The `Scanner` **class** in Java is a **built-in utility class** that allows your program to **read input from various sources**, most commonly from the **keyboard (user input)**. It's part of the `java.util` **package**, so you need to import it before using it:

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

Key Points About `Scanner`:

1. **Purpose:** The Scanner class reads data typed by the user, such as text, numbers, or boolean values, and stores it in variables for your program to use.
2. **Creating a Scanner Object:** Before reading input, you create a Scanner object and connect it to a source. For keyboard input:

   ```
   Scanner input = new Scanner(System.in);
   ```

   - `input` → the name of the Scanner object
   - `System.in` → tells Scanner to listen to the standard input (keyboard)
3. **Common Methods:**

Method	Reads	Example Input
`nextLine()`	Whole line of text (`String`)	Hello World
`next()`	Single word (`String`)	Hello
`nextInt()`	Integer (`int`)	25
`nextDouble()`	Decimal number (`double`)	9.5
`nextBoolean()`	True/False (`boolean`)	true

To make your program interactive, we use a built-in Java tool called the **Scanner**. This allows the program to pause and wait for the user to type something on their keyboard.

The Interactive Greeting Program – Java Scanner Example

This program shows how to make your Java program **interactive**. Instead of just printing fixed messages, it can **ask the user for input** and respond dynamically.

Save the code in a file named:

```
InputExample.java
```

```java
import java.util.Scanner; // Step 1: Import the Scanner class

public class InputExample {
    public static void main(String[] args) {
        // Step 2: Create a Scanner object to read input
        Scanner reader = new Scanner(System.in);

        System.out.print("Enter your name: ");

        // Step 3: Read the input typed by the user
        String name = reader.nextLine();

        System.out.println("Nice to meet you, " + name + "!");

        // Step 4: Close the Scanner to free resources
        reader.close();
    }
}
```

How This Program Works

Think of **Scanner** as a "telephone line" connecting the keyboard to your code. The program waits for the user to type and then responds.

1 Import the Scanner Class

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

- Java doesn't automatically load all tools to keep programs efficient.
- This line tells Java:

> "I want to use the Scanner class from the utility library."

---

2 Create the Scanner Object

```java
Scanner reader = new Scanner(System.in);
```

- `reader` → the name of your Scanner object
- `System.in` → tells Scanner to listen to the keyboard
- Now your program can "hear" what the user types.

---

3 Read Input From the User

```
String name = reader.nextLine();
```

- `nextLine()` waits for the user to type a full line of text and press **Enter**.
- The text is stored in the variable `name`.
- You can now use this variable anywhere in your program.

---

4 Display Output

```
System.out.println("Nice to meet you, " + name + "!");
```

- Combines the fixed text with the user input using **concatenation (+)**.
- Prints a personalized greeting on the screen.

---

5 Close the Scanner

```
reader.close();
```

- Always close the Scanner when you are done.
- Frees system resources and prevents memory issues.

Quick Tip for Beginners

⚠ **Common Issue:** Mixing `nextInt()` and `nextLine()`

If you use `nextInt()` (or `nextDouble()`) and then immediately call `nextLine()`, Java may **skip the input**.
☑ Fix: Add an extra `nextLine()` after `nextInt()` to clear the leftover newline.

```
int age = reader.nextInt();
reader.nextLine(); // clears the input buffer
String name = reader.nextLine();
```

💡 For absolute beginners, it's easiest to use `nextLine()` for all input at first.

Example Run

```
Enter your name: Ali
Nice to meet you, Ali!
```

Common Beginner Mistakes

❌ Forgetting import statement ❌ Wrong file name ❌ Not closing scanner ❌ Using wrong input method

---

Tasks

Task #1

Modify the program to ask:

- Name
- Age

Then print:

```
Hello Ali, you are 15 years old!
```

Task #2

Create a program that asks:

1. Name
2. Favorite subject
3. Favorite number

Then prints a personalized message.

The Simple Calculator

Create a file named `Calculator.java` and paste this in:

```java
import java.util.Scanner;

public class Calculator {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        System.out.println("--- Java Addition Tool ---");

        // 1. Get the first number
        System.out.print("Enter first number: ");
        double num1 = input.nextDouble();

        // 2. Get the second number
        System.out.print("Enter second number: ");
        double num2 = input.nextDouble();
```

```
        // 3. Perform the calculation
        double sum = num1 + num2;

        // 4. Output the result
        System.out.println("The total is: " + sum);

        input.close();
    }
}
```

Key Concepts Used

1. The `double` Type

We used `double` instead of `int`. Why? Because `int` only handles whole numbers (1, 2, 3). If a user types **5.5**, an `int` would crash the program. A `double` can handle both.

2. Basic Arithmetic

Java uses standard symbols for math:

- Addition: `+`
- Subtraction: `-`
- Multiplication: `*`
- Division: `/`

3. Order of Operations

Just like in math class, Java follows **PEMDAS** (Parentheses, Exponents, Multiplication/Division, Addition/Subtraction). If you want to do a complex calculation, use parentheses: `double result = (num1 + num2) * 10;`

Common Beginner Mistake: The "Input Bug"

If you try to type a **letter** when the program asks for a `nextDouble()`, the program will crash with an `InputMismatchException`. Java is very strict—if it expects a number, you *must* give it a number!

Related Topics

- Standard Output
- Data Types and Variables