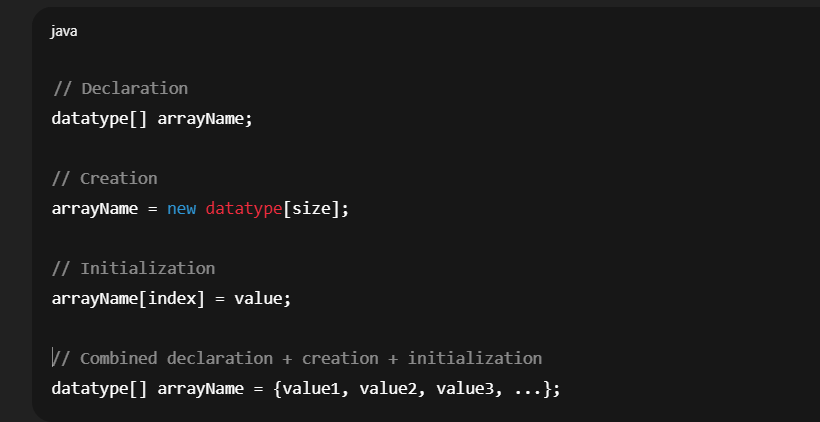
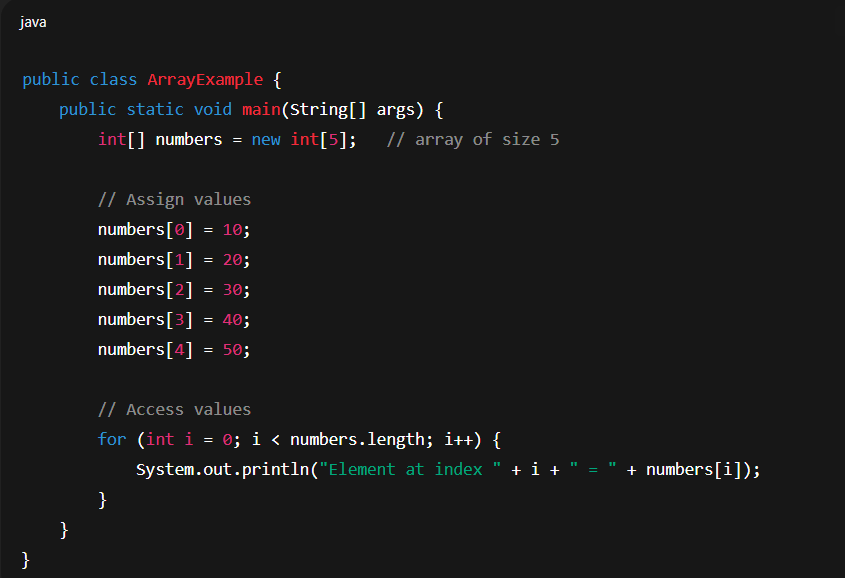
## ****Array****

## ****What is an Array in Java?****

* An **array** in Java is a **collection of elements of the same type** stored in a **contiguous memory location**.
* It is a **fixed-size data structure** → once created, its size cannot be changed.
* Each element is accessed using an **index** (starts from 0).
* **Why use Arrays?**
* To store multiple values in a single variable instead of declaring separate variables.
* Provides fast access using **indexing**.
* Useful for loops and bulk operations.
* **Syntax of Array**



* Example



* Output



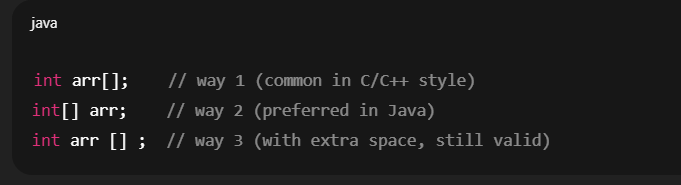
## ****Types of Arrays in Java****

1. **Single-Dimensional Array (1D Array)**

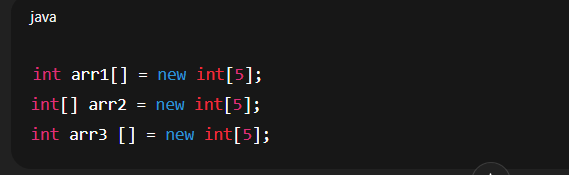
* Linear collection of elements.

## ****1D Array Declarations****

* There are **3 valid ways** to declare a one-dimensional array:



* Example

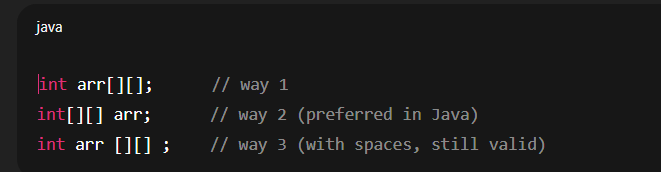


**2.Multi-Dimensional Array**

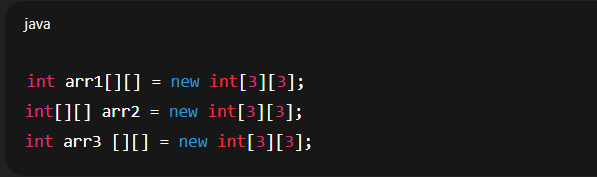
* An array of arrays. Most commonly **2D array**.

## ****2D Array Declarations****

* Similarly, a two-dimensional array also has **3 valid ways**:

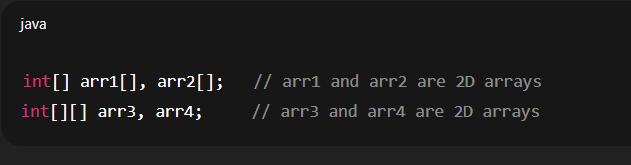


* Example



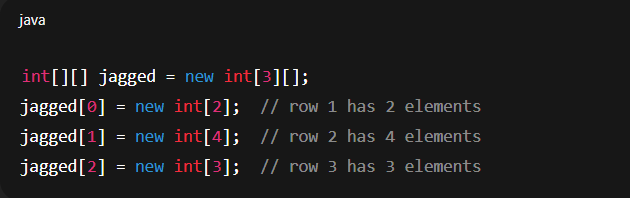
## Special Case: Mixed Style (also valid)

* Java also allows **placing brackets after the variable name**:



**3.Jagged Array**

* A 2D array where rows have **different lengths**.



## ****Important Properties****

* Array size is fixed once created.
* arr.length gives the size (number of elements).
* Default values:
* Numeric → 0
* Boolean → false
* Objects → null

## Valid and Invalid Array Declarations:

* int [][] arr; ✅ → Valid

2D array of integers.

* int arr[][]; ✅ → Valid

2D array of integers.

* int []arr[]; ✅ → Valid

2D array (1D array of 1D arrays).

* int [] arr []; ✅ → Valid
* int []a, b; ✅ **valid**

a → 1D array

b → **just an int**, NOT an array.

* int []a[], b; ✅ **valid**

a → 2D array

b → just an int.

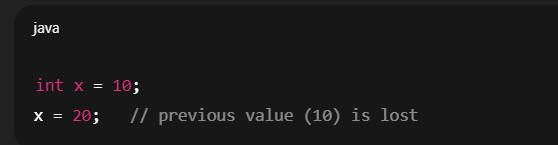
* int [][] a, b; ✅ **valid**

Both a and b are 2D arrays.

* int [][]a[], b;✅ **valid**
* **(Loopholes / Limitations of variable,Array,collection framework)**

## ****1.Variables****

* A variable can hold **only one value at a time**.

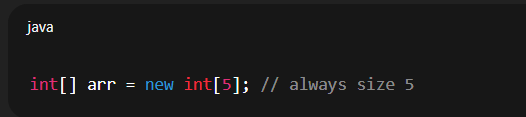


· To store multiple values → need **arrays or collections**.

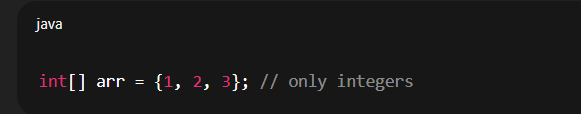
· Fixed type → can’t store different data types in same variable.

## ****2.Arrays****

* **Fixed Size**: Once declared, size cannot be changed.



* **Homogeneous elements only**: Can store only **same type** of data.



· No built-in **methods** for common operations (like add, remove, search).

· Wastage of memory if array not fully used.

· Insertion/deletion is costly (shifting required).

## ****3.Collection Framework****

✅ Collections solve many array problems but have their own issues:

* **Overhead**: Collections (like ArrayList, HashMap) use more memory and processing compared to arrays.
* **Thread-Safety**: Most collections (e.g., ArrayList, HashMap) are **not synchronized** by default. Need Collections.synchronizedList() or concurrent collections.
* **Type Safety (before Java 5)**: Collections stored Object, requiring explicit casting. (Solved by Generics in Java 5).
* **Slower than arrays** in some cases (because of dynamic resizing, boxing/unboxing with wrappers).
* Choosing the right collection (List, Set, Map) can be confusing.