

NAME _____

DATE _____

PERIOD _____

UNIT 4 · TOPIC 4.4 · WORKSHEET

Lily Pad Loops

Hop through arrays forward, backward and by twos, and catch the loop variable that's only holding a copy.

About 30 minutes

10 questions

Cumulative through 4.4

TOPIC

4.4

**DIRECTIONS**

Work each puzzle in pencil. For code, write exactly what prints. You may use the Java Quick Reference. Some puzzles include pieces you won't need.

PART A · 3 QUESTIONS

A

Warm up your brain

1

YOU'RE THE JVM

Hopper hops across a row of lily pads, landing only where the loop says. Run this code in your head. What prints? (Each value is followed by a space.)

```
int[] pads = {3, 8, 5, 1, 9, 4};
for (int i = 1; i < pads.length - 1; i += 2) {
    System.out.print(pads[i] + " ");
}
```

A 8 1**B** 8 1 4**C** 3 5 9**D** 8 5 1 9**THINK FIRST**

`int[] lily = {2, 4, 6};` runs, then `for (int x : lily) { x = x * 10; }`. Does lily now hold 20, 40 and 60? Guess in the margin; question 6 settles it.

2 PENCIL CHECK

`int[] a = new int[6];` has indexes 0 through 5. Each loop's body uses `a[i]`. Write the indexes each loop visits, in order. If the loop crashes, say at which index.

Loop	Indexes visited
<code>for (int i = 0; i < a.length; i++)</code>	
<code>for (int i = 1; i < a.length; i += 2)</code>	
<code>for (int i = a.length - 1; i >= 3; i--)</code>	
<code>int i = 0;</code> <code>while (i < a.length) { ... i += 3; }</code>	
<code>for (int i = 0; i <= a.length; i++)</code>	

3 YOU'RE THE COMPILER

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

Snippet	Compiles?	Output, or why not
1 <code>double[] w = {1.5, 2.0};</code> <code>for (int x : w) {</code> <code>System.out.println(x);</code> <code>}</code>		
2 <code>String[] s = {"ox", "emu"};</code> <code>for (String t : s) {</code> <code>System.out.println(t.length());</code> <code>}</code>		
3 <code>int[] h = {4, 7};</code> <code>System.out.println(h.length());</code>		
4 <code>int[] h = {4, 7};</code> <code>for (int i = 0; i <= h.length; i++) {</code> <code>System.out.println(h[i]);</code> <code>}</code>		
5 <code>int[] h = {4, 7};</code> <code>for (int x : h) {</code> <code>System.out.println(h[x]);</code> <code>}</code>		

PART B · 4 PUZZLES

B Puzzles

4 CODE TILES

A depth sensor's readings end early at the first 0, if there is one. `showUntilZero` prints each reading up to, but not including, the first 0, and must work when there is no 0 at all. Build its body from five tiles. One tile is a decoy.

```
public static void showUntilZero(int[] d) {
    // your tiles go here
}
```

<code>System.out.println(d[i]);</code>	<code>while (d[i] != 0 && i < d.length) {</code>
<code>int i = 0;</code>	<code>}</code>
<code>i++;</code>	<code>while (i < d.length && d[i] != 0) {</code>

main runs	Target output
<code>int[] a = {4, 7, 0, 2};</code>	4
<code>int[] b = {6, 1};</code>	7
<code>showUntilZero(a);</code>	--
<code>System.out.println("--");</code>	6
<code>showUntilZero(b);</code>	1

THE BODY OF SHOWUNTILZERO

5 SNIPPET POOL

Hopper's jump lengths are in `double[] jumps = {1.5, 2.0, 0.5};`. The enhanced for loop `for (double j : jumps) { System.out.println(j * 2); }` prints 3.0, 4.0 and 1.0. Fill each hole from the pool so this indexed loop prints exactly the same thing. Not every snippet is used.

```
double[] jumps = {1.5, 2.0, 0.5};
for (int i = (1); i (2); i++) {
    double j = (3);
    System.out.println(j * 2);
}
```

THE POOL

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

0	jumps[i]	<= jumps.length	1
i	< jumps.length	jumps.length - 1	< jumps.length()

6 MATCH-UP

Each loop runs on a fresh array `int[] f = {1, 2, 3};`. Match each loop to what `f` holds afterward, from the list below. Each answer is used once.

Loop	Code	f holds
1	<pre>for (int x : f) { x = x * 10; }</pre>	
2	<pre>for (int i = 0; i < f.length; i++) { f[i] = f[i] * 10; }</pre>	
3	<pre>for (int i = f.length - 1; i > 0; i--) { f[i] = f[i] * 10; }</pre>	
4	<pre>for (int x : f) { f[0] = f[0] + x; }</pre>	

P 1 20 30 **Q** 10 20 30 **R** 7 2 3 **S** 1 2 3



FAIR QUESTION

If the enhanced for loop is shorter, why not use it every time?

Because it hides the index. It always goes forward through every element, and it never tells you where you are. When a task needs a position ("Day 3"), a different order, only part of the array, or a new value stored in a slot, you need an indexed for or while loop.

7

TWO TRUTHS AND A LIE

Here is a small Frog class, and the code main runs with it. Two claims are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
class Frog {
    private String name;
    private int hops;
    public Frog(String n) {
        name = n;
        hops = 0;
    }
    public void hop() { hops++; }
    public String getName() { return name; }
    public int getHops() { return hops; }
}
```

```
// in main:
Frog[] pond = {new Frog("Ada"), new Frog("Bo")};
for (Frog f : pond) {
    f.hop();
    f = new Frog("Cy");
    f.hop();
}
```

- **1** After the loop, `pond[0].getHops()` returns 1.
- **2** After the loop, `pond[1].getName()` returns "Cy".
- **3** Two Cy frogs are made, and each one hops once, but no element of pond refers to either.

THE LIE, REWRITTEN

PART C · 3 QUESTIONS

C Catch the mistake

8 ONE-TOKEN FIX

This loop should print the scores backward: 9, 3, 7. It prints only 9 and 3. Change exactly one token to fix it.

```
1 int[] scores = {7, 3, 9};
2 for (int i = scores.length - 1; i > 0; i--) {
3     System.out.println(scores[i]);
4 }
```

LINE, OLD TOKEN, NEW TOKEN

9 FACE-OFF

Tonight: the Enhanced For Loop and the Indexed For Loop argue about who's the better way to traverse an array. Fill in the blanks.

ENHANCED	Give me an array and I visit every element, first to last. Nobody has to write an _____.
INDEXED	And on each pass you hold only a _____ of the element. Assign to your variable and the array never notices.
ENHANCED	At least I can't run off the end. You crash with an _____ the moment someone writes <code><=</code> instead of <code><</code> .
INDEXED	Worth it. I can start in the middle, skip by twos, or go _____. You only go one way.
ENHANCED	When my elements are objects, my variable holds a _____, so calling a method on it changes the real object.
INDEXED	Sure. But to store a new value in a slot, you need <code>arr[i] =</code> , and you never have an _____.

**WATCH OUT**

An out-of-bounds index is a **runtime** error, not a compile error. If `t` has length 3 and `k` holds 3, then `t[k] = 1`; still compiles, because the compiler doesn't track what `k` holds. It crashes only when that line runs.

10**WRAP-UP****REFLECTION**

In one sentence: give one task where an enhanced for loop works fine, and one where you must use an indexed loop.

YOUR ANSWER