

NAME

DATE

PERIOD

UNIT 4 · TOPIC 4.11 · WORKSHEET

Grid Lock

Build grids, pick out boxes, borrow rows, and catch the index that wandered off the edge.

About 30 minutes

9 questions

Cumulative through 4.11

TOPIC

4.11

**DIRECTIONS**

Work each puzzle in pencil. Sketch each 2D array as a grid before you trace it. For code, write exactly what prints. Some puzzles include pieces you won't need.

**THINK FIRST**

A grid has 3 rows and 5 columns. What is the index of its last row, and of its last column? Guess in the margin; question 1 checks the same idea on a different grid.

PART A · 2 QUESTIONS

A Warm up your brain

1 PENCIL CHECK

Draw `g` as a grid in the margin, then fill in each value. If an expression would crash, write *crash*.

```
int[][] g = { {4, 9, 2, 7},
              {3, 8, 6, 1},
              {5, 0, 11, 10} };
```

Expression	Value
<code>g[1][2]</code>	
<code>g[2][1]</code>	
<code>g.length</code>	
<code>g[0].length</code>	
<code>g[2][g[0].length - 1]</code>	
<code>g[g.length - 1][0] + g[0][3]</code>	
<code>g[1].length</code>	
<code>g[3][0]</code>	

2 YOU'RE THE JVM

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

```
double[][] fees = new double[2][3];
boolean[][] paid = new boolean[2][3];
fees[1][2] = 4.5;
paid[0][1] = true;
System.out.println(fees[0][2] + fees[1][2] + " " + paid[1][1]);
```

A 4.5 false

B 4.5 true

C 4.5 null

D 0.04.5 false



WATCH OUT

Swapping the two indices doesn't always crash. On a 4-by-4 board, `board[col][row]` uses two valid indices and quietly reads the wrong square. Say it as you write it: row first, then column.

PART B · 4 PUZZLES

B

Puzzles

3

CODE TILES

Someone knocked this program off the fridge. Put the tiles back in order so it prints the output below. One tile is a decoy and isn't used.

```
System.out.println(row.length);
int[] row = pad[pad.length];
row[0] = row[1] * 2;
int[][] pad = { {1, 2}, {3, 4}, {5, 6} };
System.out.println(pad[2][0]);
int[] row = pad[pad.length - 1];
```

Target output

12

2

YOUR PROGRAM

4 SNIPPET POOL

Fill each hole from the pool so the program prints the dish in the bottom-right corner and then the grid's size. Each snippet is used at most once, and some aren't needed. Target output: `figs`, then 2 rows, 4 columns.

```
String[][] menu = { {"taco", "soup", "pie", "plum"},
                    {"rice", "stew", "cake", "figs"} };
int lastRow = ;
int lastCol = ;
System.out.println(menu[lastRow][lastCol]);
int rows = ;
int cols = ;
System.out.println(rows + " rows, " + cols + " columns");
```

THE POOL

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

<code>menu.length - 1</code>	<code>menu.length()</code>	<code>menu[0].length</code>
<code>menu[2].length</code>	<code>menu.length</code>	<code>menu[0].length - 1</code>



FAIR QUESTION

Why count the columns with row 0? Wouldn't any row work?

In this course, yes: every row of a 2D array has the same length, so any row gives the same answer. Row 0 is the habit because it always exists in a grid with at least one row, while a row like `grid[5]` might not.

5 YOU'RE THE COMPILER

You are the compiler. For each snippet, decide whether it compiles. If it does, write what it prints (if anything). If it doesn't, say why in a few words.

Snippet	Compiles?	Output, or why not
1 <code>int[] g = { {1, 2}, {3, 4} };</code>		
2 <code>int[][] g = { {1, 2}, {3, 4} };</code> <code>int x = g[1];</code>		
3 <code>int[][] g = new int[2][3];</code> <code>System.out.println(g[1][2]);</code>		
4 <code>int[][] g = { {1, 2}, {3, 4} };</code> <code>System.out.println(g[1, 0]);</code>		
5 <code>String[][] g = new String[2][2];</code> <code>System.out.println(g[0][1]);</code>		
6 <code>int[][] g = { {1, 2, 3}, {4, 5, 6} };</code> <code>int[] r = g[0];</code> <code>System.out.println(r.length);</code>		

6 FACE-OFF

Tonight: `grid.length` and `grid[0].length` argue about who matters more. Fill in the blanks.

GRID.LENGTH	I count the elements of the outer array, so I tell you how many _____ there are.
GRID[0].LENGTH	I'm the length of one row, so I tell you how many _____ there are.
GRID.LENGTH	For new <code>int[3][5]</code> , I'm _____ and you're _____.
GRID[0].LENGTH	The last valid column index is me minus _____. Use me as an index and Java throws an _____.
GRID.LENGTH	On a square grid we're equal, so mixing us up stays hidden. On that 3-by-5 grid, using me minus 1 for the last column picks column _____ instead of column 4.

PART C · 3 QUESTIONS

C

Catch the mistake

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.

```
String[][] names = new String[2][2];  
names[0][0] = "Ivy";  
names[1][0] = names[0][0] + "!";
```

- **1** `names[1][0].length()` is 4.
- **2** `names[0][1]` holds the empty string "", so `names[0][1].length()` is 0.
- **3** `System.out.println(names[1][1]);` prints null.

THE LIE, REWRITTEN

8

ONE-NUMBER FIX

This program should print `Dee Eli`, but it crashes with an `ArrayIndexOutOfBoundsException`. Change exactly one number to fix it.

```
1 String[][] seats = new String[3][2];  
2 seats[2][1] = "Dee";  
3 seats[1][2] = "Eli";  
4 System.out.println(seats[2][1] + " " + seats[1][1]);
```

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

9 WHO AM I?

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

Clue	I am...
I'm the number of rows in <code>grid</code> .	
Put one pair of brackets after <code>grid</code> and you get me: a whole row, as a 1D array.	
I'm in every box of <code>new String[2][3]</code> until you fill it.	
I'm in every box of <code>new boolean[4][4]</code> .	
I'm the index that always comes first in <code>grid[a][b]</code> .	
I'm thrown when a row or column index is negative or too big.	