## **Arrays in Java**

* Group of liked typed variable
* All arrays are dynamically allocated
* Array must be stored in contiguous memory(consecutive memory allocation)
* Arrays are objects in java so we can find the length of the array by using object property “length”
* Array variable can also be declatred like other variable with [] afte the data type
* Index starts from 0
* Arrays acan also be used as static field, loca; variable and method parameter

## **Creating, Initializing, and Accessing an Arrays**

### **One-Dimensional Arrays**

The general form of a one-dimensional array declaration is

-- type var-name[];

-- type[] var-name;

* Type declares the element type of the array
* Type determines the datatypes of each element that comprises the array

**Example 👍**

**// both are valid declarations**

**int intArray[];**

**int[] intArray;**

**// similar to int we can declare**

**// byte , short, boolean, long, float**

**// double, char**

**// an array of references to objects of**

**// the class MyClass (a class created by user)**

**MyClass myClassArray[];**

**// array of Object**

**Object[] ao,**

**// array of Collection**

**// of unknown type**

**Collection[] ca;**

**Note :**

**int intArray[];**

**int[] intArray;**

* **Int array is an array variable but there is no actual array exist**
* **It merely tells the compiler that the variable(int Array) will hold an array of integer type**
* **To link int array with the actual , physical array of integers , you must allocate one using new and assign it to int array**

## **Instantiating an Array in Java**

* When an array is declared, only a reference of an array is created.
* To create or give memory to the array, you create an array like this: The general form of new as it applies to one-dimensional arrays appears as follows:
* var-name = new type [size];
* Here, type specifies the type of data being allocated, size determines the number of elements in the array, and var-name is the name of the array variable that is linked to the array.
* To use new to allocate an array, you must specify the type and number of elements to allocate.
* Example:

**//declaring array**

**int intArray[];**

**// allocating memory to array**

**intArray = new int[20];**

**// combining both statements in oneint[]**

**intArray = new int[20];**

***Note:***

1. ***The elements in the array allocated by new will automatically be initialized to zero (for numeric types), false (for boolean), or null (for reference types).***
2. ***Obtaining an array is a two-step process.***

* ***First, you must declare a variable of the desired array type.***
* ***Second, you must allocate the memory to hold the array, using new, and assign it to the array variable.***

***Thus, in Java, all arrays are dynamically allocated.***

## **Array Literal in Java**

In a situation where the size of the array and variables of the array are already known, array literals can be used.

// Declaring array literal

int[] intArray = new int[]{ 1,2,3,4,5,6,7,8,9,10 };

* The length of this array determines the length of the created array.
* There is no need to write the new int[] part in the latest versions of Java.

### **Accessing Java Array Elements using for Loop**

* Each element in the array is accessed via its index.
* The index begins with 0 and ends at (total array size)-1.
* All the elements of array can be accessed using Java for Loop.
* // accessing the elements of the specified array

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

System.out.println("Element at index " + i + " : "+ arr[i]);

## **Arrays of Objects in Java**

An array of objects is created like an array of primitive-type data items in the following way.

Student[] arr = new Student[5]; //student is a user-defined class

Syntax:

-- data type[] arrName;

-- datatype arrName[];

-- datatype [] arrName;

Example :

**class** GFG {

**public** **static** **void** main (String[] args) {

**int** [] arr=**new** **int** [4];

// 4 is the size of arr

System.out.println("Array Size:"+arr.length);

}

}

OUTPUT :

Array Size:4

The student Array contains five memory spaces each of the size of student class in which the address of five Student objects can be stored. The Student objects have to be instantiated using the constructor of the Student class, and their references should be assigned to the array elements in the following way

**Example :**

**class Student {**

**public int roll\_no;**

**public String name;**

**Student(int roll\_no, String name)**

**{**

**this.roll\_no = roll\_no;**

**this.name = name;**

**}**

**}**

**// Elements of the array are objects of a class Student.**

**public class GFG {**

**public static void main(String[] args)**

**{**

**// declares an Array of Students**

**Student[] arr;**

**// allocating memory for 5 objects of type Student.**

**arr = new Student[5];**

**// initialize the first elements of the array**

**arr[0] = new Student(1, "aman");**

**// initialize the second elements of the array**

**arr[1] = new Student(2, "vaibhav");**

**// so on...**

**arr[2] = new Student(3, "shikar");**

**arr[3] = new Student(4, "dharmesh");**

**arr[4] = new Student(5, "mohit");**

**// accessing the elements of the specified array**

**for (int i = 0; i < arr.length; i++)**

**System.out.println("Element at " + i + " : "**

**+ arr[i].roll\_no + " "**

**+ arr[i].name);**

**}**

**}**

**Output**

**Element at 0 : 1 aman**

**Element at 1 : 2 vaibhav**

**Element at 2 : 3 shikar**

**Element at 3 : 4 dharmesh**

**Element at 4 : 5 mohit**

### **Q : What happens if we try to access elements outside the array size?**

Ans : JVM throws **ArrayIndexOutOfBoundsException** to indicate that the array has been accessed with an illegal index. The index is either negative or greater than or equal to the size of an array.

## **Multidimensional Arrays in Java**

* Multidimensional arrays are arrays of arrays with each element of the array holding the reference of other arrays.
* These are also known as [Jagged Arrays](https://www.geeksforgeeks.org/jagged-array-in-java/).
* A multidimensional array is created by appending one set of square brackets ([]) per dimension.

#### Syntax of Java Multidimensional Array

There are 2 methods to declare Java Multidimensional Arrays as mentioned below:

-- datatype [][] arrayrefvariable;

-- datatype arrayrefvariable[][];

Example :

public static void main(String[] args)

{

// Syntax

int[][] arr = new int[3][3];

// 3 row and 3 column

// Number of Rows

System.out.println("Number of Rows:"+ arr.length);

// Number of Columns

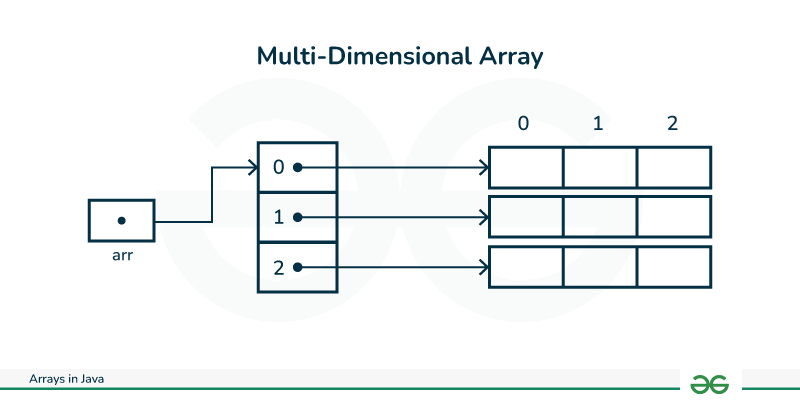
System.out.println("Number of Columns:"+ arr[0].length);

}

**Output**

Number of Rows:3

Number of Columns:3



### **Multidimensional Array Declaration**

int[][] intArray = new int[10][20]; //a 2D array or matrix

int[][][] intArray = new int[10][20][10]; //a 3D array

### **Passing Arrays to Methods**

Like variables, we can also pass arrays to methods. For example, the below program passes the array to method *sum* to calculate the sum of the array’s values.

**Example :**

**public class Test {**

**// Driver method**

**public static void main(String args[])**

**{**

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

**// passing array to method m1**

**sum(arr);**

**}**

**public static void sum(int[] arr)**

**{**

**// getting sum of array values**

**int sum = 0;**

**for (int i = 0; i < arr.length; i++)**

**sum += arr[i];**

**System.out.println("sum of array values : " + sum);**

**}**

**}**

**Output**

**sum of array values : 15**

### **Returning Arrays from Methods**

As usual, a method can also return an array. For example, the below program returns an array from method m1.

**Example :**

**class Test {**

**// Driver method**

**public static void main(String args[])**

**{**

**int arr[] = m1();**

**for (int i = 0; i < arr.length; i++)**

**System.out.print(arr[i] + " ");**

**}**

**public static int[] m1()**

**{**

**// returning array**

**return new int[] { 1, 2, 3 };**

**}**

**}**

**Output**

**1 2 3**

### **Class Objects for Arrays**

Every array has an associated Class object, shared with all other arrays with the same component type.

**class** Test {

**public** **static** **void** main(String args[])

{

**int** intArray[] = **new** **int**[3];

**byte** byteArray[] = **new** **byte**[3];

**short** shortsArray[] = **new** **short**[3];

// array of Strings

String[] strArray = **new** String[3];

System.out.println(intArray.getClass());

System.out.println(

intArray.getClass().getSuperclass());

System.out.println(byteArray.getClass());

System.out.println(shortsArray.getClass());

System.out.println(strArray.getClass());

}

}

**Output**

**class [I**

**class java.lang.Object**

**class [B**

**class [S**

**class [Ljava.lang.String;**

#### **Explanation of the above method:**

1. The string “[I” is the run-time type signature for the class object “array with component type int.”
2. The only direct superclass of an array type is [java.lang.Object](https://www.geeksforgeeks.org/object-class-in-java/).
3. The string “[B” is the run-time type signature for the class object “array with component type byte.”
4. The string “[S” is the run-time type signature for the class object “array with component type short.”
5. The string “[L” is the run-time type signature for the class object “array with component type of a Class.” The Class name is then followed.

## **Java Array Members**

Now, as you know that arrays are objects of a class, and a direct superclass of arrays is a class Object. The members of an array type are all of the following:

* The public final field length contains the number of components of the array. Length may be positive or zero.
* All the members are inherited from class Object; the only method of Object that is not inherited is its [clone](https://www.geeksforgeeks.org/clone-method-in-java-2/) method.
* The public method clone() overrides the clone method in class Object and throws no [checked exceptions](https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/).

### Arrays Types and Their Allowed Element Types

| Array Types | Allowed Element Types |
| --- | --- |
| Primitive Type Arrays | Any type which can be implicitly promoted to declared type. |
| Object Type Arrays | Either declared type objects or it’s child class objects. |
| Abstract Class Type Arrays | Its child-class objects are allowed. |
| Interface Type Arrays | Its implementation class objects are allowed. |

## **Cloning of Single-Dimensional Array in Java**

When you clone a single-dimensional array, such as Object[], a “deep copy” is performed with the new array containing copies of the original array’s elements as opposed to references.

**class Test {**

**public static void main(String args[])**

**{**

**int intArray[] = { 1, 2, 3 };**

**int cloneArray[] = intArray.clone();**

**// will print false as deep copy is created**

**// for one-dimensional array**

**System.out.println(intArray == cloneArray);**

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

**System.out.print(cloneArray[i] + " ");**

**}**

**}**

**}**

**Output**

**false**

**1 2 3**

## **Cloning Multidimensional Array in Java**

**A clone of a multi-dimensional array (like Object[][]) is a “shallow copy,” however, which is to say that it creates only a single new array with each element array a reference to an original element array, but subarrays are shared.**

## Java

| **// Java program to demonstrate**  **// cloning of multi-dimensional arrays**    **class Test {**  **public static void main(String args[])**  **{**  **int intArray[][] = { { 1, 2, 3 }, { 4, 5 } };**  **int cloneArray[][] = intArray.clone();**  **// will print false**  **System.out.println(intArray == cloneArray);**  **// will print true as shallow copy is created**  **// i.e. sub-arrays are shared**  **System.out.println(intArray[0] == cloneArray[0]);**  **System.out.println(intArray[1] == cloneArray[1]);**  **}**  **}** |
| --- |

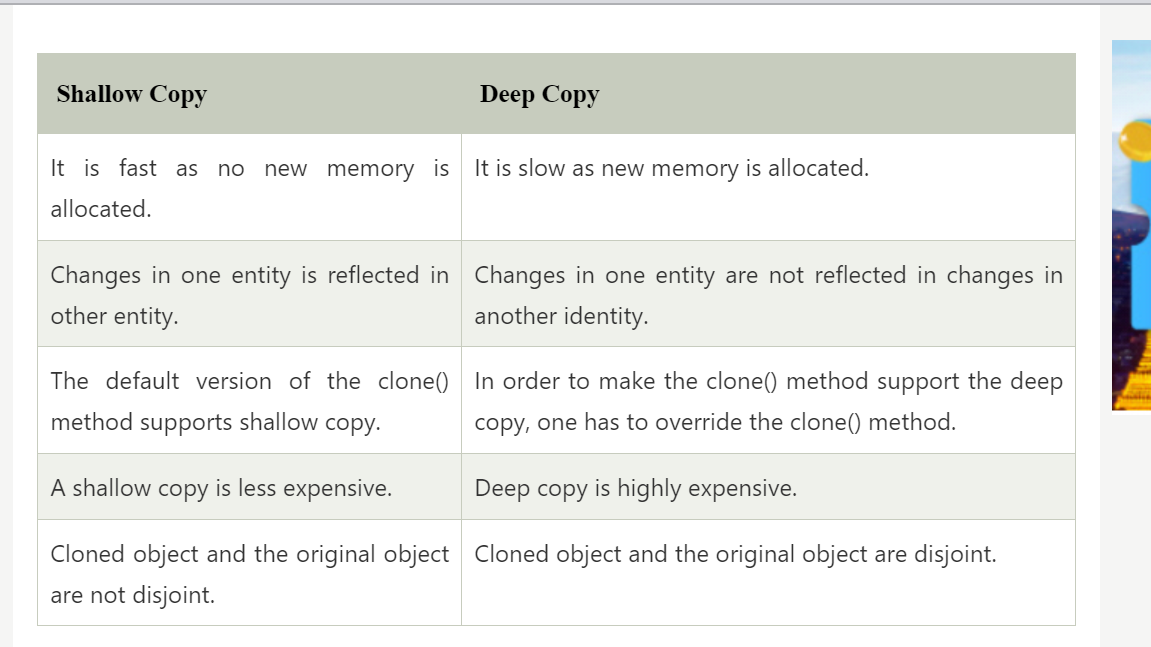
**Output**

**False**

**true**

**true**

### **NOTE : DEEP COPY VS SHALLOW COPY**



https://www.javatpoint.com/shallow-copy-vs-deep-copy-in-java

### **1. Can we specify the size of array as long?**

***No we can’t specify the size of array as long but we can specify it as int or short.***

### **2. Which is the direct superclass of an array in Java?**

***An*** [***Object***](https://www.geeksforgeeks.org/object-class-in-java/) ***is direct superclass of an array in Java.***

### **3. Which Interfaces are implemented by Arrays in Java?**

***Every array type implements the interfaces Cloneable and*** [***java.io.Serializable***](https://www.geeksforgeeks.org/serialization-in-java/)***.***

### **4. Can we alter the size of Array?**

***The size of the array cannot be altered(once initialized). However, an array reference can be made to point to another array.***

### Advantages

* Code Optimization: It makes the code optimized, we can retrieve or sort the data efficiently.
* Random access: We can get any data located at an index position.

### Disadvantages

* Size Limit: We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.