

NAME

DATE

PERIOD

UNIT 4 · TOPIC 4.3 · WORKSHEET

Mind the Index

Count from zero, read the defaults, and catch the crash.

About 30 minutes

9 questions

Cumulative through 4.3

TOPIC

4.3



DIRECTIONS

Work each puzzle in pencil. For Java code, write exactly what prints. If a program crashes, write what prints before the crash and name the exception.



THINK FIRST

An array holds 5 elements. What is the last one's index? Question 1 checks.

PART A · 3 QUESTIONS

A

Build an array

1

PENCIL CHECK

A swimmer's laps: `int[] laps = {12, 9, 15, 7, 11};` Write the value of each expression.

Expression	Value
<code>laps[0]</code>	
<code>laps[3]</code>	
<code>laps.length</code>	
<code>laps[laps.length - 1]</code>	
<code>laps[1] + laps[2]</code>	
<code>laps[laps[3] - 5]</code>	

2 PENCIL CHECK

Each statement creates an array. Write what is stored in element [0] right after it runs, and the array's length.

Statement	Element [0]	Length
<code>int[] goals = new int[4];</code>		
<code>double[] tips = new double[2];</code>		
<code>boolean[] seen = new boolean[6];</code>		
<code>String[] pets = new String[3];</code>		
<code>int[] bins = {0, 5, 2, 8};</code>		

3 YOU'RE THE COMPILER

You are the Java compiler. For each snippet, write *yes* or *no* for whether it compiles. If it compiles, write what happens when it runs. If it doesn't, say why.

Snippet	Compiles?	Output, or why not
<code>int[] ages = new double[4];</code>		
<code>int[] nums; nums = {1, 2, 3};</code>		
<code>int[] a = {4, 8}; System.out.println(a.length());</code>		
<code>int[] a = {4, 8}; System.out.println(a[2]);</code>		
<code>String[] s = {"hi", "hey"}; System.out.println(s[1].length());</code>		
<code>int[] a = {5, 6}; a = new int[3]; System.out.println(a[0]);</code>		

PART B · 3 QUESTIONS

B Read it, change it, crash it

4 YOU'RE THE JVM

A game keeps points for four players. Run this code in your head. What prints?

```
1 int[] pts = new int[4];
2 pts[1] = 6;
3 pts[3] = pts[1] + 2;
4 pts[1]++;
5 System.out.println(pts[0] + " " + pts[1] + " " + pts[3]);
```

A 0 7 8

B 0 7 9

C 6 7 8

D 0 6 8

5 SNIPPET POOL

Bus stops, in order. Fill each hole from the pool so the program prints Elm, then Bay, then Maple.
5. Each snippet is used at most once.

```
String[] stops = {"Elm", "Oak", "Pine", "Main", "Bay"};
System.out.println(stops[ (1) ]);
System.out.println(stops[ (2) ]);
stops[ (3) ] = "Maple";
System.out.println(stops[2] + " " + stops.length);
```

THE POOL

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

1	stops.length	0	3
stops.length - 1	5	2	

6 CRASH COURSE

A theater counts the seats sold in three sections. This program compiles. Run it in your head.

```
1 int[] seats = {14, 22, 9};
2 System.out.println(seats[0]);
3 seats[2] = seats[0] + seats[1];
4 System.out.println(seats[2]);
5 System.out.println(seats[3]);
6 System.out.println("done");
```

EVERYTHING IT PRINTS, WHAT HAPPENS ON LINE 5, AND A NEW LINE 5 THAT PRINTS THE LAST ELEMENT FOR AN ARRAY OF ANY LENGTH

**WATCH OUT**

`arr[-1]` is not a shortcut for the last element, the way it is in some other languages. In Java, an index below 0 crashes the program, just like an index that is too big.

**FAIR QUESTION****Why does Java start counting at 0 instead of 1?**

An index tells Java how many steps to take from the start of the array. The first element is 0 steps from the start, the next is 1 step away, and so on. You met the same rule with the characters of a `String`.

PART C · 3 QUESTIONS

C

Size and references

7

TWO TRUTHS AND A LIE

A school stores lockers in an array. Two claims about this program are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
1  class Locker {
2      private int items;
3
4      public void add(int n) {
5          items += n;
6      }
7
8      public int getItems() {
9          return items;
10     }
11 }
12
13 public class Hallway {
14     public static void main(String[] args) {
15         Locker[] row = new Locker[3];
16         row[0] = new Locker();
17         row[1] = row[0];
18         row[1].add(5);
19         System.out.println(row[0].getItems());
20         row[2].add(1);
21     }
22 }
```

- **1** Line 19 prints 5, because row[0] and row[1] refer to the same Locker.
- **2** Line 20 throws a NullPointerException, because row[2] holds null.
- **3** Line 15 creates three Locker objects, one in each slot.

THE LIE, REWRITTEN

8 FACE-OFF

Tonight: an Array and a String compare notes. Fill in the blanks.

STRING	Want my size? Call my method, <code>Length()</code> , parentheses and all.
ARRAY	Mine's an attribute, not a method. You write <code>Length</code> with _____.
STRING	Use an index past my last character and I throw a <code>StringIndexOutOfBoundsException</code> .
ARRAY	Same idea. An index below 0, or at least my length, throws an _____.
STRING	At least you can grow. Just squeeze in another element.
ARRAY	Never. My length is fixed when I'm _____. Point my variable at a new array, and my own length stays the _____.
STRING	Well, you can't change one of my characters at all.
ARRAY	Ha! Any of my elements can be changed, using its index in _____.

9 WHO AM I?

Some array ideas are playing a party game. Each gives a clue about itself. Name the idea.

Clue	I am...
I'm one value stored in an array.	
I'm an element's position, and I start at 0.	
I'm values in curly braces that create an array and fill it in one step.	
I'm what every slot of new <code>String[4]</code> holds before you store anything.	
I'm read with no parentheses, and I can never change.	
I'm thrown when an index is below 0 or at least the length.	