# Arrays

An array is a collection of similar types of data.

For example, if we want to store the names of 100 people then we can create an array of the string type that can store 100 names.

```
String[] array = new String[100];
```

Here, the above array cannot store more than 100 names. The number of values in a Java array is always fixed.

---

## How to declare an array in Java?

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

```
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?**

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

# How to Initialize Arrays in Java?

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

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

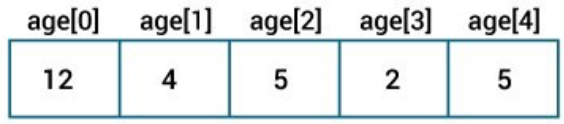
Here, we have created an array named age and initialized it with the values inside the curly brackets.

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;
..
```



Java Arrays initialization

**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*.

---

# 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[index]
```

Let's see an example of accessing array elements using index numbers.

### Example: Access Array Elements

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

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

Run Code

**Output**

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

In the above example, notice that we are using the index number to access each element of the array.

We can use loops to access all the elements of the array at once.

---

# Looping Through Array Elements

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

### Example: Using For Loop

```
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++) {
      System.out.println(age[i]);
    }
  }
```

```
}
```
Run Code

**Output**

```
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

```
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);
   }
 }
}
```
Run Code

**Output**

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

---

## Example: 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);
  }
}
```

Run Code

**Output**:

```
Sum = 36
Average = 3.6
```

In the above example, we have created an array of named *numbers*. We have used the `for...each` loop to access each element of the array.

Inside the loop, we are calculating the sum of each element. Notice the line,

```
int arrayLength = number.length;
```

Here, we are using the length attribute of the array to calculate the size of the array. We then calculate the average using:

```
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.

---

## Multidimensional Arrays

Arrays we have mentioned till now are called one-dimensional arrays. However, we can declare multidimensional arrays in Java.

A multidimensional array is an array of arrays. That is, each element of a multidimensional array is an array itself. For example,

```
double[][] matrix = {{1.2, 4.3, 4.0},
      {4.1, -1.1}
};
```

# Java Multidimensional Arrays

A multi-dimensional array is as *an array of arrays that stores homogeneous data in tabular form.*

Where are Multi-dimentional arrays use? - Multi-dimentional arrays are frequently used **to store data for mathematic computations, image processing, and record management.**

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`.

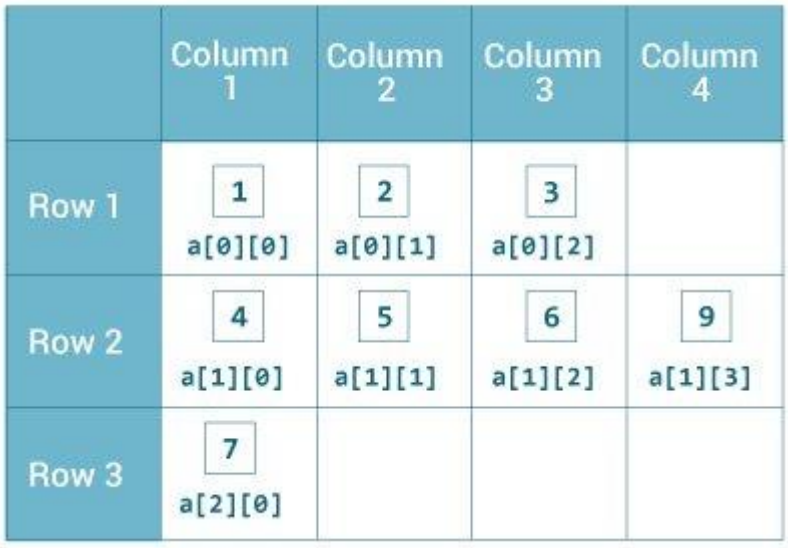
---

## How to initialize a 2d array in Java?

Here is how we can initialize a 2-dimensional array in Java.

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

As we can see, each element of the multidimensional array is an array itself. And also, unlike C/C++, each row of the multidimensional array in Java can be of different lengths.




Initialization of 2-dimensional Array

## Example: 2-dimensional Array

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

        // create a 2d array
        int[][] a = {
            {1, 2, 3},
            {4, 5, 6, 9},
            {7},
        };

        // calculate the length of each row
        System.out.println("Length of row 1: " + a[0].length);
        System.out.println("Length of row 2: " + a[1].length);
        System.out.println("Length of row 3: " + a[2].length);
    }
}
```

Run Code

**Output**:

```
Length of row 1: 3
Length of row 2: 4
Length of row 3: 1
```

In the above example, we are creating a multidimensional array named *a*. Since each component of a multidimensional array is also an array (`a[0]`, `a[1]` and `a[2]` are also arrays).

Here, we are using the `length` attribute to calculate the length of each row.

---

## Example: Print all elements of 2d array Using Loop

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

        int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };

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

Run Code

**Output**:

```
1
-2
3
-4
-5
6
9
7
```

We can also use the for...each loop to access elements of the multidimensional array. For example,

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

        // create a 2d array
        int[][] a = {
            {1, -2, 3},
            {-4, -5, 6, 9},
            {7},
        };

        // first for...each loop access the individual array
        // inside the 2d array
        for (int[] innerArray: a) {
            // second for...each loop access each element inside the row
            for(int data: innerArray) {
                System.out.println(data);
            }
        }
    }
}
```

Run Code

**Output**:

```
1
-2
3
-4
-5
6
9
7
```

In the above example, we are have created a 2d array named *a*. We then used `for` loop and `for...each` loop to access each element of the array.

---

# How to initialize a 3d array in Java?

Let's see how we can use a 3d array in Java. We can initialize a 3d array similar to the 2d array. For example,

```
// test is a 3d array
int[][][] test = {
        {
          {1, -2, 3},
          {2, 3, 4}
        },
        {
          {-4, -5, 6, 9},
          {1},
          {2, 3}
        }
};
```

Basically, a 3d array is an array of 2d arrays. The rows of a 3d array can also vary in length just like in a 2d array.

---

### Example: 3-dimensional Array

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

        // create a 3d array
        int[][][] test = {
            {
              {1, -2, 3},
              {2, 3, 4}
            },
            {
              {-4, -5, 6, 9},
              {1},
              {2, 3}
            }
        };
```

```
        // for..each loop to iterate through elements of 3d array
        for (int[][] array2D: test) {
            for (int[] array1D: array2D) {
                for(int item: array1D) {
                    System.out.println(item);
                }
            }
        }
    }
}
```

Run Code

**Output**:

```
1
-2
3
2
3
4
-4
-5
6
9
1
2
3
```

**Program below uses for loops to access the array elements and print them to the console**

```
public class Main
{
    public static void main(String[] args) {
        //initialize 3-d array
        int[][][] myArray = { { { 1, 2, 3 }, { 4, 5, 6 } },  { { 1, 4, 9 }, { 16, 25, 36 } },
                { { 1, 8, 27 }, { 64, 125, 216 } } };
        System.out.println("3x2x3 array is given below:");
        //print the 3-d array
        for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 2; j++) {
        for (int k = 0; k < 3; k++) {
        System.out.print(myArray[i][j][k] + "\t");
        }
        System.out.println();
                    }
        System.out.println();
        }
    }
}
```

**This program also uses enhanced for loop to traverse through the array and display its elements.**

```
public class Main
{
    public static void main(String[] args) {
        //initialize 3-d array
        int[][][] intArray = {
              {{10, 20, 30},{20, 40, 60}},
              { {10, 30,50,70},{50},{80, 90}}
              };
    System.out.println("Multidimensional Array (3-d) is as follows:");
    // use for..each loop to iterate through elements of 3d array
    for (int[][] array_2D: intArray) {
        for (int[] array_1D: array_2D) {
            for(intelem: array_1D) {
            System.out.print(elem + "\t");
             }
           System.out.println();
        }
        System.out.println();
    }
    }
}
```