

The Array Workshop

Implementing array algorithms



One array, many questions

PAUSE AND TRY

What questions could a teacher answer about this array with one loop?

```
int[] scores = {72, 95, 88, 62, 93};
```

Try to list at least four. For each one, ask:

- What do I keep track of while I loop?
- Could I stop before the end?

YOUR LIST

By the end of class you can...

- 1 Compute a sum or an average, and find a minimum or maximum (and where it is).
- 2 Count the elements with a property, and decide whether at least one or all of them have it.
- 3 Visit every pair of neighbors without going past the end.
- 4 Use a loop inside a loop to decide whether an array has a duplicate.
- 5 Shift, rotate, and reverse an array without losing any elements.

A large, white, rounded square containing the number '1' in a bold, teal, sans-serif font.

1

PART 1

Sum, average, and the best

An accumulator collects as you go

SUM AND AVERAGE

```
int[] scores = {72, 95, 88, 62};
int sum = 0;
for (int s : scores) {
    sum += s;
}
double avg = (double) sum / scores.length;
System.out.println(sum);
System.out.println(avg);
```

OUTPUT

```
317
79.25
```



An **accumulator** starts **before** the loop and is updated **inside** it, once per element.

Average = sum / count
The count is scores.length.

Cast before dividing, or int division drops the .25.

Keep the index of the best

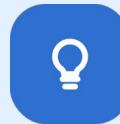
SCORES IS {72, 95, 88, 62}

```
int best = 0;
for (int i = 1; i < scores.length; i++) {
    if (scores[i] > scores[best]) {
        best = i;
    }
}
System.out.println(best);
System.out.println(scores[best]);
```

OUTPUT

1
95

i	scores[i]	scores[best]	best
-		72	0
1	95	72	1
2	88	95	1
3	62	95	1



Start the best at **element 0**, never at 0 or a guess. A minimum works the same way with <.

An average that lost its decimals



An **int** sum divided by an **int** length is int division. Cast one of them to double before you divide.

SUM IS 317, SCORES.LENGTH IS 4

```
System.out.println(sum / scores.length);  
System.out.println((double) sum / scores.length);
```

OUTPUT

```
79  
79.25
```

Also too late: `double avg = sum / scores.length;` stores 79.0, because the int division happens first and only then is 79 widened to a double.

A large, bold, teal-colored number '2' centered within a white rounded square with a slight drop shadow.

2

PART 2

Count, at least one, and all

Counters and flags

HOW MANY 90S?

```
int high = 0;
for (int s : scores) {
    if (s >= 90) {
        high++;
    }
}
System.out.println(high);
```

OUTPUT

2

scores is {72, 95, 88, 62, 93}

AT LEAST ONE, AND ALL

```
boolean anyFail = false;
boolean allPass = true;
for (int s : scores) {
    if (s < 65) {
        anyFail = true;
        allPass = false;
    }
}
System.out.println(anyFail);
System.out.println(allPass);
```

OUTPUT

true
false

Count the evens... or the even positions?

PAUSE AND TRY

Both counters compile. What are a and b at the end?

```
int[] nums = {7, 8, 3, 6, 5};  
int a = 0;  
int b = 0;  
for (int i = 0; i < nums.length; i++) {  
    if (i % 2 == 0) { a++; }  
    if (nums[i] % 2 == 0) { b++; }  
}
```

a = _____

b = _____

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

3

PART 3

Pairs and duplicates

KEY IDEA

Neighbors: i and $i + 1$

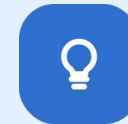
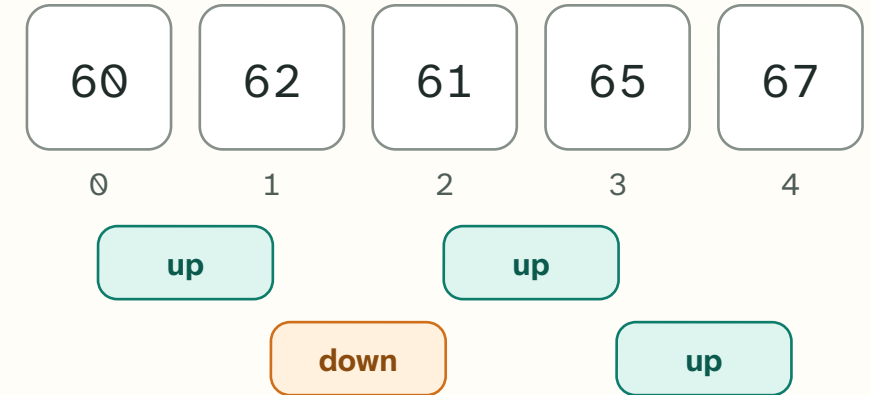
COUNT THE RISES

```
int[] temps = {60, 62, 61, 65, 67};
int rises = 0;
for (int i = 0; i < temps.length - 1; i++) {
    if (temps[i] < temps[i + 1]) {
        rises++;
    }
}
System.out.println(rises);
```

OUTPUT

3

THE FOUR NEIGHBOR PAIRS



An array of length n has $n - 1$ neighbor pairs. Stop at $i < \text{length} - 1$ so $i + 1$ is always valid.

Duplicates: every pair, once

IS ANY NAME REPEATED?

```
String[] names = {"Ada", "Bo", "Cy", "Bo"};
boolean dup = false;
for (int i = 0; i < names.length; i++) {
    for (int j = i + 1; j < names.length; j++) {
        if (names[i].equals(names[j])) {
            dup = true;
        }
    }
}
System.out.println(dup);
```

OUTPUT

true

i	j values	Pairs compared
0	1, 2, 3	(0,1) (0,2) (0,3)
1	2, 3	(1,2) (1,3)
2	3	(2,3)
3	none	



Start the inner loop at **i + 1**: no element meets itself, and no pair is checked twice.

How many checks?

PAUSE AND TRY

An array has 5 elements. Count the comparisons each loop makes.

A

```
for (int i = 0; i < a.length - 1; i++) {  
    // compare a[i] with a[i + 1]  
}
```

B

```
for (int i = 0; i < a.length; i++) {  
    for (int j = i + 1; j < a.length; j++) {  
        // compare a[i] with a[j]  
    }  
}
```

A: ____ checks

B: ____ checks

Bonus: what goes wrong if loop B starts j at 0?

A large, bold, teal-colored number '4' is centered within a white square with rounded corners. The square is slightly tilted to the right.

4

PART 4

Shift, rotate, and reverse

Rotate right: save, shift, place

ROTATE RIGHT BY ONE

```
int[] q = {3, 6, 9, 12};  
int last = q[q.length - 1];  
for (int i = q.length - 1; i > 0; i--) {  
    q[i] = q[i - 1];  
}  
q[0] = last;
```

OUTPUT

12 3 6 9

(Printed with an enhanced for loop and print, one value and a space each.)

1 Save last = 12

2 Shift back to front: 3 3 6 9

3 Place q[0] = last: 12 3 6 9



Loop in the direction **opposite** to the way the elements travel.

Forgetting to save the one that falls off



A shift overwrites one element. In a rotation, **save it in a variable before the loop** or it is lost.

NO SAVED COPY

```
int[] q = {3, 6, 9, 12};  
for (int i = q.length - 1; i > 0; i--) {  
    q[i] = q[i - 1];  
}  
q[0] = q[q.length - 1];
```

OUTPUT

9 3 6 9

The 12 was overwritten on the first pass.
It is gone.

Reverse: swap the ends, stop halfway

REVERSE IN PLACE

```
String[] c = {"red", "tan", "gold", "teal", "blue"};
for (int i = 0; i < c.length / 2; i++) {
    String temp = c[i];
    c[i] = c[c.length - 1 - i];
    c[c.length - 1 - i] = temp;
}
```

OUTPUT

blue teal gold tan red

i	Swaps
0	c[0] and c[4]
1	c[1] and c[3]
2	stop: 2 < 2 is false



A **swap** needs a temp variable. Stop at **length / 2**, or you undo your work.

Rotate the letters

PAUSE AND TRY

Rotate {"a", "b", "c", "d"} right by one. What does it hold after?

```
String[] w = {"a", "b", "c", "d"};
String last = w[w.length - 1];
for (int i = w.length - 1; i > 0; i--) {
    w[i] = w[i - 1];
}
w[0] = last;
```

w holds:

— — — —

Which element had to be saved first?

Words to know

Accumulator

A variable that starts before a loop and collects a sum or count inside it.

Best so far

The current minimum or maximum (or its index), started at element 0.

Flag

A boolean that records whether one element (at least one) or all elements have a property.

Consecutive pairs

Neighbors $a[i]$ and $a[i + 1]$; there are length - 1 of them.

Shift and rotate

Move each element one spot over; a rotation puts the one that falls off at the other end.

Swap

Exchange two elements using a temp variable. Reversing swaps mirror pairs.

Which algorithm is it?



With your partner, name the algorithm for each question and what it starts with before the loop.

1. What was the class average?
2. Did anyone score below 65?
3. Did two students get exactly the same score?

Next: The Array Workshop puzzle worksheet.