Java Arrays

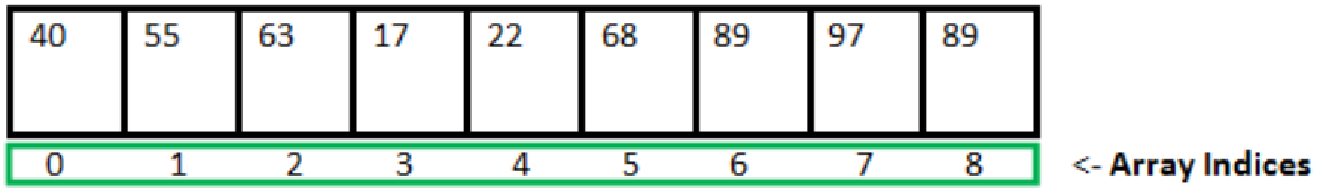
Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

Java array is an object which contains elements of a similar data type. Additionally, The elements of an array are stored in a contiguous memory location. It is a data structure where we store similar elements. We can store only a fixed set of elements in a Java array.

Arrays in Java are index-based, the first element of the array is stored at the 0th index, 2nd element is stored on the 1st index and so on.

1. In Java, all arrays are dynamically allocated. (discussed below)
2. Arrays are stored in contiguous memory [consecutive memory locations].
3. Since arrays are objects in Java, we can find their length using the object property *length*. This is different from C/C++, where we find length using sizeof.
4. A Java array variable can also be declared like other variables with [] after the data type.
5. The variables in the array are ordered, and each has an index beginning from 0.
6. Java arrays can also be used as a static field, a local variable, or a method parameter.
7. The **size** of an array must be specified by int or short value and not long.
8. The direct superclass of an array type is Object.
9. Every array type implements the interfaces Cloneable and java.io.Serializable.
10. This storage of arrays helps us randomly access the elements of an array [Support Random Access].
11. The size of the array cannot be altered(once initialized). However, an array reference can be made to point to another array.

An array can contain primitives (int, char, etc.) and object (or non-primitive) references of a class depending on the definition of the array. In the case of primitive data types, the actual values are stored in contiguous memory locations. In the case of class objects, the actual objects are stored in a heap segment.



Array Length = 9
First Index = 0
Last Index = 8

Types of Array in java

There are two types of arrays.

- Single Dimensional Array
- Multidimensional Array

Single Dimensional Array:

How to declare an array in Java?

In Java, here is how we can declare an array.

Syntax:

dataType[] arrayName;

dataType - it can be primitive data types like int, char, double, byte, etc. or Java objects

arrayName - it is an identifier

For example,

double[] data;

Here, data is an array that can hold values of type double.

But, how many elements can array this hold?

Good question! To define the number of elements that an array can hold, we have to allocate memory for the array in Java. For example,

Array Creation:

Syntax:
arrayname = new data type[size];

For Example:
// allocate memory
data = new double[10];

Here, the array can store 10 elements. We can also say that the size or length of the array is 10.

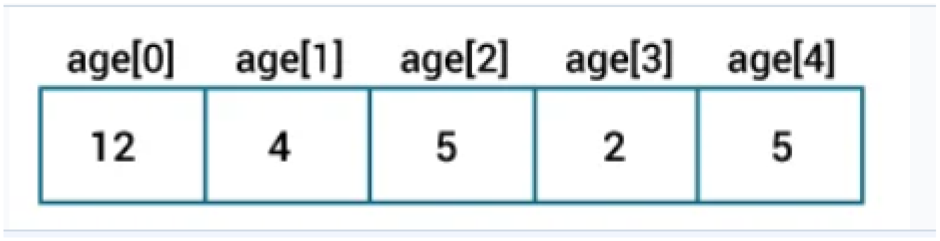
In Java, we can declare and allocate the memory of an array in one single statement.

For example,
double[] data = new double[10];

How to Initialize Arrays in Java?

// declare & create an array
int[] age = new int[5];

// initialize array
age[0] = 12;
age[1] = 4;
age[2] = 5;
..



Note:

- Array indices always start from 0. That is, the first element of an array is at index 0.
- If the size of an array is n, then the last element of the array will be at index n-1

Note:

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). Do refer to default array values in Java.

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.

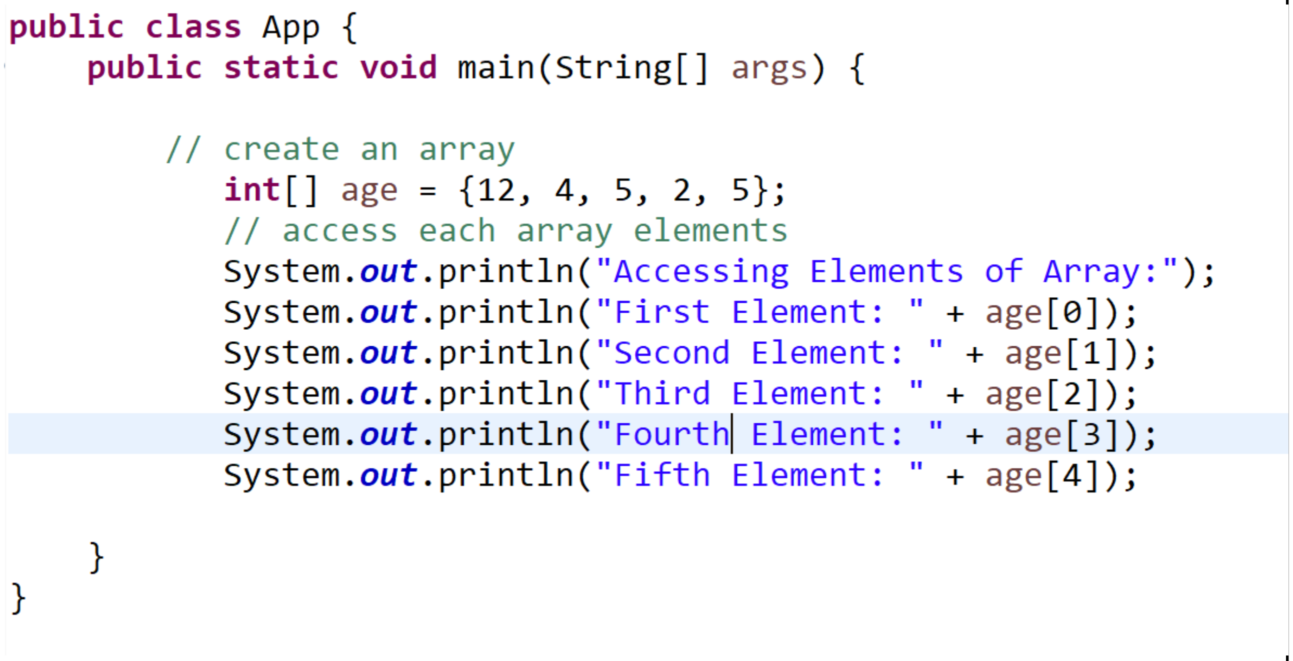
How to Access Elements of an Array in Java?

We can access the element of an array using the index number. Here is the syntax for accessing elements of an array,

// access array elements

array name[index]

Example:

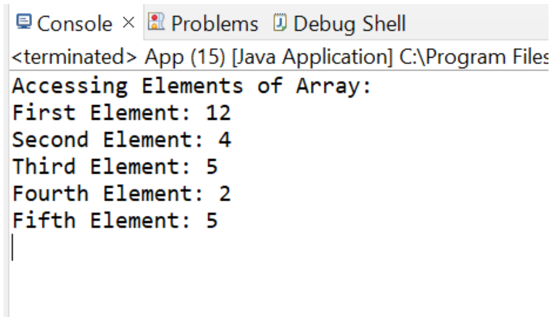


```
public class App {
    public static void main(String[] args) {

        // create an array
           int[] age = {12, 4, 5, 2, 5};
           // access each array elements
           System.out.println("Accessing Elements of Array:");
           System.out.println("First Element: " + age[0]);
           System.out.println("Second Element: " + age[1]);
           System.out.println("Third Element: " + age[2]);
           System.out.println("Fourth Element: " + age[3]);
           System.out.println("Fifth Element: " + age[4]);

    }
}
```

Result:


Console × Problems Debug Shell

<terminated> App (15) [Java Application] C:\Program Files

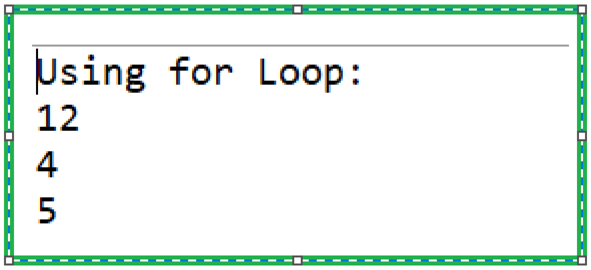
```
Accessing Elements of Array:
First Element: 12
Second Element: 4
Third Element: 5
Fourth Element: 2
Fifth Element: 5
```

Looping Through Array Elements

In Java, we can also loop through each element of the array. For example,

```java
public class App {
    public static void main(String[] args) {
        // create an array
        int[] age = { 12, 4, 5 };
        // loop through the array
        // using for loop
        System.out.println("Using for Loop:");
        for (int i = 0; i < age.length; i++) {
            System.out.println(age[i]);
        }

    }
}
```



```
Using for Loop:
12
4
5
```

In the above example, we are using the for Loop in Java to iterate through each element of the array.

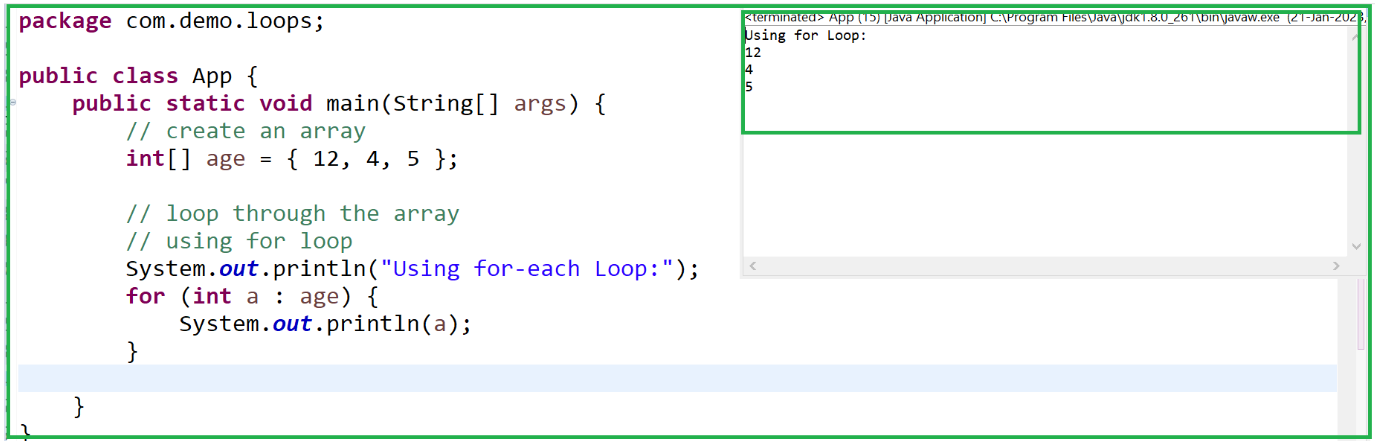
Notice the expression inside the loop,

age.length

Here, we are using the length property of the array to get the size of the array.

We can also use the for-each loop to iterate through the elements of an array. For example,

Example: Using the for-each Loop



```
package com.demo.loops;

public class App {
    public static void main(String[] args) {
        // create an array
        int[] age = { 12, 4, 5 };

        // loop through the array
        // using for loop
        System.out.println("Using for-each Loop:");
        for (int a : age) {
            System.out.println(a);
        }

    }
}
```

```
<terminated> App (15) [Java Application] C:\Program Files\Java\jdk1.8.0_261\bin\javaw.exe  (21-Jan-202
Using for Loop:
12
4
5
```

How to Initialize Arrays in Java in Single line?

In Java, we can initialize arrays during declaration. For example,

//declare and initialize and array
int[] age = {12, 4, 5, 2, 5};

Here, we have created an array named age and initialized it with the values inside the curly brackets.

Note that we have not provided the size of the array. In this case, the Java compiler automatically specifies the size by counting the number of elements in the array (i.e. 5).

Java Multidimensional Arrays

In this tutorial, we will learn about the Java multidimensional array using 2-dimensional arrays and 3-dimensional arrays with the help of examples.

Before we learn about the multidimensional array, make sure you know about Java array.

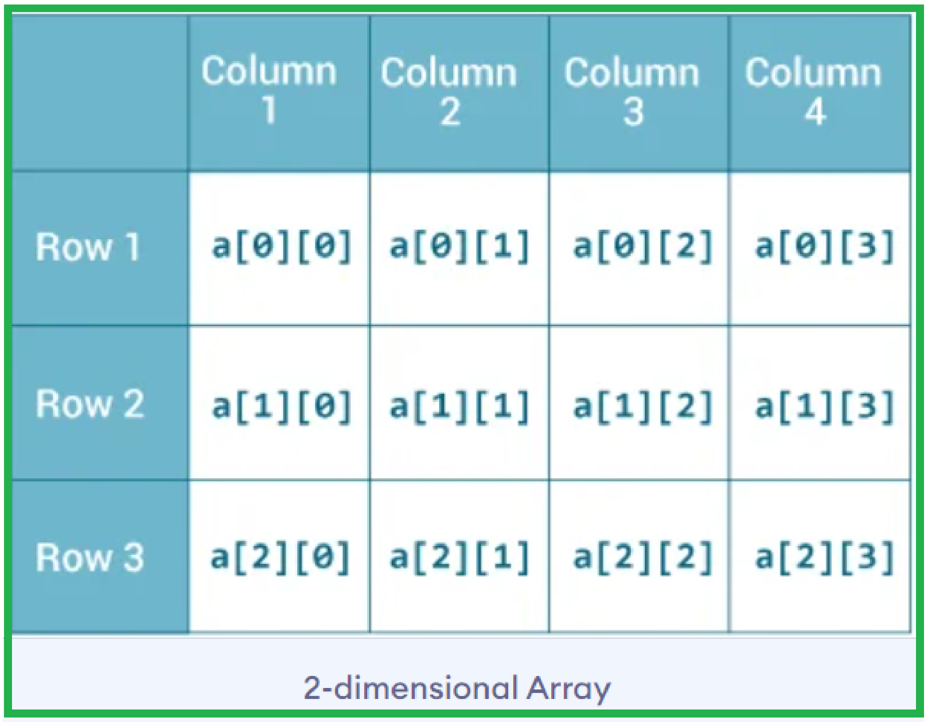
A multidimensional array is an array of arrays.

Each element of a multidimensional array is an array itself.

For example,

int[][] a = new int[3][4];

Here, we have created a multidimensional array named a. It is a 2-dimensional array, that can hold a maximum of 12 elements,



	Column 1	Column 2	Column 3	Column 4
Row 1	a[0][0]	a[0][1]	a[0][2]	a[0][3]
Row 2	a[1][0]	a[1][1]	a[1][2]	a[1][3]
Row 3	a[2][0]	a[2][1]	a[2][2]	a[2][3]

2-dimensional Array

Remember, Java uses zero-based indexing, that is, indexing of arrays in Java starts with 0 and not 1.

Let's take another example of the multidimensional array. This time we will be creating a 3-dimensional array.

For example,

String[][][] data = new String[3][4][2];

Here, data is a 3d array that can hold a maximum of 24 (3*4*2) elements of type String.

Declaring 2D Arrays

In Java, 2D arrays are stored as arrays of arrays. Therefore, the way 2D arrays are declared is similar to 1D array objects. 2D arrays are declared by defining a data type followed by two sets of square brackets.

Examples:

```
int[][] twoDIntArray;

String[][] twoDStringArray;

double[][] twoDDoubleArray;
```

Another notes will be uploaded for Multidimensional Array