

NAME _____

DATE _____

PERIOD _____

UNIT 3 · TOPIC 3.6 · WORKSHEET

Same Bucket

Follow buckets into methods and back out, catch a leaky return, and swap that doesn't.

About 30 minutes

9 questions

Cumulative through 3.6

TOPIC

3.6

**DIRECTIONS**

Work each puzzle in pencil. When code prints, write exactly what it prints. Every puzzle uses the Bucket class below, plus the small class shown in the puzzle. Some puzzles include pieces you won't need.

**KEY IDEA****Meet the Bucket class**

A Bucket is **mutable**: add changes how many liters it holds.

```
class Bucket {
    private String color;
    private int liters;
    public Bucket(String paint, int amount) {
        color = paint;
        liters = amount;
    }
    public String getColor() {
        return color;
    }
    public int getLiters() {
        return liters;
    }
    public void add(int amount) {
        liters = liters + amount;
    }
}
```

PART A · 1 QUESTION

A Warm up your brain



THINK FIRST

You pass your bucket to a method, and the method calls `add(6)` on its parameter. Does *your* bucket change? Guess in the margin; question 1 settles it.

1 PENCIL CHECK

Each row starts fresh with `Bucket blue = new Bucket("blue", 4);` and `Gardener g = new Gardener();`. Write what `blue.getLiters()` returns after the row's calls.

```
class Gardener {
    public void fill(Bucket b) {
        b.add(6);
    }
    public void swapOut(Bucket b) {
        b = new Bucket("green", 9);
    }
    public void fillCopy(Bucket b) {
        Bucket c = new Bucket(b.getColor(), b.getLiters());
        c.add(6);
    }
}
```

Call	<code>blue.getLiters()</code> after
<code>g.fill(blue);</code>	
<code>g.swapOut(blue);</code>	
<code>g.fillCopy(blue);</code>	
<code>g.fill(blue); then g.swapOut(blue);</code>	

PART B · 3 PUZZLES

B Puzzles

2 CODE TILES

bigger should return whichever bucket holds more (the first one on a tie): the bucket itself, not a copy. Build its body from four tiles. Two are decoys.

```
class Judge {
    public Bucket bigger(Bucket a, Bucket b) {
        // your tiles go here
    }
}
```

return b;	if (a.getLiters() >= b.getLiters()) {
return new Bucket("red", 7);	}
return a;	return a.getLiters();

main runs	Target output
<pre>Bucket x = new Bucket("red", 7); Bucket y = new Bucket("tan", 4); Bucket z = new Bucket("blue", 9); Judge j = new Judge(); Bucket big = j.bigger(x, y); big.add(1); System.out.println(big == x); System.out.println(x.getLiters()); System.out.println(j.bigger(x, z) == z);</pre>	<pre>true 8 true</pre>

THE BODY OF BIGGER

3 SNIPPET POOL

`pourAll` pours every liter from one bucket into the other and returns the bucket that now holds it all. `main` runs the lines below. Fill each hole from the pool so it prints `0 8` and then `true`.

```
class Pourer {
    public Bucket pourAll(Bucket from, Bucket to) {
        int amount = (1);
        to.add(amount);
        from.add((2));
        return (3);
    }
}

// in main:
Bucket a = new Bucket("red", 6);
Bucket b = new Bucket("tan", 2);
Bucket full = new Pourer().pourAll(a, b);
System.out.println(a.getLiters() + " " + b.getLiters());
System.out.println(full == b);
```

THE POOL

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

<code>from.getLiters()</code>	<code>from.liters</code>	<code>to.getLiters()</code>
<code>amount</code>	<code>-amount</code>	<code>from</code>
<code>to</code>	<code>new Bucket("tan", 8)</code>	



FAIR QUESTION

A String is an object too. Can a method change mine?

No. A `String` is **immutable**: no method can change its characters. If a method does `s = s + "!"`; to its `String` parameter, that builds a new `String` and points only the parameter at it. The caller's variable still refers to the original, so passing a `String` is always safe.

4 YOU'RE THE COMPILER

You are the compiler. Each method is added to the class named, and `main` calls it with `red` (5 liters) and `tan` (4 liters). Does it compile? If so, write what printing the call shows. If not, say why.

Method	Compiles?	Prints, or why not
1 In <code>Bucket</code> ; call <code>red.holdsMore(tan)</code> <pre>public boolean holdsMore(Bucket other) { return liters > other.liters; }</pre>		
2 In <code>Gardener</code> ; call <code>g.total(red, tan)</code> <pre>public int total(Bucket a, Bucket b) { return a.liters + b.liters; }</pre>		
3 In <code>Gardener</code> ; call <code>g.total(red, tan)</code> <pre>public int total(Bucket a, Bucket b) { return a.getLiters() + b.getLiters(); }</pre>		
4 In <code>Gardener</code> ; call <code>g.pick(red)</code> <pre>public Bucket pick(Bucket a) { return a.getLiters(); }</pre>		
5 In <code>Gardener</code> ; call <code>g.tag(red)</code> <pre>public String tag(Bucket a) { return a.getColor() + a.getLiters(); }</pre>		

PART C · 5 QUESTIONS

C

Catch the mistake

5

COPY OR LEAK?

A Shed keeps a private spare bucket. main runs the lines below the class. What prints?

```
class Shed {
    private Bucket spare;
    public Shed() {
        spare = new Bucket("gray", 1);
    }
    public Bucket getSpare() {
        return spare;
    }
    public Bucket copySpare() {
        return new Bucket(spare.getColor(), spare.getLiters());
    }
}

// in main:
Shed shed = new Shed();
Bucket s = shed.getSpare();
s.add(10);
Bucket c = shed.copySpare();
c.add(100);
System.out.println(shed.getSpare().getLiters() + " " + (s == c));
```

A 1 false

B 11 false

C 111 false

D 11 true

6

SPEC CHECK

The specification: `preview(Bucket b, int extra)` returns how many liters `b` would hold if `extra` were added, **without changing** `b`. Which body meets it?

A `b.add(extra); return b.getLiters();`

B `return b.getLiters() + extra;`

C `Bucket c = b; c.add(extra); return c.getLiters();`

D `int before = b.getLiters(); b.add(extra); return before + extra;`

7 TWO TRUTHS AND A LIE

main makes `Bucket a = new Bucket("red", 5);` and `Bucket b = new Bucket("tan", 2);`, calls `new Swapper().swap(a, b);`, and prints `a.getColor()`, a space, and `b.getColor()`. One claim is a lie. Circle it and rewrite it so it's true.

```
class Swapper {
    public void swap(Bucket x, Bucket y) {
        Bucket temp = x;
        x = y;
        y = temp;
    }
}
```

- **1** At the end of swap, x refers to the tan bucket and y to the red one.
- **2** No Bucket object changes state during the call.
- **3** The program prints tan red, because x and y refer to main's buckets.

THE LIE, REWRITTEN

8 FACE-OFF

Tonight: a Parameter and the Object it refers to argue about who's in charge. Fill in the blanks.

PARAMETER	When main passed you in, I got a _____ of your reference, not a copy of you.
OBJECT	So when you call add through it, I'm the one who _____, and main sees it too.
PARAMETER	But if someone assigns me a _____ bucket, only I move. main's variable still points at you.
OBJECT	Then mind your manners: don't change me unless your method's _____ says to.

9 WHO AM I?

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

Clue	I am...
I'm what a parameter of a class type really holds: an object's address.	
I'm a variable that refers to the same object as another variable.	
I'm a change a method makes to an object passed to it that nobody asked for.	
I'm the only way to get a separate copy of an object.	
Nobody can change me once I'm built, so passing me to a method is always safe.	