

1. Single-Dimensional (1D) Array

A linear array with elements stored in a single row.

Declaration & Initialization

```
class OneDArray {
    public static void main(String[] args) {
        // Method 1: Declaration + memory allocation
        int[] arr1 = new int[5]; // default values = 0

        // Assign values
        arr1[0] = 10;
        arr1[1] = 20;
        arr1[2] = 30;
        arr1[3] = 40;
        arr1[4] = 50;

        // Method 2: Declaration + initialization
        int[] arr2 = {1, 2, 3, 4, 5};

        // Print array
        System.out.println("Array 1:");
        for (int i = 0; i < arr1.length; i++) {
            System.out.print(arr1[i] + " ");
        }

        System.out.println("\nArray 2 (Enhanced for loop):");
        for (int val : arr2) {
            System.out.print(val + " ");
        }
    }
}
```

2. Two-Dimensional (2D) Array

Represents a **matrix** (rows and columns).

Declaration & Initialization

```

class TwoDArray {
    public static void main(String[] args) {
        // Method 1: Declaration + memory allocation
        int[][] arr1 = new int[2][3]; // 2 rows, 3 columns

        // Assign values
        arr1[0][0] = 1;
        arr1[0][1] = 2;
        arr1[0][2] = 3;
        arr1[1][0] = 4;
        arr1[1][1] = 5;
        arr1[1][2] = 6;

        // Method 2: Direct initialization
        int[][] arr2 = {
            {10, 20, 30},
            {40, 50, 60}
        };

        // Print 2D array
        System.out.println("Array 1:");
        for (int i = 0; i < arr1.length; i++) {
            for (int j = 0; j < arr1[i].length; j++) {
                System.out.print(arr1[i][j] + " ");
            }
            System.out.println();
        }

        System.out.println("Array 2:");
        for (int[] row : arr2) {
            for (int val : row) {
                System.out.print(val + " ");
            }
            System.out.println();
        }
    }
}

```

3. Three-Dimensional (3D) Array

Represents data in a **cube form** (layers → rows → columns).

Declaration & Initialization

```
class ThreeDArray {
    public static void main(String[] args) {
        // Method 1: Declaration + memory allocation
        int[][][] arr1 = new int[2][2][3]; // 2 layers, 2 rows, 3
columns

        // Assign values
        int num = 1;
        for (int i = 0; i < 2; i++) {           // layers
            for (int j = 0; j < 2; j++) {       // rows
                for (int k = 0; k < 3; k++) {   // columns
                    arr1[i][j][k] = num++;
                }
            }
        }
    }
}
```

4. Jagged Array (Irregular 2D Array)

Different rows can have different column sizes.

```
class JaggedArray {
    public static void main(String[] args) {
        // Declare 2D array with variable column size
        int[][] jagged = new int[3][];
        jagged[0] = new int[2]; // row 1 has 2 columns
        jagged[1] = new int[3]; // row 2 has 3 columns
        jagged[2] = new int[4]; // row 3 has 4 columns

        // Assign values
        int val = 1;
        for (int i = 0; i < jagged.length; i++) {
            for (int j = 0; j < jagged[i].length; j++) {
                jagged[i][j] = val++;
            }
        }
    }
}
```

```
// Print jagged array
for (int i = 0; i < jagged.length; i++) {
    for (int j = 0; j < jagged[i].length; j++) {
        System.out.print(jagged[i][j] + " ");
    }
    System.out.println();
}
}
```

✅ Summary of Array Types in Java

1. **1D Array** → Linear list.
 2. **2D Array** → Matrix (rows × cols).
 3. **3D Array** → Cube (layers × rows × cols).
 4. **Jagged Array** → Different column sizes per row.
-

👉 Do you also want me to make you a **visual diagram (with memory layout of 1D, 2D, 3D arrays in Java)** so it's easier for placements and interviews?

You said:

array functions

ChatGPT said:

Perfect 👍 You're asking about **array functions in Java**.

In Java, arrays are **objects**, but they don't have built-in "methods" like in Python. Instead, we use **utility methods** from the `java.util.Arrays` class and the array's built-in **length property**.

Here's the full list of commonly used **array functions** with examples 👉

♦ 1. **length** (property, not method)

Gives the size of the array.

```
class ArrayLength {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40};
        System.out.println("Array length: " + arr.length);
    }
}
```

◆ 2. Arrays.toString()

Converts array to a string for easy printing.

```
import java.util.Arrays;

class ToStringDemo {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        System.out.println(Arrays.toString(arr)); // [1, 2, 3, 4,
5]
    }
}
```

◆ 3. Arrays.sort()

Sorts array in ascending order.

```
import java.util.Arrays;

class SortDemo {
    public static void main(String[] args) {
        int[] arr = {40, 10, 30, 20};
        Arrays.sort(arr);
        System.out.println(Arrays.toString(arr)); // [10, 20, 30,
40]
    }
}
```

◆ 4. **Arrays.fill()**

Fills all elements with a given value.

```
import java.util.Arrays;

class FillDemo {
    public static void main(String[] args) {
        int[] arr = new int[5];
        Arrays.fill(arr, 7);
        System.out.println(Arrays.toString(arr)); // [7, 7, 7, 7, 7]
    }
}
```

◆ 5. **Arrays.equals()**

Checks if two arrays are equal (element by element).

```
import java.util.Arrays;

class EqualsDemo {
    public static void main(String[] args) {
        int[] arr1 = {1, 2, 3};
        int[] arr2 = {1, 2, 3};
        System.out.println(Arrays.equals(arr1, arr2)); // true
    }
}
```

◆ 6. **Arrays.copyOf()** and **Arrays.copyOfRange()**

Copies elements into a new array.

```
import java.util.Arrays;

class CopyDemo {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40, 50};
```

```

        int[] copy1 = Arrays.copyOf(arr, 3);           // first 3
elements
        int[] copy2 = Arrays.copyOfRange(arr, 1, 4); // index 1 to 3

        System.out.println(Arrays.toString(copy1)); // [10, 20, 30]
        System.out.println(Arrays.toString(copy2)); // [20, 30, 40]
    }
}

```

◆ 7. Arrays.binarySearch()

Finds the index of an element in a **sorted array**.

```

import java.util.Arrays;

class SearchDemo {
    public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40, 50};
        int index = Arrays.binarySearch(arr, 30);
        System.out.println("Index of 30: " + index); // 2
    }
}

```

✓ Summary Table of Array Functions

| Function | Use Case |
|------------------------------------|-----------------------|
| <code>length</code> | Size of array |
| <code>Arrays.toString()</code> | Print 1D array |
| <code>Arrays.deepToString()</code> | Print 2D/3D array |
| <code>Arrays.sort()</code> | Sort array |
| <code>Arrays.fill()</code> | Fill array with value |
| <code>Arrays.equals()</code> | Compare arrays |
| <code>Arrays.copyOf()</code> | Copy whole/part array |

`Arrays.copyOfRange()` Copy specific range

`Arrays.binarySearch()` Search in sorted array

`Arrays.stream()` Advanced operations (sum, avg, filter)

`System.arraycopy()` Copy array elements

👉 Do you want me to also give you a **practice sheet of array coding questions (from beginner to advanced)** for placements?