

Count Every Step

Informal run-time analysis



Who claps more?

PAUSE AND TRY

A row of students plays two clapping games.

THE RULES

Game A: clap once for each student.

Game B: clap once for each pair of students.

With 10 students, how many claps in each game? What about with 20 students?

By the end of class you can...

- 1 Find a statement's execution count by tracing.
- 2 Read how many times a loop runs straight from its header.
- 3 Count statements inside an if, and loops that never run.
- 4 Count the work in nested loops.
- 5 Compare two code segments by how much work they do, and how it grows.

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

PART 1

Counting executions

How many times does each statement run?

```
int rep = 1;
while (rep <= 3) {
    System.out.println("rep " + rep);
    rep++;
}
System.out.println("done");
```

OUTPUT

```
rep 1
rep 2
rep 3
done
```

Statement	Runs
int rep = 1;	1
rep <= 3	4
println("rep " ...)	3
rep++;	3
println("done");	1

A **statement execution count** is how many times a statement runs.

The parts of a for loop run different amounts

```
int sum = 0;
for (int lap = 0; lap < 5; lap++) {
    sum += lap;
}
System.out.println(sum);
```

OUTPUT

10

Part	Runs
int lap = 0	1
lap < 5	6
sum += lap;	5
lap++	5



The condition is always checked **one more time** than the body runs. The last check is the false one that ends the loop.

Doubling up

PAUSE AND TRY

How many times does the body run? How many condition checks?

```
int h = 1;
int passes = 0;
while (h < 50) {
    h *= 2;
    passes++;
}
System.out.println(passes + " " + h);
```

List the values of h at each check.

A large, bold, teal number '2' is centered within a white rounded square that has a slight 3D effect and a shadow.

PART 2

Reading the header

Count the values the variable takes

Header shape (count by 1)	Values	Passes
<code>for (int k = a; k < b; k++)</code>	$a, a + 1, \dots, b - 1$	$b - a$
<code>for (int k = a; k <= b; k++)</code>	$a, a + 1, \dots, b$	$b - a + 1$

```
int a = 0;
for (int k = 2; k < 9; k++) {
    a++;
}
int b = 0;
for (int k = 2; k <= 9; k++) {
    b++;
}
System.out.println(a + " " + b);
```

OUTPUT

7 8



From 2 to 9 **including both ends** is one more value than $9 - 2$.

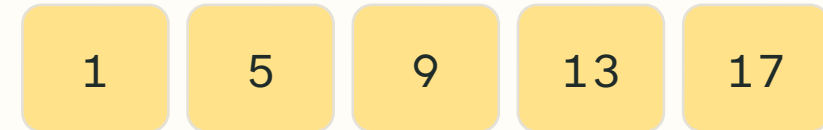
Steps other than 1: list the values

```
int up = 0;
for (int k = 1; k < 20; k += 4) {
    up++;
}
int down = 0;
for (int k = 15; k >= 0; k -= 5) {
    down++;
}
System.out.println(up + " " + down);
```

OUTPUT

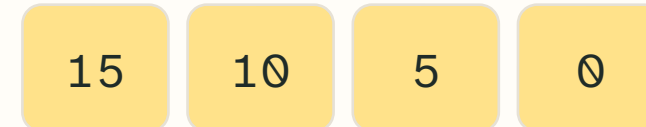
5 4

UP: VALUES OF K



21 fails

DOWN: VALUES OF K



-5 fails

One pass too many



Check both ends: the first value, and the **last value that still passes** the condition.

WANTED 5 ROWS

```
int rows = 0;
for (int r = 0; r <= 5; r++) {
    System.out.println("row " + r);
    rows++;
}
System.out.println(rows + " rows");
```

OUTPUT

```
row 0
row 1
row 2
row 3
row 4
row 5
6 rows
```

Fix: $r < 5$

A large, white, rounded square containing a bold, teal number '3'.

3

PART 3

Ifs and nested loops

An if inside a loop: two counts

```
int checks = 0;
int hits = 0;
for (int n = 1; n <= 24; n++) {
    checks++;
    if (n % 4 == 0) {
        hits++;
    }
}
System.out.println(checks + " " + hits);
```

OUTPUT

24 6

Statement	Runs
<code>n % 4 == 0</code>	24
<code>hits++;</code>	6

The **check** runs every pass.

The **body of the if** runs only when the check is true.

Nested loops multiply

```
int inner = 0;
int outer = 0;
for (int r = 0; r < 4; r++) {
    for (int c = 0; c < 6; c++) {
        inner++;
    }
    outer++;
}
System.out.println(inner + " " + outer);
```

OUTPUT

24 4

Statement	Runs
inner++;	4 × 6 = 24
outer++;	4

Same inner length every pass:
multiply.

Inner length changes with the outer
variable: **add** the counts.

Nested is not one after the other



Loops **one after another** add. Loops **one inside another** multiply.

IN A ROW: $4 + 6 = 10$

```
for (int r = 0; r < 4; r++) {  
    work++;  
}  
for (int c = 0; c < 6; c++) {  
    work++;  
}
```

NESTED: $4 \times 6 = 24$

```
for (int r = 0; r < 4; r++) {  
    for (int c = 0; c < 6; c++) {  
        work++;  
    }  
}
```

The inner loop gets shorter

PAUSE AND TRY

What prints? Make a table: i, and how many inner passes.

```
int count = 0;
for (int i = 0; i < 5; i++) {
    for (int j = i; j < 5; j++) {
        count++;
    }
}
System.out.println(count);
```

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

4

PART 4

Comparing run times

Same answer, less work

A: CHECK EVERY LETTER

```
boolean found = false;
for (int i = 0; i < w.length(); i++) {
    looks++;
    String c = w.substring(i, i + 1);
    if (c.equals("z")) {
        found = true;
    }
}
```

OUTPUT

true 5

B: STOP WHEN FOUND

```
boolean found = false;
int i = 0;
while (i < w.length() && !found) {
    looks++;
    String c = w.substring(i, i + 1);
    if (c.equals("z")) {
        found = true;
    }
    i++;
}
```

OUTPUT

true 3

Both run with `w = "pizza"` and print `found + " " + looks`.

How does the work grow?

Loop shape	n = 8	n = 16	n = 32	When n doubles...
Single loop, n passes	8	16	32	doubles
Nested, $n \times n$ passes	64	256	1024	× 4
Halving: size $\div 2$ until 1	3	4	5	+ 1



Compare code by its execution counts for the same input, **and** by how those counts grow as the input gets bigger.

Predict the bigger input

PAUSE AND TRY

Fill in the counts. Whose work grows faster?

Segment	Body runs	n = 30	n = 60
P: one loop	n		
Q: nested loop	n × n		

Words to know

Execution count

How many times a statement runs while the program runs.

Run time

How long a program takes to run.

Informal run-time analysis

Estimating and comparing run times by counting executions.

Counter variable

Starts at 0 and goes up by 1 each time a statement runs.

Pass

One run through a loop's body.

Off-by-one error

A loop that runs one time too many or too few.

Multiply or add?



Explain to your partner how to count the inner body of a nested loop. Use an example from today.

SENTENCE STARTER

If the inner loop runs the same number of times on every pass, I ____; if its length changes, I ____.

Next: the Count Every Step puzzle worksheet.