

NAME

DATE

PERIOD

UNIT 4 · TOPIC 4.12 · WORKSHEET

Every Box, In Order

Walk grids across and down, backward and by row, and catch the loop that changed only a copy.

About 30 minutes

9 questions

Cumulative through 4.12

TOPIC

4.12

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints; `print` stays on the same line and `println` ends it. Some puzzles include pieces you won't need.

**THINK FIRST**

Nested loops visit every element of a 2-by-3 grid. How many times does the *inner* loop's body run? How many times does the *outer* loop's body run? Guess in the margin; question 1 settles it.

PART A · 2 QUESTIONS

A Warm up your brain

1 PENCIL CHECK

Each row describes a traversal of g. Write the values in the order they are visited, or the count asked for.

```
int[][] g = { {1, 2, 3},
              {4, 5, 6} };
```

Traversal of g	Visits, in order
Row-major order	
Column-major order	
Row-major, but both loops count backward	
Column 1 only, top to bottom	
Row 1 only, right to left	
How many times the inner body runs in a full traversal	
How many times the outer body runs, row-major	
How many times the outer body runs, column-major	

2 YOU'RE THE JVM

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

```
String[][] w = { {"h", "p", "e"},
                 {"o", "p", "r"} };
for (int c = 0; c < w[0].length; c++) {
    for (int r = 0; r < w.length; r++) {
        System.out.print(w[r][c]);
    }
}
```

- A** hpeopr
 B hopper
- C** rpoeph
 D An ArrayIndexOutOfBoundsException

PART B · 3 PUZZLES

B Puzzles

3 CODE TILES

Put tiles in order after the declaration so the program prints each column of `g` on its own line, as shown. Two tiles are decoys.

```
int[][] g = { {1, 2, 3},
              {4, 5, 6} };
```

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

```
for (int c = 0; c < g.length; c++) {
```

```
for (int r = 0; r < g.length; r++) { System.out.print(" " + g[r][c]); }
```

```
}
```

```
System.out.print("Col " + c + ":");
```

```
for (int c = 0; c < g[0].length; c++) {
```

```
for (int r = 0; r < g.length; r++) { System.out.print(" " + g[c][r]); }
```

Target output

```
Col 0: 1 4
```

```
Col 1: 2 5
```

```
Col 2: 3 6
```

YOUR PROGRAM

4 SNIPPET POOL

Fill each hole from the pool so the program prints efcdab: the rows from bottom to top, each row left to right. Each snippet is used at most once, and some aren't needed.

```
String[][] t = { {"a", "b"},
                 {"c", "d"},
                 {"e", "f"} };
for (int r = (1); (2); r--) {
    for (int c = (3); c < t[0].length; c++) {
        System.out.print(t[r][c]);
    }
}
```

THE POOL

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

t.length	r > 0	1	t.length - 1
0	r >= 0	t[0].length - 1	

5 FACE-OFF

Tonight: the indexed for loop and the enhanced for loop argue about who should traverse a 2D array. Fill in the blanks.

INDEXED FOR	I can go row-major, column-major, even bottom to top. You only know one order: _____.
ENHANCED FOR	But I never get a bound wrong. My outer variable holds a whole _____ of the grid, so for an int[][] its type is _____.
INDEXED FOR	And when someone wants to add 10 to every element?
ENHANCED FOR	x += 10 on my inner variable changes only a _____. The array stays the same.
INDEXED FOR	Exactly. I assign to grid[r][c], so my changes are stored in the _____ itself.



WATCH OUT

An enhanced for loop hands you *values*, not positions. In `for (int x : row)`, writing `row[x]` uses the value as an index: it reads the wrong element, or crashes when the value is too big. If you need positions, use indexes.

PART C · 4 QUESTIONS

C

Catch the mistake

6

YOU'RE THE COMPILER

Each snippet starts with `int[][] g = { {1, 2, 3}, {4, 5, 6} };`. 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 for (int[] row : g) { for (int x : row) { System.out.print(x); } }</pre>		
<pre>2 for (int row : g) { System.out.print(row); }</pre>		
<pre>3 for (int[] row : g) { for (int x : g) { System.out.print(x); } }</pre>		
<pre>4 for (int[] row : g) { System.out.print(row.length); }</pre>		
<pre>5 for (int[] row : g) { for (int x : row) { x = 0; } } System.out.print(g[0][0]);</pre>		



FAIR QUESTION

Could an enhanced for loop print the rows bottom to top?

No. A nested enhanced for loop always goes through the rows first to last, and across each row first to last: row-major order, every time. For any other order, such as backward or column by column, you need indexes.

7

TWO TRUTHS AND A LIE

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

```
int[][] pts = { {3, 8},
                {5, 1} };
for (int[] row : pts) {
    for (int p : row) {
        p += 10;
    }
    row[0] = 0;
}
```

- **1** After the loops, `pts[1][1]` is still 1.
- **2** After the loops, `pts[0][0]` is 0.
- **3** After the loops, `pts[0][1]` is 18.

THE LIE, REWRITTEN

8 ONE-TOKEN FIX

This program prints `ad` and then crashes with an `ArrayIndexOutOfBoundsException`. It should print `abc` on one line and `def` on the next. Change exactly one token to fix it.

```
1 String[][] g = { {"a", "b", "c"},
2                 {"d", "e", "f"} };
3 for (int r = 0; r < g.length; r++) {
4     for (int c = 0; c < g[0].length; r++) {
5         System.out.print(g[r][c]);
6     }
7     System.out.println();
8 }
```

WHY IT CRASHES, AND YOUR FIX (LINE, OLD TOKEN, NEW TOKEN)

9 WHO AM I?

Some 2D traversal ideas are playing a party game. Each gives a clue about itself. Name it.

Clue	I am...
I finish a whole row before I move down to the next one.	
I finish a whole column before I move right to the next one.	
I'm the type of the outer variable in a nested enhanced for loop over an <code>int[][]</code> .	
I'm the bound for a loop that counts the columns.	
I'm what an enhanced for loop's inner variable holds. Change me, and the array never notices.	