

Loop the Loop

while loops



One word, two different jobs

PAUSE AND TRY

A plate needs 3 scrubs. How many scrubs does each instruction give it?

INSTRUCTION A

If the plate is dirty, scrub it.

Scrubs: _____

INSTRUCTION B

While the plate is dirty, scrub it.

Scrubs: _____

By the end of class you can...

- 1 Recognize when a task needs a loop, because a step must repeat.
- 2 Write a while loop and name its condition and its body.
- 3 Trace a while loop with a table, including the check that ends it.
- 4 Explain why a loop can run zero times, and spot an infinite loop.
- 5 Find and fix off-by-one errors.

A large, white, rounded square containing the number '1' in a bold, black, sans-serif font, tilted slightly to the right.

1

PART 1

Check, run, repeat

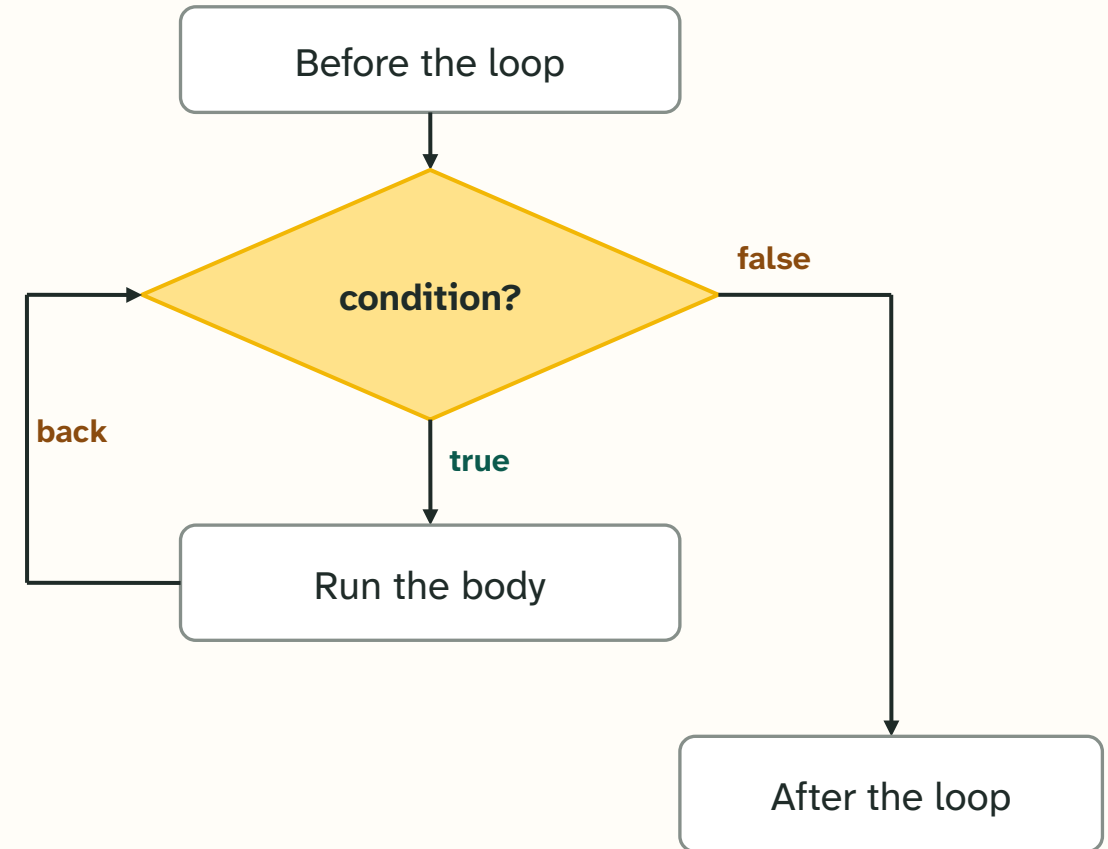
A while loop checks before every pass

THE SHAPE

```
while (condition) {  
    // body: the statements  
    // that repeat  
}  
// runs after the loop ends
```



The condition is checked **before every iteration**, including the first. When it is `true`, the body runs; when it is `false`, the loop ends.



WORKED EXAMPLE

Pour three cups

```
int cups = 0;
while (cups < 3) {
    System.out.println("Pour cup " + (cups + 1));
    cups++;
}
System.out.println("Done");
```

OUTPUT

```
Pour cup 1
Pour cup 2
Pour cup 3
Done
```

CONDITION `cups < 3`: checked 4 times, true 3 times, false once.

BODY The two statements inside the braces, run 3 times.

UPDATE `cups++` moves cups toward making the condition false.

WORKED EXAMPLE

Trace it with a table

```
int temp = 95;
int minutes = 0;
while (temp > 70) {
    temp -= 8;
    minutes++;
}
System.out.println(minutes + " min, "
    + temp + " degrees");
```

OUTPUT

4 min, 63 degrees

Check	temp	minutes	temp > 70
1	95	0	true
2	87	1	true
3	79	2	true
4	71	3	true
5	63	4	false

5 checks, 4 passes. The last check ends the loop.

if checks once. while keeps checking.



An `if` never goes back to its condition. Writing `if` where you meant `while` compiles fine and quietly runs the body only once.

WITH IF

```
int dirt = 3;  
if (dirt > 0) {  
    dirt--;  
}  
System.out.println(dirt);
```

OUTPUT

2

WITH WHILE

```
int dirt = 3;  
while (dirt > 0) {  
    dirt--;  
}  
System.out.println(dirt);
```

OUTPUT

0

if: a decision made once. **while:** a step that repeats until its condition is false.

Trace it

PAUSE AND TRY

Make a trace table. What prints? How many times does the body run?

```
int a = 1;
while (a < 40) {
    a = a * 3;
}
System.out.println(a);
```

Check	a	a < 40
1		
2		
3		
4		
5		

2

PART 2

Zero times, or forever

A loop can run zero times



If the condition is false at the **first** check, the body never runs. The loop does **zero iterations**.



The values a loop's variables need before it starts, so it works as intended, are its **initial conditions**.

```
int tickets = 0;
while (tickets > 0) {
    System.out.println("Ticket sold");
    tickets--;
}
System.out.println("Sold out");
```

OUTPUT

Sold out

Loops that never end



An **infinite loop** happens when the condition is true at **every** check. The program never moves past it.

NO UPDATE

```
int n = 5;
int sum = 0;
while (n > 0) {
    sum += n;
}
```

Nothing changes `n`.

SKIPS THE TARGET

```
int k = 1;
while (k != 10) {
    k += 2;
}
```

`k` is always odd, never 10.

UPDATE OUTSIDE

```
int t = 3;
int sum = 0;
while (t > 0) {
    sum += t;
}
t--;
```

`t--` never runs in the loop.

Does it end?

PAUSE AND TRY

For each loop, write "ends after _ passes" or "never ends".

```
int n = 12;
while (n != 0) {
    n -= 4;
}
```

A

```
int n = 10;
while (n != 0) {
    n -= 4;
}
```

B

```
int n = 5;
while (n > 0) {
    System.out.println(n);
}
```

C

```
int n = 5;
while (n > 10) {
    n--;
}
```

D

3

PART 3

Off by one

Check the first and the last



An **off-by-one error** runs a loop one time too many or one time too few. Check the start value, < versus <=, and the last value used.

VERSION 1

```
int k = 1;
while (k < 5) {
    System.out.print(k + " ");
    k++;
}
```

Version	Start	Condition	Prints	Passes
1	k = 1	k < 5	1 2 3 4	4
2	k = 1	k <= 5	1 2 3 4 5	5
3	k = 0	k < 5	0 1 2 3 4	5

Versions 2 and 3 change one thing each from version 1.

Update first, or print first?



Moving the update above or below the print changes **what the body sees**, so the first and last values printed shift.

PRINT, THEN UPDATE

```
int pts = 0;
while (pts < 30) {
    System.out.println(pts);
    pts += 10;
}
```

OUTPUT

```
0
10
20
```

UPDATE, THEN PRINT

```
int pts = 0;
while (pts < 30) {
    pts += 10;
    System.out.println(pts);
}
```

OUTPUT

```
10
20
30
```

Fix the off-by-one

PAUSE AND TRY

This should print 2 4 6 8 10. What does it print? Change one thing to fix it.

```
int even = 2;
while (even < 10) {
    System.out.print(even + " ");
    even += 2;
}
```

IT PRINTS

YOUR FIX

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

4

PART 4

Loops that stop on a goal

Repeat until the goal is reached



Use a loop when a step must **repeat**, especially when the number of repetitions depends on the data. The condition says **when to keep going**.

Task	Loop?
Warn once if the battery is low	No: if
Double until at least 500	Yes
Read scores until -1 appears	Yes

```
int followers = 30;
int weeks = 0;
while (followers < 500) {
    followers *= 2;
    weeks++;
}
System.out.println(weeks + " weeks, "
    + followers + " followers");
```

OUTPUT

5 weeks, 960 followers

Keep reading until the stop value

SUMS SCORES UNTIL -1

```
import java.util.Scanner;
Scanner console = new Scanner(System.in);
int total = 0;
int score = console.nextInt();
while (score != -1) {
    total += score;
    score = console.nextInt();
}
System.out.println("Total: " + total);
```

INPUT

12 7 30 -1

OUTPUT

Total: 49

-1 is a **sentinel**: a value that marks the end of the data.



Read the next value as the **last** statement of the body, so the condition checks it before the body uses it.

Words to know

while loop

Repeats its body as long as its condition is true, checking before every pass.

Iteration

Repeating code; also, one pass through a loop's body.

Loop condition

The boolean expression that decides whether the body runs again.

Infinite loop

A loop whose condition is true at every check, so it never ends.

Off-by-one error

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

Initial conditions

The starting values a loop needs to work as intended.

Will it stop?



Explain to your partner how you can tell, before running it, whether a while loop will ever stop.

SENTENCE STARTER

A while loop stops when _____, so the body has to _____.

Next: the Loop the Loop puzzle worksheet.