Java 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?

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,

// 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];

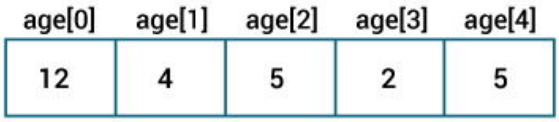
// initialize array

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) {
  // 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]);
 }
}
```

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}
};
```

Here, we have created a multidimensional array named matrix. It is a 2-dimensional array.

Let's put your knowledge of Java Array (With Examples) to the test.

Challenge:

Write a function to calculate the average of an array of numbers.

- Return the average of all numbers in the array arr with the size arrSize.
- For example, if arr[] = {10, 20, 30, 40} and arrSize = 4, the expected output is **25**.

```
public class Solution {
   public static double calculateAverage(int[] arr, int arrSize)
  {
   }
}
```
