

NAME

DATE

PERIOD

UNIT 3 · TOPIC 3.8 · WORKSHEET

Where Names Live

Map every variable's block, catch the name that wandered out of scope, and unmask the parameter hiding an instance variable.

About 30 minutes

9 questions

Cumulative through 3.8

TOPIC

3.8



DIRECTIONS

Work in pencil. Lines marked *in main*, *in another class* run in a separate class's `main`, in order. Write output exactly as Java prints it. Some puzzles include pieces you won't need.



THINK FIRST

A method starts with `int total = 0;` and then adds 10 to `total`. If you call the method three times, does `total` reach 30? Guess; question 2 settles it.

PART A · 3 QUESTIONS

A

Warm up your brain

1

FACE-OFF

Tonight: a local variable and an instance variable argue. Fill in the blanks.

LOCAL

I live from my declaration to the end of my _____. Then I'm gone.

INSTANCE

You start over on every _____. I last as long as my _____.

LOCAL

Nobody outside my block can touch me. I never need `public` or _____.

INSTANCE

Every method in my class can use me. My `private` keeps other classes out.

LOCAL

Share your name and I _____ you: in my method, the name means me.

2 YOU'RE THE JVM

You are the JVM. What prints?

```
class Card {
    private int stamps = 0;
    public int punch() {
        int bonus = 0;
        bonus += 10;
        stamps++;
        return stamps + bonus;
    }
}
// in main, in another class
Card c = new Card();
System.out.println(c.punch() + " " + c.punch() + " " + c.punch());
```

A 11 12 13

B 11 21 31

C 11 11 11

D 11 22 33

3 ONE-WORD FIX

This should print the warmest reading, 75. It prints 0. Delete exactly one word to fix it.

```
1 class Thermometer {
2     private int warmest = 0;
3     public void reading(int temp) {
4         if (temp > warmest) {
5             int warmest = temp;
6         }
7     }
8     public int getWarmest() { return warmest; }
9 }
10 // in main, in another class
11 Thermometer t = new Thermometer();
12 t.reading(75);
13 t.reading(61);
14 System.out.println(t.getWarmest());
```

LINE, THE WORD YOU DELETE, AND WHY

PART B · 3 PUZZLES

B Puzzles

4 PENCIL CHECK

Map the scope. For each variable, write what kind it is, then whether its name could be used on line 13 and on line 16.

```

1  class Password {
2      private String secret;
3
4      public Password(String s) {
5          secret = s;
6      }
7
8      public int digits() {
9          int found = 0;
10         for (int i = 0; i < secret.length(); i++) {
11             String ch = secret.substring(i, i + 1);
12             if ("0123456789".indexOf(ch) >= 0) {
13                 found++;
14             }
15         }
16         return found;
17     }
18 }
```

Variable	Kind	Line 13?	Line 16?
secret			
s			
found			
i			
ch			

5 CODE TILES

Build the body of `grade` so it returns "A" for a score of 90 or more and "B" otherwise. Arrange the tiles so `main` prints the target output. One tile is a decoy.

```
class Report {
    public String grade(int score) {
        // your tiles go here
    }
}
// in main, in another class
Report r = new Report();
System.out.println(r.grade(95));
System.out.println(r.grade(80));
```

<code>letter = "A";</code>	<code>return letter;</code>	<code>String letter = "A";</code>
<code>if (score >= 90) {</code>	<code>String letter = "B";</code>	<code>}</code>

Target output

A

B

YOUR METHOD BODY

6 SNIPPET POOL

Fill each hole so the program prints `BoLt : 40`. Careful: some fills compile and still print the wrong thing.

```
class Robot {
    private String name;
    private int battery;
    public Robot(String (1), int charge) {
        name = (2);
        battery = charge;
    }
    public void work() {
        (3) -= 10;
    }
    public String status() {
        return name + ": " + battery;
    }
}

// in main, in another class
Robot r = new Robot("Bolt", 50);
r.work();
System.out.println(r.status());
```

THE POOL

A snippet may be used more than once, and not every snippet is needed.

name	battery	n	int battery	charge
------	---------	---	-------------	--------



FAIR QUESTION

If `name = name;` does nothing useful, why does it even compile?

Both sides are legal names in scope: they both mean the parameter. Assigning a variable to itself is allowed, just pointless. Java checks that names exist, not that you meant the right one.

PART C · 3 QUESTIONS

C

Catch the mistake

7

YOU'RE THE COMPILER

Each row runs in main (another class) right after `Seat t = new Seat(12.0);`. Does it compile? If it does, write the output. If not, say why.

```
class Seat {
    private double price;
    public Seat(double p) {
        price = p;
    }
    public double getPrice() { return price; }
}
```

Code in main	Compiles?	Output, or why not
1 <code>System.out.println(t.getPrice());</code>		
2 <code>System.out.println(t.price);</code>		
3 <code>System.out.println(price);</code>		
4 <code>for (int i = 0; i < 2; i++) { }</code> <code>for (int i = 7; i < 8; i++) {</code> <code>System.out.println(i);</code> <code>}</code>		
5 <code>int n = 3;</code> <code>if (n > 2) {</code> <code>int big = n * 2;</code> <code>}</code> <code>System.out.println(big);</code>		
6 <code>private int tip = 2;</code> <code>System.out.println(tip);</code>		

8 TWO TRUTHS AND A LIE

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

```
class Timer {
    private int seconds;
    public Timer(int start) {
        seconds = start;
    }
    public void tick() {
        int step = 1;
        seconds -= step;
    }
    public int getSeconds() { return seconds; }
}
// in main, in another class
Timer t = new Timer(10);
t.tick();
System.out.println(t.getSeconds());
```

- **1** It prints 9.
- **2** A new step is created every time tick runs.
- **3** getSeconds could just return start; instead.

THE LIE, REWRITTEN

9 WHO AM I?

Each scope idea gives a clue about itself. Name the idea.

Clue	I am...
I'm the part of a program where a variable's name can be used.	
I'm the code between a matching { and }.	
I'm declared inside a block and gone when it closes.	
I'm declared in a method header and start as a copy of an argument.	
A local name matches mine, so inside that method I'm hidden.	