

NAME

DATE

PERIOD

UNIT 1 · TOPIC 1.9 · WORKSHEET

Read the Header

Pull headers apart, follow the jump into a method and back, and catch the method that hands back nothing.

About 30 minutes

9 questions

Cumulative through 1.9

TOPIC

1.9



DIRECTIONS

Work each puzzle in pencil. For code, write exactly what prints, one line per `println`. Many headers start with `public static`; those words are explained in a later topic, so read past them for now. Some puzzles include pieces you won't need.

PART A · 2 QUESTIONS

A Warm up your brain

1 PENCIL CHECK

Each header comes from the `Math` or `String` documentation, except the last, which is made up. You only need to read them; calling `String` methods comes in Topic 1.14. Fill in the return type and the signature.

Header	Return type	Signature
<code>public static double sqrt(double a)</code>		
<code>public int length()</code>		
<code>public int indexOf(String str)</code>		
<code>public static double pow(double a, double b)</code>		
<code>public static void cheer(int times)</code>		

**THINK FIRST**

A method is written in the file above `main`, but nothing ever calls it. Does its `println` ever run? Guess in the margin; question 2 settles it.

2 YOU'RE THE JVM

You are the Java Virtual Machine. Java starts every program at `main`. Run this one in your head and write the exact output.

Snack.java

```
public class Snack {  
    public static void sayHi() {  
        System.out.println("hi");  
    }  
  
    public static int doubleIt(int n) {  
        System.out.println("doubling");  
        return n * 2;  
    }  
  
    public static void main(String[] args) {  
        System.out.println("start");  
        int x = doubleIt(4) + 1;  
        System.out.println(x);  
        System.out.println("end");  
    }  
}
```

OUTPUT

PART B · 3 PUZZLES

B Puzzles

3 YOU'RE THE COMPILER

You are the compiler. Two made-up methods are in the same class as `main`, so a call is just the name and parentheses. `printBadge` prints the name, then the level. `average` returns the average of `a` and `b`.

```
public static void printBadge(String name, int level)
public static double average(int a, int b)
```

For each snippet, decide whether it compiles. If it does, write what it prints (write *nothing* if it prints nothing). If it doesn't, say why in a few words.

Snippet	Compiles?	Output, or why not
1 <code>double avg = average(80, 91);</code> <code>System.out.println(avg);</code>		
2 <code>int avg = average(80, 91);</code>		
3 <code>printBadge(3, "Ada");</code>		
4 <code>String b = printBadge("Ada", 3);</code>		
5 <code>average(80, 91);</code>		
6 <code>System.out.println(average(7, 8));</code>		



FAIR QUESTION

Why can't Java just figure out which argument is which?

Because it pairs arguments with parameters by position, and nothing else. It can't tell that 3 looks like a level and "Ada" looks like a name; it only checks that the first argument fits the first parameter's type, the second fits the second, and so on. The header's order is the order you must use.

4 SNIPPET POOL

Three made-up methods live in the same class as `main`. Fill each hole with a snippet from the pool so the program prints `ready`, then `150`, then `75.0`. Each snippet is used at most once.

Documentation

```
public static int minutesIn(int hours) // returns hours * 60
public static double half(double x) // returns x / 2
public static void printLine(String s) // prints s
```

Planner.java (inside main)

```
int m = minutesIn( (1) ) + 30;
double h = half( (2) );
printLine( (3) );
System.out.println(m);
System.out.println(h);
```

THE POOL

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

<code>m / 2</code>	<code>"ready"</code>	<code>2.0</code>	<code>m</code>
<code>ready</code>	<code>2</code>	<code>"m"</code>	

5 FACE-OFF

Tonight: a void method and a non-void method argue about who's more helpful. Fill in the blanks.

VOID My header says _____ before my name. I do my job and hand back _____.

NON-VOID My header names a type. My _____ sends back a value of that type.

VOID People write `double x = average(4, 5);` with you. Try that with me: error.

NON-VOID Your call must stand alone as a _____. Mine can go in a _____.

VOID And when someone calls you and ignores what you return?

NON-VOID Then my value is _____, and it still compiles. Sneaky.

PART C · 4 QUESTIONS

C

Catch the mistake

6

TWO TRUTHS AND A LIE

Math has two `abs` methods; each returns its argument's distance from zero. (Calls like `Math.abs(-7.0)` are taught fully in Topic 1.10.) Two claims are true and one is a lie. Circle the lie and rewrite it.

```
public static int abs(int a)
public static double abs(double a)
```

- 1 These two headers have different signatures, so `abs` is overloaded.
- 2 `Math.abs(-7.0)` runs `abs(double)` and returns `7.0`.
- 3 `public static double abs(int a)` could also be added, since its return type differs.

THE LIE, REWRITTEN



FAIR QUESTION

If the return type isn't in the signature, why is it in the header?

Because the caller needs it: it tells you what kind of value comes back and where you can store it. Java picks *which* method runs from the call alone, and a call like `abs(-7)` shows only the name and the argument types.

7 MATCH-UP

Two more made-up methods are in the same class as `main`. `addBonus` adds 10 to its parameter and prints nothing. `withBonus` returns its parameter plus 10.

```
public static void addBonus(int score)
public static int withBonus(int score)
```

Match each snippet to what it prints. Each output is used once.

Snippet	Code	Output
1	<pre>int s = 80; addBonus(s); System.out.println(s);</pre>	
2	<pre>int s = 80; s = withBonus(s); System.out.println(s);</pre>	
3	<pre>int s = 80; System.out.println(withBonus(s) + 5);</pre>	
4	<pre>int s = 80; int t = withBonus(withBonus(s)); System.out.println(t);</pre>	

- **P** 95
- **Q** 80
- **R** 100
- **S** 90



WATCH OUT

A method can't send an answer back by changing its parameter; the parameter is only a copy. If the caller needs a new value, the method has to `return` it, and the caller has to store it or use it.

8 WHO AM I?

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

Clue	I am...
I'm the word in a header that says this method hands nothing back.	
I send a value back to the caller and end the method on the spot.	
I'm a variable declared in a method's header, and I exist only inside that method.	
We share a name but have different parameter lists.	
I let you use a method by knowing what it does, without knowing how it was written.	

9 WRAP-UP



REFLECTION

In one or two sentences: what's the difference between a parameter and an argument?
Use `average(80, 91)` as your example.

YOUR ANSWER