**Array in Java**

Sometimes in our programs we need to deal with collection of data that connected with some logic together. It would be great to put them all in one box and carry them all together. The array is data type for that exact purpose, it can hold collection of data.

The array is a collection of data of the same type.

* The array is a fixed-size collection of data. We need to specify the size of the array while creating it, and the size cannot be changed.
* We have random access to each element in the array by using its index. Indexes start with 0.
* Array works with references as Objects do.

Let’s see the syntax:

String[] colors = new String[5];  
colors[0] = "red";  
colors[1] = "blue";  
colors[2] = "white";  
colors[3] = "yellow";  
colors[4] = "green";

In above example we created array of type String with size 5.

[] — square brackets after type means it’s array.

new String[5] — here we are specifying the size for our new array.

colors[0] = "red";

by using the reference of our array and the index of element we can assign the value. Also, to read the value we use similar syntax

System.out.println(colors[0]); // red

Let’s see one more example

int[] numbers = new int[3];  
numbers[0] = 11;  
numbers[1] = 25;  
numbers[2] = 39;

In this example we have created array of type int with size 3. The data type is int but syntax looks similar.

We can create array and assign values in the same statement as well

String[] names = {"John", "Bek", "Alex", "Donald"};  
System.out.println(names[0]); // John  
System.out.println(names[3]); // Donald

We can create array of almost any data types including all primitives and objects as well. Even we can create array of array, basically, when element of array is another array. We call them multidimensional arrays.

int[][] numbers = new int[5][5];  
numbers[0][0] = 1;  
numbers[0][1] = 2;  
numbers[0][2] = 3;  
// ..

Another detail we need to know about arrays. The array values initially have default values

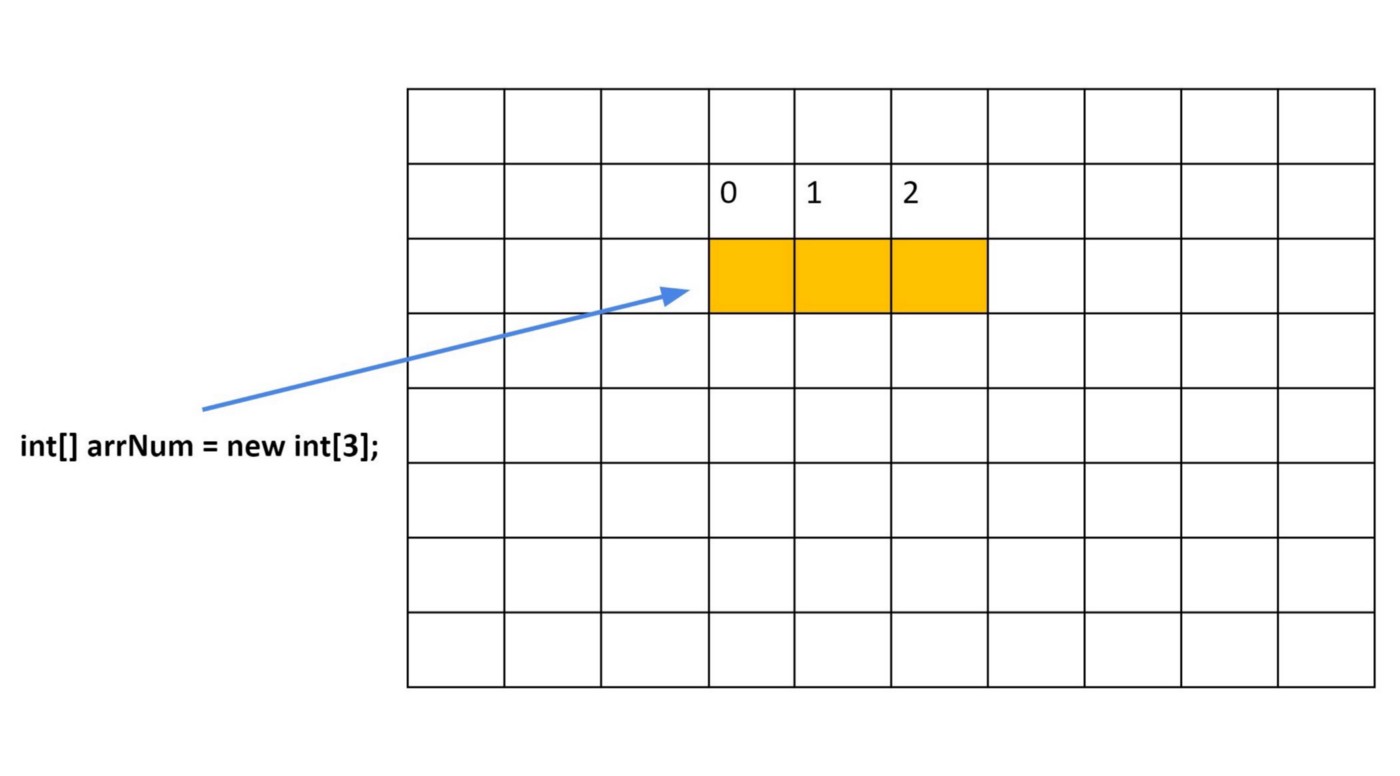
int[] arrNum = new int[3];  
System.out.println(arrNum[0]); // 0  
System.out.println(arrNum[1]); // 0  
System.out.println(arrNum[2]); // 0

String[] colors = new String[3];  
System.out.println(colors[0]); // null  
System.out.println(colors[1]); // null  
System.out.println(colors[2]); // null

The default value for objects is null, for numeric data type is 0, and for boolean is false.

Ok, now we are familiar with syntax, let’s talk more about how array works internally and about the fixed size.

We created an array of type int with size 3 in the below example.  
int[] arrNum = new int[3];  
If we will imagine memory as the canvas it will look like this:

Java will allocate space for 3 elements and the name of the array(reference) will point there. You see, all elements of the array are stored one by one after each other. So that’s why we have random access with arrays. Each element has an index that starts from 0.

Let’s see what will happen if try to access 4th element in our array which does not exist because we created array of size 3.

int[] arrNum = new int[3];  
arrNum[3] = 9; // ArrayIndexOutOfBoundsException

We will get runtimeArrayIndexOutOfBoundsException exception. Compiler cannot catch this issue because it does know the size of array and only during the runtime, we will get this exception because element with index 3 does not exist for this array.

Ok, we said array works with references, similarly as objects. Let’s take a look an example

import java.util.Arrays;  
  
public class Main {  
 public static void main(String[] args) {  
 int[] myArr = {1, 55, 3, 8, 21, 25, 73, 90};  
   
 // [1, 55, 3, 8, 21, 25, 73, 90]   
 System.*out*.println(Arrays.*toString*(myArr));   
  
 *makeAllSeven*(myArr);   
  
 // [7, 7, 7, 7, 7, 7, 7, 7]  
 System.*out*.println(Arrays.*toString*(myArr));   
 }  
  
 public static void makeAllSeven(int[] arrNum) {  
 for (int i = 0; i < arrNum.length; i++) {  
 arrNum[i] = 7;  
 }  
 }  
}

* we have our void method makeAllSeven that accepts int[] arrNum as argument and will assign value to each element as 7.
* we can see that after using this method the original value of array got changed. The reason it got changed because we are passing the reference that points to actual array in the memory. The argument of method will simply take a copy of reference. It means it will also point to the original object. That’s the reason in the main method, when use our myArr references, we see the all sevens after the method run.

We can check the size of our array by using length instance variable. It’s not a method so we don’t put () after it.

String[] colors = new String[6];  
System.out.println(colors.length); // 6

let’s see another example

public void printArray(String[] arr) {  
 // we can use .length to loop over our array  
 for (int i = 0; i < arr.length; i++) {  
 System.out.println(arr[i]);  
 }  
}

we can use for each loop to loop over our array as well

public void printArray(String[] arr) {  
 for (String str : arr) {  
 System.out.println(arr);  
 }  
}

if we would like to print array values as string, to sort or just compare two arrays with each other, we can use java.util.Arrays class. It’s helper class and it has many usefully static methods to work with arrays.

import java.util.Arrays;  
  
public class Main {  
 public static void main(String[] args) {  
 int[] myArr = {1, 55, 3, 8, 21, 25, 73, 90};  
  
 // print using Arrays  
 // [1, 55, 3, 8, 21, 25, 73, 90]  
 System.*out*.println(Arrays.*toString*(myArr));  
  
 // sort using Arrays  
 Arrays.*sort*(myArr);  
  
 // [1, 3, 8, 21, 25, 55, 73, 90]  
 System.*out*.println(Arrays.*toString*(myArr));  
   
 int[] myArr2 = {0, 4, 5, 3}; // compare two arrays using Arrays  
 System.*out*.println(Arrays.*equals*(myArr, myArr2));  
 }  
}

Let’s discuss the advantages and disadvantages of the array. The first advantage is random access. If we know the correct index we can read or update elements value with one step using the index.

arrNum[0] = 7; // assign to first element value 7

In another hand, the biggest disadvantage is fixed size. For example, I need an array in my program and I’m not quite sure how many elements it should have. I will create an array with the approximate size and let’s say in the middle of my program I need to put more elements than the array size. I will need to resize my array; resizing is expensive because I need to create a new array with a bigger size and then copy all elements from an existing array. Creating a new array with a different size is the only way to resize an array.