

NAME _____

DATE _____

PERIOD _____

UNIT 2 · TOPIC 2.3 · WORKSHEET

Fork in the Code

Decide which lines run, rebuild a ticket booth, and catch the semicolon that ruins a forecast.

About 35 minutes

10 questions

Cumulative through 2.3

TOPIC

2.3

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints, one line per `println`. If nothing prints, write *nothing*. Some puzzles include pieces you won't need.

PART A · 2 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 coins = 40;
String msg = "Just looking";
if (coins >= 50) {
    coins -= 50;
    msg = "Bought a hat";
}
coins += 5;
System.out.println(msg + " " + coins);
```

A Just looking 45**B** Bought a hat -5**C** Just looking 40**D** Bought a hat 45**THINK FIRST**

Can a program with two separate `if` statements skip *both* bodies? Guess in the margin; question 2 settles it.

2 PENCIL CHECK

A museum works out a ticket fee. For each age, write the value of fee after the code runs.

```
int fee = 12;
if (age <= 12) {
    fee = 8;
}
if (age >= 65) {
    fee -= 3;
}
```

age	fee
8	
12	
13	
64	
65	



WATCH OUT

The parentheses around a condition aren't optional. `if score >= 70 {` won't compile; Java needs `if (score >= 70) {`. The condition inside them must be a boolean: a comparison, a boolean variable, or a call like `equals`.

PART B · 3 PUZZLES

B Puzzles

3 CODE TILES

At Hopper's movie booth, kids under 12 get a \$6 ticket and everyone else gets a \$10 ticket. Then *everyone* is told about the \$2 3D glasses. Put all six tiles where the comment is so the program prints the target output for both test ages.

```
int age = 12;    // Hopper tests age 12, then age 8
// tiles go here
```

System.out.println("3D glasses: \$2");	} else {
System.out.println("Adult: \$10");	if (age < 12) {
}	System.out.println("Kid: \$6");

Test	Target output
age is 12	Adult: \$10 3D glasses: \$2
age is 8	Kid: \$6 3D glasses: \$2

YOUR TILES, IN ORDER

4 SNIPPET POOL

Fill each numbered hole with a snippet from the pool so the program prints Hi , Hopper, then Pass, then 70. Each snippet is used at most once.

Checkup.java

```
String name = new String("Hopper");
int score = 70;
boolean late = false;
if (name (1) ) {
    System.out.println("Hi, Hopper");
}
if (score (2) 70) {
    System.out.println("Pass");
} else {
    System.out.println("Retake");
}
if ( (3) ) {
    score -= 5;
}
System.out.println(score);
```

THE POOL

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

>	late == false	.equals("hopper")
=	late	>=
.equals("Hopper")		



FAIR QUESTION

Why doesn't else get a condition of its own?

Because it doesn't need one. The else body runs exactly when the if condition is false, so the question has already been asked. Java checks the condition once and picks one of the two bodies.

5 MATCH-UP

Each variant starts with `int temp = 80;`. Match each one to what it prints, from the list below. Each output is used once.

Variant	Code	Output
1	<pre>if (temp > 75) { System.out.println("Hot"); } if (temp > 60) { System.out.println("Warm"); }</pre>	
2	<pre>if (temp > 75) { System.out.println("Hot"); } else { System.out.println("Warm"); }</pre>	
3	<pre>if (temp > 85) { System.out.println("Hot"); System.out.println("Warm"); }</pre>	
4	<pre>if (temp > 85) System.out.println("Hot"); System.out.println("Warm");</pre>	

- **P** Hot
- **Q** Warm
- **R** Hot then Warm
- **S** nothing prints

PART C · 5 QUESTIONS

C

Catch the mistake

6

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.

```
int score = 85;
String grade = "B";
if (score >= 90) {
    grade = "A";
} else {
    score += 5;
}
System.out.println(grade + " " + score);
```

- **1** It prints B 90.
- **2** The condition is checked once, before either body runs.
- **3** Once score reaches 90 in the else body, the if body runs too, so grade becomes "A".

THE LIE, REWRITTEN

7

ONE-TOKEN FIX

At 68 degrees this still prints Wear a coat before Have a good day; it should print only Have a good day. Change or delete exactly one token so it's right for every temperature.

```
1 int temp = 68;
2 if (temp < 50); {
3     System.out.println("Wear a coat");
4 }
5 System.out.println("Have a good day");
```

LINE, TOKEN, AND WHY IT FIXES IT

8 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
<pre>1 int x = 5; if (x = 5) { System.out.println("five"); }</pre>		
<pre>2 int x = 5; if (x > 3) System.out.println("big");</pre>		
<pre>3 int x = 5; if (x > 9); { System.out.println("big"); } else { System.out.println("small"); }</pre>		
<pre>4 int x = 5; String s; if (x > 3) { s = "big"; } System.out.println(s);</pre>		
<pre>5 boolean ready = true; if (ready) { System.out.println("Go"); }</pre>		

9 WHO AM I?

The parts of an if statement are playing a party game. Each gives a clue about itself. Name it.

Clue	I am...
I'm the Boolean expression in the parentheses after <code>if</code> .	
I'm the code that runs only when the condition is <code>true</code> .	
I have no condition of my own. I run when the <code>if</code> condition is <code>false</code> .	
I group several statements so they run, or get skipped, together.	
Put me right after <code>if (. . .)</code> and I quietly end the whole if statement.	
I'm the kind of selection where exactly one of two bodies runs.	

10 WRAP-UP



REFLECTION

In one sentence: when would you write two separate if statements instead of one if-else?

YOUR ANSWER