

A Warm up your brain**1** PENCIL CHECK

After `String word = new String("lilypad");`, the variable `word` refers to a `String` object. Fill in the value of each expression.

Expression	Value
<code>word.length()</code>	
<code>word.indexOf("l")</code>	
<code>word.indexOf("z")</code>	
<code>word.length() - word.indexOf("pad")</code>	

2 YOU'RE THE JVM

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

```
Counter a = new Counter();
Counter b = new Counter();
a.click();
b.add(4);
a.add(b.getCount());
System.out.println(a.getCount());
```

A 5**B** 4**C** 1**D** 9

PART B · 4 PUZZLES

B Puzzles

3 YOU'RE THE COMPILER

You are the compiler. In every snippet, `word` is a `String` holding pond. Decide whether it compiles. If it does, write what it prints, or what goes wrong when it runs. If it doesn't, say why in a few words.

Snippet	Compiles?	Output, or why not
1 <code>System.out.println(word.length());</code>		
2 <code>int n = length(word);</code>		
3 <code>int n = String.length();</code>		
4 <code>int n = word.length;</code>		
5 <code>String w = null;</code> <code>int n = w.length();</code>		
6 <code>int n = word.length(2);</code>		



FAIR QUESTION

length is a String method. So why can't you write `String.length()`?

Because `length` is an *instance* method: it measures one particular `String` object.

`String` is the class, the blueprint, and a blueprint has no characters to count. Java needs to know *which* object you're asking, and the name before the dot tells it. A class name goes before the dot only for a class method, like `Math.abs(-6)`, whose header has `static`.

4 SNIPPET POOL

Fill each hole from the pool so that when the user types banana, the program prints 6 and then 1. Each snippet is used at most once.

Letters.java

```
import java.util.Scanner;
Scanner console = new Scanner(System.in);
String word = console.;
int size = ;
int spot = ;
System.out.println(size);
System.out.println(spot);
```

THE POOL

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

word.indexOf("a")	length(word)	next()
"a".indexOf(word)	word.length()	nextInt()
console.length()		

5 CODE TILES

A tally program fell off the fridge. Put the tiles in order so it prints 5. One tile is a decoy and isn't used. (Hint: `tally.add(tally.getCount())` adds the current count to itself.)

tally.add(3);	System.out.println(tally.getCount());
Counter.click();	Counter tally = new Counter();
tally.add(tally.getCount());	tally.click();

YOUR PROGRAM

6 FACE-OFF

Tonight: the compiler and the JVM argue about `String w = null;` followed by `w.length();`. Fill in the blanks.

COMPILER I check _____. `w` is declared as a `String`, so `w.length()` looks fine to me.

JVM Then I run it, and `w` holds _____. There's no object to ask.

COMPILER Not my problem. I never know what a variable holds until the program _____.

JVM So I stop everything with a _____.

COMPILER At least I catch `w.length` with no parentheses. That's a _____ error, so nothing runs at all.

JVM Fine. Every instance method call needs a real _____ before the dot.

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.

```
Counter x = new Counter();
Counter y = new Counter();
Counter z = y;
x.add(2);
z.add(5);
```

- 1 After the last line, `x.getCount()` returns 2.
- 2 After the last line, `y.getCount()` returns 5.
- 3 This code creates three `Counter` objects, one for each variable.

THE LIE, REWRITTEN

8 ONE-TOKEN FIX

This program crashes, but it should print 3. Change exactly one token on line 3 to fix it.

```
1 String name = null;
2 String nick = new String("Kit");
3 System.out.println(name.length());
```

OLD TOKEN, NEW TOKEN, AND WHY IT CRASHED

9 WHO AM I?

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

Clue	I am...
I sit between an object's name and the method being called.	
I belong to an object, so you call me with that object's name in front.	
I'm what you get when you call a method on null.	
I'm the Scanner method that reads the next whole number.	
Call me on a String and I tell you how many characters it has.	

10 WRAP-UP



REFLECTION

In one or two sentences: in a `getCount()`, what job does a do, and why can the same call give different answers for different objects?

YOUR ANSWER