

Lily Pad Loops

Array traversals



Five temperatures, or five hundred?

PAUSE AND TRY

Print every value in this array. How many lines of code did you need?

```
int[] temps = {61, 64, 58, 70, 66};
```

Now imagine the array holds 500 temperatures. What changes from one print line to the next? What stays the same?



Talk with a partner. Write your answer on paper first.

By the end of class you can...

- 1 Use a for loop and the array's length to visit every element by its index.
- 2 Traverse part of an array, traverse it backward, or skip elements, without going out of bounds.
- 3 Traverse an array with a while loop, including one that can stop early.
- 4 Read and write an enhanced for loop, and explain why assigning to its variable never changes the array.
- 5 Explain what happens when an enhanced for loop calls a method on objects in an array, and rewrite any enhanced for loop as an indexed loop.

1

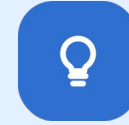
PART 1

Visiting every element

The loop counter is the index

INDEXED FOR LOOP

```
int[] temps = {61, 64, 58, 70};  
for (int i = 0; i < temps.length; i++) {  
    System.out.println(i + ": " + temps[i]);  
}
```



Traversing an array is using a loop to reach its elements in order. An indexed loop reaches each one through its **index**.

OUTPUT

```
0: 61  
1: 64  
2: 58  
3: 70
```

Start at 0

the first index

Test `i < temps.length`

the last index is length - 1

Update `i++`

one element at a time

One step past the end



The last valid index is **length - 1**. A test of **i <= temps.length** runs one pass too many and throws an `ArrayIndexOutOfBoundsException`.

```
int[] temps = {61, 64, 58, 70};  
for (int i = 0; i <= temps.length; i++) {  
    System.out.println(temps[i]);  
}
```

OUTPUT

```
61  
64  
58  
70
```

THEN THE PROGRAM CRASHES

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException:  
Index 4 out of bounds for length 4
```

Which elements does each loop visit?

PAUSE AND TRY

temps holds {61, 64, 58, 70, 66}. Write what each loop prints.

LOOP A

```
for (int i = temps.length - 1;  
     i >= 0; i -= 2) {  
    System.out.println(temps[i]);  
}
```

LOOP B

```
for (int i = 1;  
     i < temps.length - 1; i++) {  
    System.out.println(temps[i]);  
}
```

A large, bold, teal-colored number '2' is centered within a white rounded square with slightly irregular edges, giving it a hand-drawn or sticker-like appearance.

2

PART 2

Backward, partway, and with while

Same array, two more traversals

BACKWARD FOR LOOP

```
for (int i = temps.length - 1; i >= 0; i--) {  
    System.out.println(temps[i]);  
}
```

OUTPUT

```
70  
58  
64  
61
```

WHILE LOOP, FORWARD

```
int i = 0;  
while (i < temps.length) {  
    System.out.println(temps[i]);  
    i++;  
}
```

OUTPUT

```
61  
64  
58  
70
```

Both use `int[] temps = {61, 64, 58, 70};`. A while loop needs its own index variable: set it, test it, update it.

A while loop that can stop early

PRINT NAMES UNTIL "DONE"

```
String[] line = {"Ana", "Ben", "done", "Cal"};
int i = 0;
while (i < line.length && !line[i].equals("done")) {
    System.out.println(line[i]);
    i++;
}
```

OUTPUT

```
Ana
Ben
```



Put the index test first. If the array were {"Ana", "Ben"} with no "done", the test `i < line.length` is false when `i` is 2, so Java never evaluates `line[2]`. Written the other way around, the loop reads `line[2]` first and throws an `ArrayIndexOutOfBoundsException`.

A large, white, rounded square containing the number '3' in a bold, teal, sans-serif font.

3

PART 3

The enhanced for loop

For each int t in temps

ENHANCED FOR LOOP

```
int[] temps = {61, 64, 58, 70};  
for (int t : temps) {  
    System.out.println(t);  
}
```

OUTPUT

```
61  
64  
58  
70
```

Pass	t holds a copy of	Value
1	temps[0]	61
2	temps[1]	64
3	temps[2]	58
4	temps[3]	70



The header holds the **enhanced for loop variable**. Each pass, it is assigned a **copy** of one element, without using an index.

Changing the copy does not change the array



Assigning a new value to the enhanced for loop variable **never** changes the value stored in the array.

CHANGES ONLY T

```
for (int t : temps) {  
    t = t + 10;  
}  
System.out.println(temps[0]);
```

OUTPUT

61

CHANGES THE ARRAY

```
for (int i = 0; i < temps.length; i++) {  
    temps[i] = temps[i] + 10;  
}  
System.out.println(temps[0]);
```

OUTPUT

71

What prints?

PAUSE AND TRY

Write the exact output, one line per println.

```
String[] words = {"cat", "owl", "emu"};
for (String w : words) {
    w = w + "!";
    System.out.println(w);
}
System.out.println(words[0]);
```

YOUR OUTPUT

A large, white, rounded square containing a bold, dark teal number '4'.

4

PART 4

Arrays of objects

A copy of a reference still reaches the object

PLANT.JAVA

```
class Plant {  
    private int height;  
    public Plant(int h) {  
        height = h;  
    }  
    public void water() {  
        height += 2;  
    }  
    public int getHeight() {  
        return height;  
    }  
}
```

IN MAIN

```
Plant[] garden = {new Plant(5), new Plant(8)};  
for (Plant p : garden) {  
    p.water();  
}  
for (Plant p : garden) {  
    p = new Plant(1);  
}  
System.out.println(garden[0].getHeight());  
System.out.println(garden[1].getHeight());
```

OUTPUT

7
10

p.water() changes the object. **p = new Plant(1)** changes only p.

Which loop puts new plants in the garden?

PAUSE AND TRY

garden starts as {new Plant(5), new Plant(8)}. What does each version print?

LOOP A

```
for (Plant p : garden) {  
    p = new Plant(1);  
}  
System.out.println(  
    garden[0].getHeight());
```

LOOP B

```
for (int i = 0; i < garden.length; i++) {  
    garden[i] = new Plant(1);  
}  
System.out.println(  
    garden[0].getHeight());
```

A large white square with rounded corners, tilted slightly to the right, containing a large teal number '5'.

5

PART 5

Choosing and rewriting

One traversal, three ways

```
for (String p : pets) {  
    System.out.println(p);  
}
```

```
for (int i = 0; i < pets.length; i++) {  
    String p = pets[i];  
    System.out.println(p);  
}
```

```
int k = 0;  
while (k < pets.length) {  
    String p = pets[k];  
    System.out.println(p);  
    k++;  
}
```

```
String[] pets = {"cat", "owl", "emu"};
```

OUTPUT

```
cat  
owl  
emu
```

Each loop prints the same three lines.



To rewrite, make the first line of the body **String p = pets[i];** Then the rest of the body stays the same.

Which loop should you choose?

The task needs to...	Use	Why
Use each value, first to last	Enhanced for	No index to get wrong
Know the position ("Day 3")	Indexed	The enhanced loop never shows the index
Go backward, skip, or do part	Indexed	The enhanced loop always visits all, in order
Store new values in the array	Indexed	Only <code>arr[i] = ...</code> changes an element
Call a method on each object	Either	The loop variable refers to the real object

Words to know

Traverse

Use a loop to reach all of an array's elements, or an ordered part, one after another.

Indexed loop

A for or while loop whose counter is used as the index of each element.

Enhanced for loop

A loop that visits every element, first to last, without an index.

Loop variable

In an enhanced for loop, it holds a copy of one element on each pass.

length - 1

The last valid index of any array.

Out of bounds

An index below 0, or at least the length. Using one crashes the program.

Copy or the real thing?



Explain to your partner why an enhanced for loop can't double every value in an int array.

SENTENCE STARTER

The loop variable only holds a _____, so to change the array I need _____.

Next: the Lily Pad Loops puzzle worksheet.