**Java Arrays Overview**

**1. One-Dimensional Arrays**

A one-dimensional array is a sequence of elements of the same data type, organized in a single row or column.

**Declaration and Initialization**:

java

CopyEdit

// Declare and initialize an array

int[] numbers = new int[5]; // Array with 5 elements

// Assign values

numbers[0] = 10;

numbers[1] = 20;

numbers[2] = 30;

numbers[3] = 40;

numbers[4] = 50;

// Alternatively, declare and initialize in one step

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

**Accessing Elements**:

java

CopyEdit

System.out.println(numbers[0]); // Prints 10

System.out.println(numbers[4]); // Prints 50

**2. Multi-Dimensional Arrays**

Multi-dimensional arrays allow you to create arrays of arrays, with 2D arrays (like matrices) being the most common.

**Declaration and Initialization of 2D Arrays**:

java

CopyEdit

// Declare and initialize a 2D array

int[][] matrix = new int[3][3]; // 3x3 matrix

matrix[0][0] = 1; matrix[0][1] = 2; matrix[0][2] = 3;

matrix[1][0] = 4; matrix[1][1] = 5; matrix[1][2] = 6;

matrix[2][0] = 7; matrix[2][1] = 8; matrix[2][2] = 9;

// Initialize directly with values

int[][] matrix = {

{1, 2, 3},

{4, 5, 6},

{7, 8, 9}

};

**Accessing Elements in 2D Arrays**:

java

CopyEdit

System.out.println(matrix[0][1]); // Prints 2

System.out.println(matrix[2][2]); // Prints 9

For arrays with more dimensions, you can define them similarly:

java

CopyEdit

int[][][] cube = new int[3][3][3]; // A 3x3x3 3D array

**3. Operations on Arrays**

**Traversing (Looping through an Array)**:

java

CopyEdit

// Using a traditional for loop

for (int i = 0; i < numbers.length; i++) {

System.out.println(numbers[i]); // Prints each element

}

// Using an enhanced for loop

for (int num : numbers) {

System.out.println(num);

}

**Searching an Element**:

java

CopyEdit

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

int target = 30;

boolean found = false;

for (int num : numbers) {

if (num == target) {

found = true;

break;

}

}

System.out.println(found ? "Element found!" : "Element not found.");

**Sorting an Array**:

java

CopyEdit

import java.util.Arrays;

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

Arrays.sort(numbers); // Sort in ascending order

for (int num : numbers) {

System.out.println(num); // Prints 10, 20, 30, 40, 50

}

**Finding Maximum or Minimum**:

java

CopyEdit

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

int max = numbers[0];

for (int num : numbers) {

if (num > max) {

max = num;

}

}

System.out.println("Maximum value: " + max); // Prints 50

**4. Arrays Utility Class (java.util.Arrays)**

The Arrays class provides helpful methods for array manipulation.

**Common Methods**:

1. **Arrays.sort()**: Sorts an array in ascending order.

java

CopyEdit

int[] numbers = {5, 1, 3, 4, 2};

Arrays.sort(numbers);

1. **Arrays.toString()**: Converts an array to a string.

java

CopyEdit

System.out.println(Arrays.toString(numbers)); // Prints [10, 20, 30, 40, 50]

1. **Arrays.copyOf()**: Copies and optionally resizes an array.

java

CopyEdit

int[] newNumbers = Arrays.copyOf(numbers, 7); // Extends the array

1. **Arrays.equals()**: Checks if two arrays are equal.

java

CopyEdit

int[] arr1 = {1, 2, 3};

int[] arr2 = {1, 2, 3};

System.out.println(Arrays.equals(arr1, arr2)); // true

1. **Arrays.fill()**: Fills an array with a specific value.

java

CopyEdit

Arrays.fill(numbers, 42); // Fills all elements with 42

1. **Arrays.binarySearch()**: Searches for a value in a sorted array.

java

CopyEdit

int index = Arrays.binarySearch(numbers, 30); // Returns index of 30

**Summary**

* **One-Dimensional Arrays**: Simple arrays for storing elements of the same type.
* **Multi-Dimensional Arrays**: Arrays of arrays (e.g., 2D arrays for matrices).
* **Operations**: Traverse, search, sort, and perform calculations on arrays.
* **Arrays Utility Class**: Offers built-in methods for efficient array handling.