

Java Scanner: The Ultimate Beginner's Guide

How to make your Java programs listen to the user

Yahya Izadi

Pooyan Motamediquarto

2025-11-24

Table of contents

Introduction: Making Your Code “Listen”	1
The 3 Golden Steps	1
Step 1: Import the Tool	2
Step 2: Create the Scanner Object	2
Step 3: Read the Data	2
Cheat Sheet: Which Command to Use?	2
Full Code Example	3
Summary	4

Introduction: Making Your Code “Listen”

By default, the Java code you write is “**deaf**.” It executes commands, does calculations, and prints results, but it cannot hear what the user wants to say.

The **Scanner** class is like a hearing aid or a translator. It allows your program to read input from the keyboard (Standard Input) and store it in variables.

The 3 Golden Steps

To use the Scanner, you must always follow these three steps in order. Memorize this pattern!

Step 1: Import the Tool

Java keeps the Scanner in a utility package to keep things organized. You must import it at the very top of your file.

```
import java.util.Scanner;
```

Step 2: Create the Scanner Object

Inside your main method, you need to create an instance of the Scanner. We usually call it `input` or `scanner`.

```
// System.in represents the keyboard  
Scanner input = new Scanner(System.in);
```

Step 3: Read the Data

Now, you command the scanner to wait for input and store it in a variable.

```
int age = input.nextInt();
```

Cheat Sheet: Which Command to Use?

Depending on the **Data Type** you want to read, you must use a specific command (method).

Data Type	Method	Example Use Case
int	input.nextInt()	Age, Year, Quantity
double	input.nextDouble()	Price, GPA, Salary
String	input.next()	A single word (e.g., First Name)
String	input.nextLine()	A full sentence (e.g., Full Name)
boolean	input.nextBoolean()	True/False inputs

Full Code Example

Here is a complete, runnable program. It asks for user details and prints them back.

```
import java.util.Scanner; // Step 1

public class Main {
    public static void main(String[] args) {
        // Step 2: Create the Scanner
        Scanner input = new Scanner(System.in);

        // Step 3: Ask and Read
        System.out.print("Enter your age: ");
        int age = input.nextInt();

        System.out.print("Enter your GPA: ");
        double gpa = input.nextDouble();

        System.out.println("--- Result ---");
        System.out.println("Age: " + age);
        System.out.println("GPA: " + gpa);
    }
}
```



The Common “New Line” Trap

There is a famous bug that confuses every beginner.

If you use `nextInt()` (to read a number) and **immediately** afterwards use `nextLine()` (to read text), the Scanner will **skip** the text input!

Why? When you type a number and hit **Enter**, `nextInt()` takes the number but leaves the “Enter” key press (newline character) in the memory. The next `nextLine()` reads that leftover “Enter” and thinks you finished typing.

The Fix: Add an extra `nextLine()` just to consume the leftover “Enter”.

```
int id = scanner.nextInt();

// Fix: Consume the leftover newline
scanner.nextLine();

String name = scanner.nextLine(); // Now this works!
```

Summary

1. **Import** `java.util.Scanner`.
2. **Create** new `Scanner(System.in)`.
3. **Use** `nextInt()`, `nextDouble()`, or `nextLine()` based on the variable type.
4. **Watch out** for the newline trap when switching from numbers to text!

Happy Coding!