

How your program talks to the world ■

■ What You Will Learn

- How to print messages to the screen
- How to read what the user types
- The difference between `print()` and `input()`
- How to combine text and numbers
- Basic formatting for output

■ Intuition First

Think of your program like a phone call. You say something (output), and the person on the other side replies (input). That's it. **Output = your program talking. Input = user talking back.**

■ ■ Real-World Analogy

An ATM is a perfect example. It shows your balance (output), asks for your PIN (input), then shows 'Transaction Complete' (output again). Every program you write will have this same rhythm.

■ Visual Explanation

YOUR PROGRAM

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52

■ ■

```
■ println("Enter your name:") ■■■ SCREEN
```

□ □

```
Scanner sc = new Scanner(System.in)
```

```
String name = sc.nextLine() ■■■ KEYBOARD
```

□ □

```
■ println("Hello, " + name) ■■■ SCREEN
```

□ □

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

FLOW: SCREEN ■■■ USER ■■■ KEYBOARD ■■■ YOUR PROGRAM

■ ■ How It Works (Step-by-Step)

■ Time & Space Complexity

■ TIME COMPLEXITY

$O(1)$

■ SPACE COMPLEXITY

$O(1)$

Reading one input and printing one output always takes constant time. No loops, no extra memory needed.

■ Common Mistakes

■ Forgetting to import Scanner

Java won't know what Scanner is! Always add: `import java.util.Scanner;` at the very top of your file.

■ Using `nextLine()` after `nextInt()`

After reading a number with `nextInt()`, there's a leftover newline. Add `sc.nextLine();` right after to clear it, or your next read will be empty.

■ Pro Tips

■ Pro Tip — `printf` for neat formatting

`System.out.printf("Name: %s, Age: %d\n", name, age);` — use this for clean, aligned output.

■ Intuition — `System.in` vs `System.out`

Think of `System.in` as an 'ear' (listening) and `System.out` as a 'mouth' (speaking). Your program has both!

■ Practice Problems

#	Problem	Difficulty
1	Print your name and favourite hobby on two lines	Easy
2	Read two numbers and print their sum	Easy
3	Ask for birth year, calculate and print age	Easy
4	Read a sentence and print it reversed word-by-word	Medium

5

Make a mini calculator: ask for op (+, -, *, /) and two numbers, print result

Medium

■ Quick Revision Box

- ✓ `System.out.println()` → prints to screen with newline
- ✓ `Scanner sc = new Scanner(System.in)` → reads from keyboard
- ✓ `sc.nextLine()` → reads a whole line of text
- ✓ `sc.nextInt()` / `sc.nextDouble()` → reads numbers
- ✓ Always close scanner: `sc.close()`
- ✓ Use `+` to join strings and variables in output

■ Visual Thinking

■ Imagine It Like This

Picture your program as a vending machine. The display screen is `System.out`. The keypad is `System.in`. When you press a button (input), the machine processes and shows a result (output). Animation idea: Show a vending machine where pressing buttons lights up corresponding code lines!