

NAME

DATE

PERIOD

UNIT 1 · TOPIC 1.13 · WORKSHEET

Under Construction

Build objects with new, pick the right constructor, and catch the variable that's holding nothing at all.

About 30 minutes

10 questions

Cumulative through 1.13

TOPIC

1.13



DIRECTIONS

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

PART A · 2 QUESTIONS

A

Warm up your brain

1

PENCIL CHECK

These five lines run in order. For each one, write how many **new** objects it creates and what the variable holds afterward.

Statement	New objects	The variable holds
<code>String a = new String("pond");</code>		
<code>String b = a;</code>		
<code>String c = null;</code>		
<code>String d = new String();</code>		
<code>Scanner e = new Scanner("4 9");</code>		

2 YOU'RE THE COMPILER

You are the compiler. The `String` constructors this course uses are `String()` and `String(String original)`. Assume `import java.util.Scanner;` is at the top. 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 <code>String s = String("lily");</code>		
2 <code>String s = new String("lily");</code> <code>System.out.println(s);</code>		
3 <code>String s = new</code> <code>Scanner(System.in);</code>		
4 <code>int n = null;</code>		
5 <code>String s = null;</code> <code>System.out.println(s);</code>		
6 <code>String s = new String(5);</code>		



FAIR QUESTION

Then how does `String name = "Ada";` work with no `new`?

A string literal is a shortcut that only `String` gets: Java builds the `String` object for you. Every other class, like `Scanner`, needs `new` and a constructor. You'll see more about literals in Topic 1.15.

PART B · 4 PUZZLES

B

Puzzles



MEET THE CLASS

Parcel, a made-up class

Questions 3 and 5 use this class. Here are its four constructors, read from its documentation. Each one prints its values, one per line, exactly the way `println` would show them.

```
Parcel()                // prints: no label
Parcel(String city)     // prints: city
Parcel(String city, int boxes) // prints: city, then boxes
Parcel(int boxes, double pounds) // prints: boxes, then pounds
```

3

SNIPPET POOL

Fill each numbered hole with a snippet from the pool so the program prints these five lines, in order: Reno, Ely, 4, 5, 2.0.

Shipping.java

```
Parcel a = new Parcel();
Parcel b = new Parcel();
Parcel c = new Parcel();
```

THE POOL

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

Reno	5, 2	"Ely", 4	4, "Ely"
"Reno"	5.0, 2	2, 5	



THINK FIRST

When `main` reaches `new Sprout(water)`, does the constructor's `println` run right then, or after the rest of `main`? Guess in the margin; question 4 settles it.

4 YOU'RE THE JVM

You are the Java Virtual Machine. Run this program in your head and write the exact output. Follow the calls, not the order of the lines on the page. (The word `public` in the main header is explained in a later topic; `static` marks `main` as a class method, as in Topic 1.10.)

Garden.java

```
public class Garden {  
    public static void main(String[] args) {  
        int water = 3;  
        System.out.println("plant");  
        Sprout s = new Sprout(water);  
        System.out.println(water);  
        Sprout t = new Sprout(water + 1);  
        System.out.println("done");  
    }  
}  
  
class Sprout {  
    Sprout(int cups) {  
        cups = cups * 2;  
        System.out.println(cups);  
    }  
}
```

OUTPUT

5 CODE TILES

A shipping program fell off the fridge. It reads a city and a box count, builds a `Parcel` (from the box above question 3), and says it's shipped. Put the tiles in order below the starter code so that when the user types `Ely 4`, it prints the output below. Two tiles are decoys. (Reading with `next` and `nextInt` works as in Topic 1.4; the full rules for calling methods on objects come in Topic 1.14.)

Ship.java (starter code, already in place)

```
import java.util.Scanner;
Scanner console = new Scanner(System.in);
```

<code>System.out.println("shipped");</code>	<code>Parcel p = new Parcel(boxes, city);</code>
<code>String city = console.next();</code>	<code>int boxes = console.nextDouble();</code>
<code>Parcel p = new Parcel(city, boxes);</code>	<code>int boxes = console.nextInt();</code>

Target output

Ely

4

shipped

YOUR PROGRAM



WATCH OUT

Java matches arguments to parameters by **position**, not by meaning. If a class had `Seat(int row, int number)`, then `new Seat(12, 3)` compiles even if you meant row 3, seat 12. Both values are `int`, so the compiler can't catch the swap. Read the header, then line up your arguments left to right.

6 FACE-OFF

Tonight: an empty `String` and `null` argue about which of them is really "nothing." Fill in the blanks.

EMPTY STRING	I'm a real _____. Someone wrote <code>new String()</code> , and the constructor built me with zero characters.
NULL	I'm not an object at all. I'm what a reference variable holds when it refers to _____.
EMPTY STRING	Print me and you get an empty line. Print a variable holding you and you get the word _____.
NULL	At least I'm cheap. <code>String s = null;</code> runs _____ constructor and creates _____ objects.
EMPTY STRING	Cheap, but picky. Try <code>int n = null;</code> and see what the compiler says.
NULL	Fine. I only fit in variables of a _____ type, like <code>String</code> or <code>Scanner</code> .

PART C · 4 QUESTIONS

C Catch the mistake

7 TWO TRUTHS AND A LIE

A made-up `Kite` class has these three constructor headers. Two of the claims are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
Kite(String color)
Kite(String color, int tail)
Kite(int tail, String color)
```

- 1 The signature of the second constructor is `Kite(String, int)`.
- 2 `new Kite(4, "red")` runs the third constructor.
- 3 The class could also have `Kite(String shade, int length)`, because its parameter names are different.

THE LIE, REWRITTEN

8 ONE-NAME FIX

This code prints `null`, but it should print `plum`. Fix it by changing exactly one variable name into the other variable's name. There are two different fixes. Find both.

```
1 String fruit = null;
2 String snack = fruit;
3 fruit = new String("plum");
4 System.out.println(snack);
```

YOUR TWO FIXES: LINE, OLD NAME, NEW NAME

9 WHO AM I?

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

Clue	I am...
I have exactly the same name as my class, and I run every time an object of my class is built.	
Put me in front of a constructor call and Java builds a brand-new object.	
I'm a name plus the ordered list of parameter types. Parameter names don't make the cut.	
We share one name but have different parameter lists, and Java picks one of us by looking at the arguments.	
I hand each parameter a copy, so the caller's variable never changes.	

10 WRAP-UP



REFLECTION

In one or two sentences: when a class has several constructors, how does Java decide which one a new call runs?

YOUR ANSWER