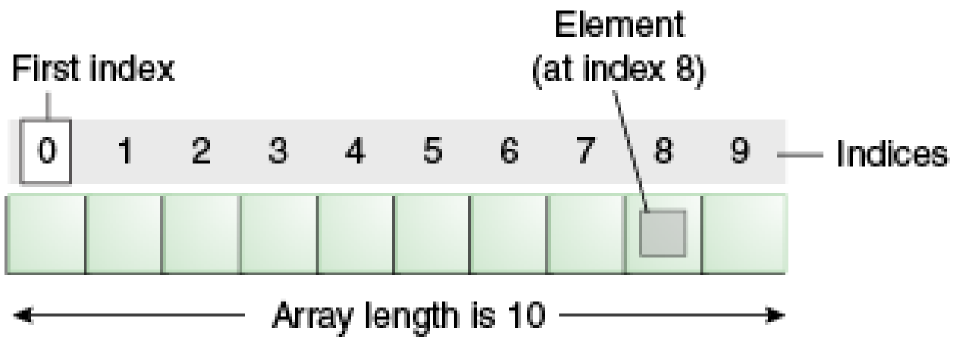
**Java Arrays**

An array is a collection of similar type of elements which has contiguous memory location.

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.

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

In Java, array is an object of a dynamically generated class. Java array inherits the Object class, and implements the Serializable as well as Cloneable interfaces. We can store primitive values or objects in an array in Java.



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.

ypes of Array in java

There are two types of array.

Single Dimensional Array

Multidimensional Array

Single Dimensional Array in Java

Syntax to Declare an Array in Java

dataType[] arrayName;

(or)

dataType []arrayName;

(or)

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.

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

**arrayName= new datatype[size];**

For example,

// declare an array

**double[] data;**

// 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];

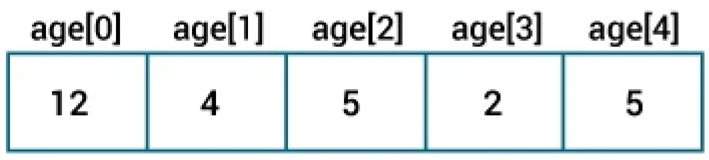
In Java, we can initialize arrays during declaration. For example,

//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).



In the Java array, each memory location is associated with a number. The number is known as an array index. We can also initialize arrays in Java, using the index number. For example,

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

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

**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[index]
```

## Access Array Elements using index number.

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

        }

}
```

```
class Main
{
        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++)  //length is the property of array
                {
                   System.out.println(age[i]);
                }
        }
}
```

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

**For-each Loop for Java Array**

We can also print the Java array using for-each loop. The Java for-each loop prints the array elements one by one. It holds an array element in a variable, then executes the body of the loop. The syntax of the for-each loop is given below:

```
        for(data_type variable:array)
        {
                //body of the loop
        }
```

Here,

array - an array or a collection

variable - each item of array/collection is assigned to this variable

dataType - the data type of the array/collection

For example,

```
class Main
{
         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);
              }
       }
}
```

## Compute Sum and Average of Array Elements

```
class Main
{
        public static void main(String[] args)
       {
              int[] numbers = {2, -9, 0, 5, 12, -25, 22, 9, 8, 12};
              int sum = 0;
              Double average;
              // access all elements using for each loop add each element in sum
              for (int number: numbers)
              {
                       sum += number;
              }
                // get the total number of elements
              int arrayLength = numbers.length;
                // calculate the average convert the average from int to double
              average =  ((double)sum / (double)arrayLength);
              System.out.println("Sum = " + sum);
              System.out.println("Average = " + average);
       }
}
```

**average = ((double)sum / (double)arrayLength);**
**As you can see, we are converting the int value into double. This is called type casting in Java. To learn more about typecasting, visit Java Type Casting.**

**Using for loop**

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

                char[] vowels = {'a', 'e', 'i', 'o', 'u'};

                // iterating through an array using a for loop
```

```
                    for (int i = 0; i < vowels.length; ++ i)
                   {
                            System.out.println(vowels[i]);
                    }
          }
}
```

Output:

```
          a
          e
          i
          o
          u
```

**Using for-each Loop**

```
class Main
{
          public static void main(String[] args)
         {
                    char[] vowels = {'a', 'e', 'i', 'o', 'u'};
                    // iterating through an array using the for-each loop
                    for (char item: vowels)
                    {
                            System.out.println(item);
                    }
          }
}
```

```
//Java Program to demonstrate the way of passing an array to method.
class Testarray2
{
         //creating a method which receives an array as a parameter
         static void min(int arr[])
         {
                  int min=arr[0];
                  for(int i=1;i<arr.length;i++)
                   if(min>arr[i])
                            min=arr[i];
                  System.out.println(min);
         }
         public static void main(String args[])
         {
                  int a[]={33,3,4,5};//declaring and initializing an array
                  min(a);//passing array to method
```

```
        }
}
```

## Multidimensional Array in Java

In such case, data is stored in row and column-based index (also known as matrix form).

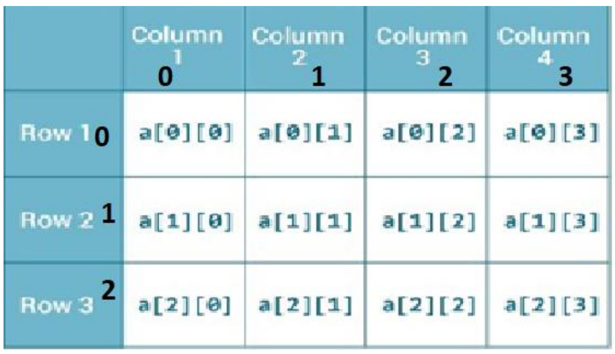
Syntax to Declare Multidimensional Array in Java

dataType[][] arrayName; (or)

dataType arrayName [][];

example:

```
        int[][] arr=new int[3][3];//3 row and 3 column
```



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

Example to initialize Multidimensional Array in Java

```
arr[0][0]=1;
arr[0][1]=2;
arr[0][2]=3;
arr[1][0]=4;
arr[1][1]=5;
arr[1][2]=6;
arr[2][0]=7;
arr[2][1]=8;
arr[2][2]=9;
```

```
//Java Program to illustrate the use of multidimensional array
class Matrix1
{
        public static void main(String args[])
        {
                //declaring and initializing 2D array
                int arr[][]={{1,2,3},{2,4,5},{4,4,5}};
                //printing 2D array
                for(int i=0;i<3;i++)
```

```
                    {
                             for(int j=0;j<3;j++)
                            {
                                     System.out.print(arr[i][j]+" ");
                            }
                            System.out.println();
                    }
          }
}
```

## Concatenate Two Arrays using array copy

```
import java.util.Arrays;
public class Concat
{
   public static void main(String[] args)
   {
      int[] array1 = {1, 2, 3};
      int[] array2 = {4, 5, 6};
      int aLen = array1.length;
      int bLen = array2.length;
      int[] result = new int[aLen + bLen];
      System.arraycopy(array1, 0, result, 0, aLen);
      System.arraycopy(array2, 0, result, aLen, bLen);

      System.out.println(Arrays.toString(result));
   }
}
```

In the above program, we've two integer arrays array1 and array2.

In order to combine (concatenate) two arrays, we find its length stored in aLen and bLen respectively. Then, we create a new integer array result with length aLen + bLen.

Now, in order to combine both, we copy each element in both arrays to result by using arraycopy() function.

The arraycopy(array1, 0, result, 0, aLen) function, in simple terms, tells the program to copy array1 starting from index 0 to result from index 0 to aLen.

Likewise, for arraycopy(array2, 0, result, aLen, bLen) tells the program to copy array2 starting from index 0 to result from index aLen to bLen.

Concatenate Two Arrays without using arraycopy

```
import java.util.Arrays;

public class Concat {
    public static void main(String[] args) {
        int[] array1 = {1, 2, 3};
        int[] array2 = {4, 5, 6};

        int length = array1.length + array2.length;

        int[] result = new int[length];
        int pos = 0;
        for (int element : array1) {
            result[pos] = element;
            pos++;
        }

        for (int element : array2) {
            result[pos] = element;
            pos++;
        }

        System.out.println(Arrays.toString(result));
    }
}
```

Output

```
[1, 2, 3, 4, 5, 6]
```

In the above program, instead of using arraycopy, we manually copy each element of both arrays array1 and array2 to result.

We store the total length required for result, i.e. array1.length + array2. length. Then, we create a new array result of the length.

Now, we use the for-each loop to iterate through each element of array1 and store it in the result. After assigning it, we increase the position pos by 1, pos++.

Likewise, we do the same for array2 and store each element in result starting from the position after array1.