

Loops (for / while / do-while)

Stop copy-pasting. Let the loop do it. ■

■ What You Will Learn

- Why loops exist and when to use them
- for loop — when you know the count
- while loop — when you loop until a condition is false
- do-while loop — runs at least once no matter what
- break and continue — loop control

■ Intuition First

Without loops, to print 1 to 100 you'd write 100 `println()` calls. That's crazy. A loop says: **keep doing this thing until I say stop**. It's your program's way of doing repetitive work automatically.

■ Think Like This

A washing machine loop: fill water → wash → spin → drain → repeat. It doesn't ask you to press wash 20 times. You set it once and it goes. That's a loop.

■ Visual: The Loop Anatomy

```
for (int i = 1; i <= 5; i++) {
  ■■■■■■ ■■■■■■ ■■■
  ■ ■ ■
  START CONDITION INCREMENT
  i=1 i<=5? i++
  EACH CYCLE:
  i=1 → check 1<=5 ✓ → run body → i becomes 2
  i=2 → check 2<=5 ✓ → run body → i becomes 3
  i=3 → check 3<=5 ✓ → run body → i becomes 4
  i=4 → check 4<=5 ✓ → run body → i becomes 5
  i=5 → check 5<=5 ✓ → run body → i becomes 6
  i=6 → check 6<=5 ✗ → STOP
```

■ Code — All Three Loop Types

```
// ■■ FOR loop: when you know the count ■■■
for (int i = 1; i <= 5; i++) {
    System.out.println("Count: " + i);
}
// ■■ WHILE loop: keep going until done ■■■
int n = 10;
while (n > 0) {
    System.out.println(n);
    n -= 2;    // subtract 2 each time
}
// ■■ DO-WHILE: runs at least ONCE ■■■■■■■■
int x = 0;
do {
    System.out.println("Hello!"); // prints once even if x>0
    x++;
} while (x < 1);
// ■■ BREAK and CONTINUE ■■■■■■■■■■■■■■■■
for (int i = 1; i <= 10; i++) {
    if (i == 5) break;    // stops at 5
    if (i % 2 == 0) continue; // skip even numbers
    System.out.println(i); // prints 1, 3
}
```

■■ Time & Space Complexity

■ TIME COMPLEXITY

$O(n)$

■ SPACE COMPLEXITY

$O(1)$

If the loop runs n times, time grows linearly. No extra memory needed — just the loop counter.

■■ Common Mistakes

■■ Infinite Loop — the classic trap

If you forget to increment i ($i++$), the condition never becomes false and your program freezes. Always double-check your update step!

■■ Off-by-one errors

$i < n$ vs $i \leq n$ makes a big difference. $i < 5$ loops 0-4 (5 times). $i \leq 5$ loops 0-5 (6 times). Count carefully!

■ Practice Problems

#	Problem	Difficulty
1	Print 1 to 50 using a for loop	Easy

2	Print sum of all numbers from 1 to n	Easy
3	Print multiplication table of a given number	Easy
4	Print Fibonacci sequence up to n terms	Medium
5	Check if a number is prime using a loop	Medium

■ Quick Revision Box

- ✓ for = use when you know how many times to loop
- ✓ while = use when you loop until a condition fails
- ✓ do-while = always runs at least once
- ✓ break = exit the loop immediately
- ✓ continue = skip current iteration, move to next
- ✓ Infinite loop risk: always ensure condition eventually becomes false

■ Visual Thinking

■ Imagine It Like This

A loop is like a racetrack. The car (your code) keeps going around laps (iterations). The condition is the flag — when it drops, the race ends. break is the pit stop that exits early. continue makes the car skip a pothole and keep going. Animation: animate a dot going around a circular track.