

NAME _____

DATE _____

PERIOD _____

UNIT 1 · TOPIC 1.6 · WORKSHEET

Update in Place

Trace a variable that keeps changing, pick the right operators, and catch the update that went the wrong way.

About 30 minutes

10 questions

Cumulative through 1.6

TOPIC

1.6

**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 · 3 QUESTIONS

A

Warm up your brain

1

YOU'RE THE JVM

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

```
int score = 10;
score += 5;
score *= 2;
System.out.println(score);
```

A 30**B** 20**C** 25**D** 15**THINK FIRST**

If a holds 10, does `a *= 1 + 2;` give 12 or 30? Guess in the margin; question 2 settles it.

2 PENCIL CHECK

Trace the code one line at a time. After each line runs, write the value stored in `a`.

```
1 int a = 7;
2 a += 5;
3 a -= 2;
4 a *= 1 + 2;
5 a /= 4;
6 a %= 4;
```

After line	a
1	
2	
3	
4	
5	
6	

3 MATCH-UP

In each line below, `x` starts at 6. Each line changes one thing. Match each line to the value `x` holds afterward, from the list below. Each value is used once.

Line	Code	x afterward
1	<code>x *= 2 + 1;</code>	
2	<code>x = x * 2 + 1;</code>	
3	<code>x += 2 * 1;</code>	
4	<code>x -= 2 - 1;</code>	

- P 5
- Q 13
- R 8
- S 18

PART B · 3 PUZZLES

B Puzzles

4 CODE TILES

Hopper types how many coins are in the jar, then finds 15 more, splits the pile evenly with a friend (an odd coin goes to the friend), and finally finds 1 more coin. The user types 21. Put the tiles in order between the lines shown so the program prints the output below. Two tiles are decoys and aren't used.

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

coins /= 2;	int coins = console.nextInt();
coins = 15;	coins++;
coins %= 2;	coins += 15;

After your tiles, the program ends with this line:

```
System.out.println(coins);
```

Target output

19

YOUR TILES, IN ORDER



WATCH OUT

`+=` is not a compound operator. `lives += 2;` compiles, but it means `lives = +2;`; it *replaces* `lives` with `2`. The operator symbol always comes first: `+=`.

5 SNIPPET POOL

Hopper's game score starts at 12. Hopper loses 3 points, then a bonus multiplies the score by 4, then a final hop earns 1 more point. Fill each hole with an operator from the pool so the program prints 37. Each operator is used at most once, and some aren't needed.

GameScore.java

```
int points = 12;
points (1) 3;
points (2) 4;
points (3);
System.out.println(points);
```

THE POOL

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

+=	-=	*=	/=
%=	++	--	

6 FACE-OFF

Tonight: /= and %= argue about who's better at sharing cookies. In each line, cookies starts at 14. Fill in the blanks.

/=	Run cookies /= 4; and cookies holds ____: the number of full bags of 4.
%=	Run cookies %= 4; and it holds ____: the cookies _____. Nobody leaves my party hungry.
/=	With an int variable, I keep only the _____ part. Make cookies a double holding 14.0, and I leave _____.
%=	Either way, we both do the math between the variable's _____ value and the right side, then store the result back in the _____.
/=	Just never ask me to divide an int by zero. That throws an _____.



FAIR QUESTION

Why bother with += when a = a + 5 already works?

They do exactly the same thing. The short form names the variable once, so there's one less chance to mistype a long name like `totalPointsThisRound`, and anyone reading it sees right away that the variable is being *updated* from its own value.

PART C · 4 QUESTIONS

C

Catch the mistake

7

YOU'RE THE COMPILER

You are the compiler. 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>int total;</code> <code>total += 5;</code>		
2 <code>int laps = 3;</code> <code>laps + 1;</code>		
3 <code>int laps = 3;</code> <code>laps++;</code> <code>System.out.println(laps);</code>		
4 <code>int score = 10;</code> <code>10 += score;</code>		
5 <code>int half = 9;</code> <code>half /= 2;</code> <code>System.out.println(half);</code>		

8 ONE-OPERATOR FIX

Hopper had 8 stamps, *got* 3 more, and then doubled the collection. The code prints 10, but it should print 22. Change exactly one operator to fix it.

```
1 int stamps = 8;
2 stamps -= 3;
3 stamps *= 2;
4 System.out.println(stamps);
```

LINE, OLD OPERATOR, NEW OPERATOR

9 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.

```
1 int big = Integer.MAX_VALUE;
2 big++;
3 double share = 7;
4 share /= 2;
```

- **1** After line 2, `big` holds `Integer.MIN_VALUE`.
- **2** `big++`; does the same thing as `big += 1`;
- **3** Line 4 does int division, because 2 is an `int`, so `share` ends up holding 3.0.

THE LIE, REWRITTEN

10 WRAP-UP**REFLECTION**

In one sentence: how are `score = bonus;` and `score += bonus;` different, and which one adds a bonus to a score?

YOUR ANSWER