ARRAYS IN JAVA

An array in Java is a collection of variables of the same type stored in a contiguous memory location. Arrays allow you to store multiple values in a single variable rather than declaring separate variables for each value.

Keys Points About Arrays:

- Fixed Size : Once an array is created, its size cannot be changed.
- Same Type : All elements in an array must be of the same type (eg., all integers, all strings).
- Indexing : Arrays are zero-indexed, meaning the first element has an index of 0.

a) Declaring and initializing Arrays

You can declare and initialize an array in multiple ways.

```
int [] arr ;  // declaration of an array of integers
```

b) Initializing an array :

1. With size :

```
arr = new int [5] ;  // Array of 5 integers , all
                        initialized to 0
```

2. With values :

```
int [] arr = {1, 2, 3, 4, 5} ; // Array with
                                  predefined values
```

Accessing Array elements :

Array elements are accessed using an index.
Remember , the index starts at 0.

```
int [] arr = { 10 , 20 , 30 , 40 , 50 };
system.out.println (arr [0]); // output : 10
system.out.println ( arr [3]); // output : 40
```

@ Curious - programmer

Array length:

The length of an array (i.e., the number of elements) can be accessed using the length property.

```
int[]arr = {1, 2, 3, 4, 5};
system.out.println ("Array length : " + arr.length);
                                        // Output : 5
```

Looping through Arrays:

You can loop through arrays using a for loop or an enhanced for-each loop.

a) Regular for loop:

```
int[] arr = {10, 20, 30, 40, 50};
for (int i = 0; i< arr.length; i++) {
   system.out.println (arr[i]);
}                    // output : 10, 20  30  40  50
```

@ Curious-. programmer

b) Enhanced for-each loop:

```
int[] arr = {10, 20, 30, 40, 50};
for (int num : arr) {
     system. out. println (num); // output : 10 20 30 40 50
 }
```

1. Single - Dimensional Arrays :

A single - dimensional array (1D array) is a simple list of elements.

You can access each element by its index.

Declaring and Initializing a Single - Dimensional array :

```
int[] arr = new int[5];   // Declaring an array
arr[0] = 10;              // Assigning values to each element
arr[1] = 20;
arr[2] = 30;
arr[3] = 40;
arr[4] = 50;
```

or, you can initialize it with values directly:

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

2. Multi - Dimensional Arrays:

A multi - dimensional array is an array of arrays. The most common multi - dimensional array is a 2D array (think of a table or matrix).

Declaring and Initializing a 2D Arrays:

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

Accessing Elements in a 2D Array:

```
system.out.println (matrix [0][0]);
                        // output : 1  (first row, first column)
system.out.println (matrix [2][2]);
                        // output : 9  (third row, third column)
```

@ Curious_.programmer

Single - Dimensional Array:

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

Memory Representation:

Index :	0	1	2	3	4
Value :	[10]	[20]	[30]	[40]	[50]

Multi - Dimensional Array:

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

Memory Representation:

Index :	0	1	2
Row 0 :	[1,	2,	3]
Row 1 :	[4,	5,	6]
Row 2 :	[7,	8,	9]

@Curious_.programmer

Looping through a 2D Array :

```
for (int i = 0 ; i < matrix .length ; i++) {
   for (int j = 0 ; j < matrix[i].length ; j++) {
      system.out.println (matrix [i][j] + "  ");
                        // output : 1 2 3 4 5 6 7 8 9
      system.out.println ();
}                                  //moves to the next line after each
                                     row
```

3. JAGGED Arrays :

A jagged array (also known as an "array of arrays") is an array whose elements are arrays themselves. Unlike a multi-dimensional array where every row has the same number of columns, a jagged array allows rows to have different numbers of columns.

```
int [][] jaggedArray = new int [3][];
  jaggedArray [0] = new int [2];
  jaggedArray [1] = new int [3];
  jaggedArray [2] = new int [1];
```

@ Curious_programmer

You can also initialize it directly:

```
int [][] jaggedArray = {
        {1, 2},
        {3, 4, 5},
        {6}
};
```

Accessing Elements in a Jagged Array:

```
system.out.println (jaggedArray[1][2]);
                    // output : 5 (second row, third column)
```

Looping through a Jagged Array:

```
for (int i=0; i<jaggedArray.length; i++) {
  for (int j=0; j < jaggedArray[i].length; j++) {
     system.out.println (jaggedArray[i][j] + " ");
                                  //output : 1 2 3 4 5 6
  }
  system.out.println ();
}
```

4. Array Manipulation Techniques :

a) Sorting Arrays :

Sorting is a common operation to arrange elements in a specific order (ascending or descending).

In java, you can use the Arrays.sort() method to sort arrays.

Sorting a 1D Array :

```
int[] arr = {5, 2, 8, 1, 4};
Arrays.sort(arr);
system.out.println(Arrays.toString(arr));
                    //output: [1, 2, 4, 5, 8]
```

Sorting a 2D Array (by rows) :

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

for (int= 0; i < matrix.length(); i++) {
    Arrays.sort(matrix[i]);
System.out.println(Arrays.toString(matrix[i]));
}                                   // sort each row
```

Output :

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

b) Searching in Arrays :

You can search for an element in an array using a linear search or binary search (for sorted arrays).

Linear Search (for unsorted arrays) :

```
int[]arr = {10, 20, 30, 40, 50};
int target = 30;
boolean found = false;

for (int i=0; i < arr.length; i++){
    if (arr[i] == target){
     found = true;
     system.out.println("Element found at index" + i);
                    //Output: Element found at index 2.
     break;
} }
if(!found){
    system.out.println("Element not found"); }
```

@ Curious_.programmer

Binary Search (for sorted arrays):

```
int[] arr = {1, 2, 4, 5, 8};
int target = 4;
int index = Arrays.binarySearch(arr, target);
System.out.println("Element found at index : "
                   + index);
        //output : 2 (index of 4)
```

@Curious-. programmer