

ARRAYS IN JAVA — COMPLETE NOTES

1. Definition

An Array in Java is a collection of elements of the same data type, stored in continuous (contiguous) memory locations.

Arrays are used to **store multiple values** in a single variable instead of declaring separate variables for each value.

Example

```
int[] marks = {85, 90, 75, 80, 95};
```

Here:

- int → data type
- marks → array name
- {85, 90, 75, 80, 95} → elements

2. Need of Arrays

1. To store multiple values of the same data type.
2. To avoid creating multiple variables.
3. To easily access and manipulate data using index.
4. To perform batch operations like sorting, searching, and looping.

3. Features of Arrays

Feature	Description
Fixed Size	Once declared, the size cannot be changed.
Homogeneous Elements	All elements must be of the same data type.
Indexed Access	Index starts from 0.
Sequential Memory	Elements are stored in continuous memory.
Stored in Heap	Arrays are objects in Java and created in heap memory.
Length Property	<code>.length</code> gives total number of elements.

4. Syntax and Structure

Declaration

```
dataType[] arrayName;
```

Creation

```
arrayName = new dataType[size];
```

Initialization

```
arrayName[index] = value;
```

Example (Combined)

```
int[] marks = new int[5];
marks[0] = 10;
marks[1] = 20;
marks[2] = 30;
marks[3] = 40;
marks[4] = 50;
```

5. Accessing Array Elements

```
System.out.println(marks[0]); // prints first element
System.out.println(marks[4]); // prints last element
Index starts at 0 and ends at length - 1.
```

6. Traversing an Array

Using For Loop

```
for (int i = 0; i < marks.length; i++) {
    System.out.println(marks[i]);
}
```

Using Enhanced For Loop

```
for (int mark : marks) {
    System.out.println(mark);
}
```

7. Types of Arrays

One-Dimensional Array

Stores elements in a single row.

```
int[] arr = {10, 20, 30, 40, 50};
```

Two-Dimensional Array (Matrix)

Stores data in rows and columns.

```
int[][] matrix = {
    {1, 2, 3},
    {4, 5, 6},
    {7, 8, 9}
};
```

Access elements:

```
System.out.println(matrix[1][2]); // prints 6
```

8. Memory Allocation of Arrays

Memory Type	Stored Item
Stack Memory	Reference variable (array name)

Heap Memory	Actual array elements
------------------------	-----------------------

Example:

```
int[] arr = new int[5];
• arr → stored in Stack
• elements → stored in Heap
```

9. Default Values of Array Elements

Data Type	Default Value
int, byte, short, long	0
float, double	0.0
char	\u0000 (empty char)
boolean	FALSE
String/Object	null

10. Important Array Properties & Methods (java.util.Arrays)

Method	Description	Example
<code>Arrays.toString(array)</code>	Converts array to readable	<code>Arrays.toString(arr)</code>
<code>Arrays.sort(array)</code>	Sorts array in ascending	<code>Arrays.sort(marks)</code>
<code>Arrays.equals(a, b)</code>	Compares two arrays	<code>Arrays.equals(exp,</code>
<code>Arrays.copyOf(array,</code>	Copies first n elements	<code>Arrays.copyOf(arr,</code>
<code>Arrays.copyOfRange(a, from, to)</code>	Copies range of elements	<code>Arrays.copyOfRange(a rr, 1, 4)</code>
<code>Arrays.fill(array,</code>	Fills array with one value	<code>Arrays.fill(arr,</code>
<code>Arrays.binarySearch(a rray, key)</code>	Searches for a value (array must be sorted)	<code>Arrays.binarySearch(arr, 30)</code>
<code>Arrays.deepToString(a</code>	Prints multi-dimensional	<code>Arrays.deepToString(</code>

11. Example Programs

1. Find Sum and Average

```
int[] arr = {10, 20, 30, 40, 50};
int sum = 0;
for (int n : arr) sum += n;
System.out.println("Sum = " + sum);
System.out.println("Average = " + (sum / arr.length));
```

2. Find Maximum and Minimum

```
int[] arr = {50, 10, 70, 30, 90};
```

```
int max = arr[0], min = arr[0];  
  
for (int n : arr) {  
    if (n > max) max = n;  
    if (n < min) min = n;  
}  
  
System.out.println("Max: " + max);  
System.out.println("Min: " + min);
```

3. Search an Element

```
int[] arr = {10, 20, 30, 40};  
int search = 30;  
boolean found = false;  
  
for (int n : arr) {  
    if (n == search) {  
        found = true;  
        break;  
    }  
}  
System.out.println(found ? "Element Found" : "Not Found");
```

4. Reverse an Array

```
int[] arr = {10, 20, 30, 40, 50};  
for (int i = arr.length - 1; i >= 0; i--) {  
    System.out.print(arr[i] + " ");  
}
```

5. Compare Two Arrays

```
import java.util.Arrays;  
String[] expected = {"Pass", "Fail"};  
String[] actual = {"Pass", "Fail"};  
System.out.println(Arrays.equals(expected, actual));
```

6. Copy One Array into Another

```
int[] original = {1, 2, 3, 4, 5};  
int[] copy = Arrays.copyOf(original, original.length);  
System.out.println(Arrays.toString(copy));
```

8. Object Array (Heterogeneous Data)

```
Object[] obj = {10, "Java", true, 99.9};
```

```
for (Object o : obj) System.out.println(o);
```

9. Two-Dimensional Array Traversal

```
int[][] matrix = {  
    {1, 2, 3},  
    {4, 5, 6},  
    {7, 8, 9}  
};  
  
for (int i = 0; i < matrix.length; i++) {  
    for (int j = 0; j < matrix[i].length; j++) {  
        System.out.print(matrix[i][j] + " ");  
    }  
    System.out.println();  
}
```

10. Find Maximum Element in 2D Array

```
int[][] matrix = {  
    {10, 25, 30},  
    {5, 45, 20},  
    {60, 15, 35}  
};  
  
int max = matrix[0][0];  
for (int[] row : matrix) {  
    for (int val : row) {  
        if (val > max) max = val;  
    }  
}  
System.out.println("Maximum: " + max);
```

12. Difference Between 1D and 2D Array

Feature	1D Array	2D Array
Structure	Single row (linear)	Rows & columns (table)
Declaration	int[] a = new int[5];	int[][] a = new int[3][3];
Access	a[i]	a[i][j]
Example	Marks of 5 students	Marksheet of 5 students in 3 subjects

13. Advantages of Arrays

1. Easy to manage large data sets.
2. Random access to elements using index.
3. Used to store multiple values efficiently.
4. Simplifies looping and iteration.

14. Limitations of Arrays

1. **Fixed size** – cannot increase or decrease dynamically.
2. Same data type only – cannot mix types.
3. **Memory waste** – if declared size is larger than needed.
4. No built-in functions for add/remove (use Collections for that).

15. Real-Time Usage (Automation Context)

Scenario	Use of Array
Store browser names	<code>String[] browsers = {"Chrome", "Edge", "Firefox"};</code>
Validate expected vs actual statuses	<code>Arrays.equals(expected, actual)</code>
Store test IDs	<code>int[] testIDs = {101, 102, 103};</code>
Handle mixed data	<code>Object[] obj = {id, name, status};</code>
Compare UI vs API values	Arrays for expected and actual responses