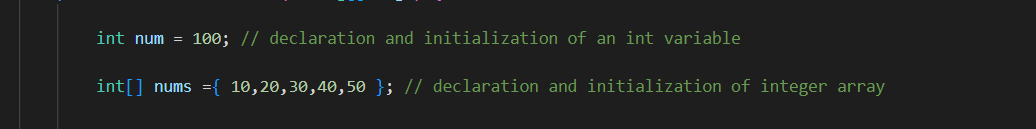
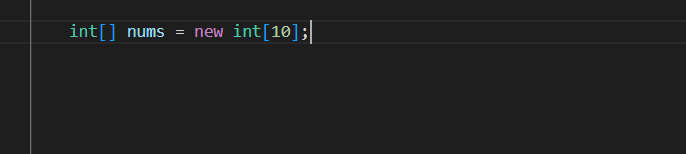


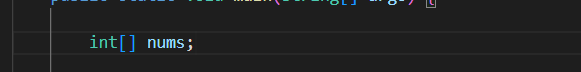
We see we have two ways of declaring and initializing arrays



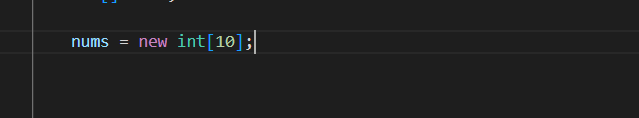
Declaration and Initialization of Array



it is both declaration and initialization

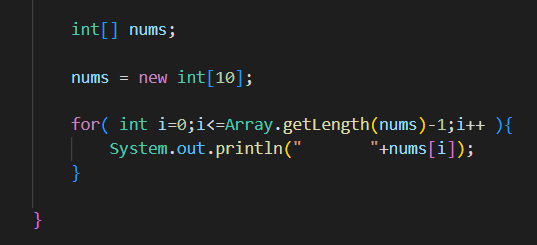


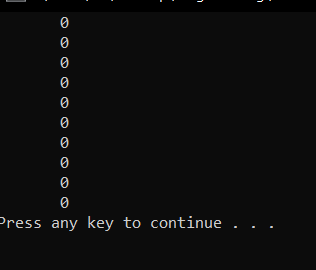
This is declaration



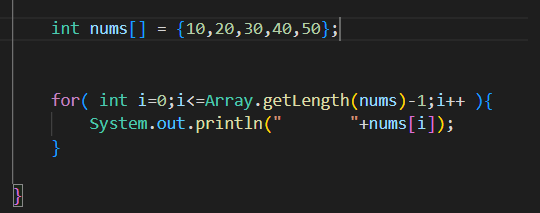
This is initialization of array

Now only the array of defined size is created in the heap memory

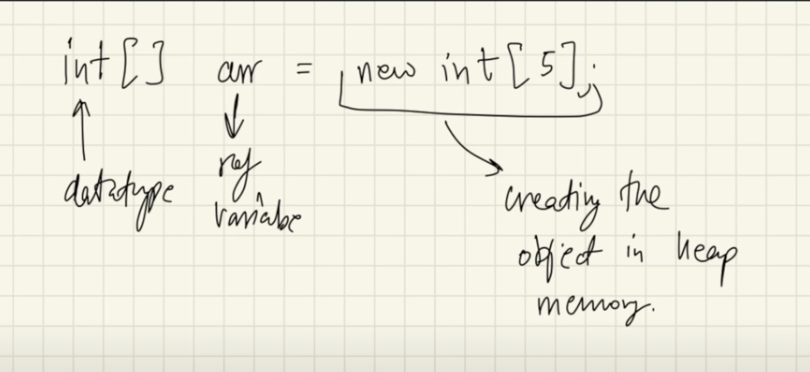




When we do not put any values, the array is filled with all zeroes

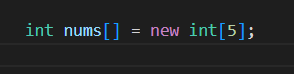


Either way of array creation is fine



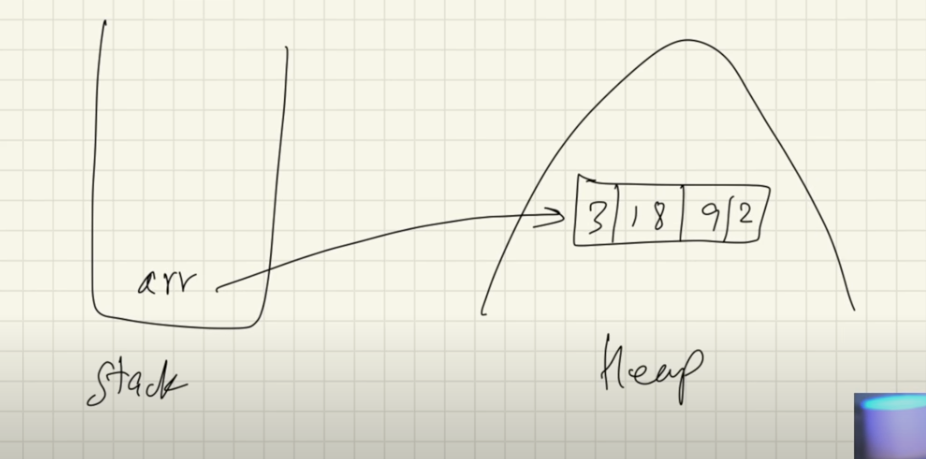
**Dynamic Memory Allocation**

It means when the memory is allocated at the run time / execution time



here, nums is just a reference variable, this is created at the compile time

But the RHS side, the actual memory allocation happens at the run time, this is DMA

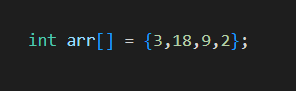


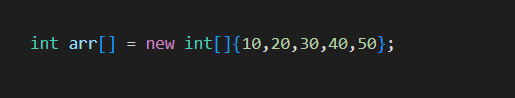
here arr is just a reference variable, which is created at the compile time and stored in the stack memory

and the actual memory allocation for the array occurs during the run time at the heap memory

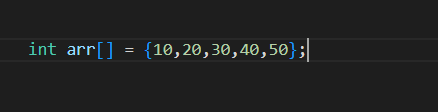
And the reference variable points to the allocated memory in the heap

this will be written something like this in code





This can also be written as



Both means same thing

So it’s not like in this one the array lives in stack, this is just a way of initializing array without new keyword, so it might seem like it lives in stack, but internally there is new keyword, and this array lives in heap not on stack

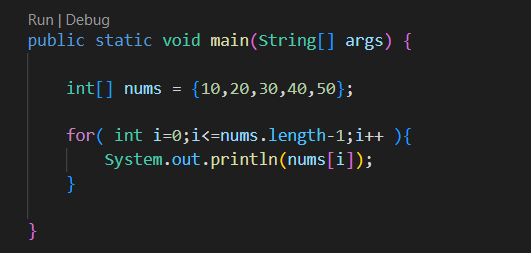
**Arrays in java may or may not be continuous**

In C and C++ it is absolute that the arrays occupies the memory in continuous form, but in Java the arrays are also objects and they are stored in heap, and in JLS ( Java Language Specifications ) it is specifically mentioned that heap objects may or may not be continuous

It is because of JVM ( Java Virtual Machine )

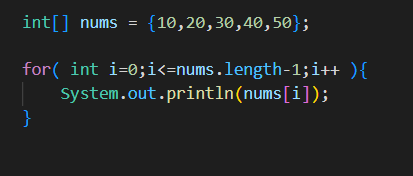
**Inputs in array**

We can use for loop or we can explicitely put values

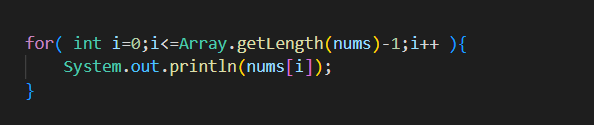


**To get length of array**

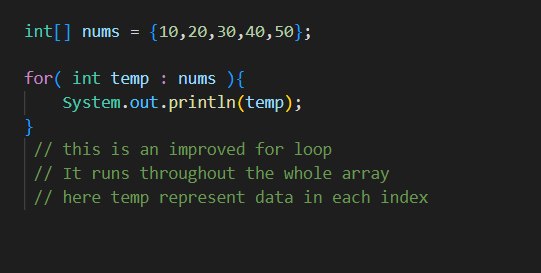
1. We can use .length property

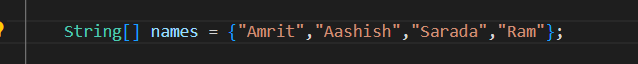


1. We can use Array.getLength(array\_name) ;



**An improved for loop**

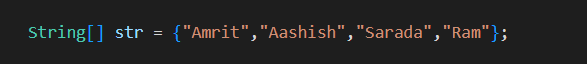


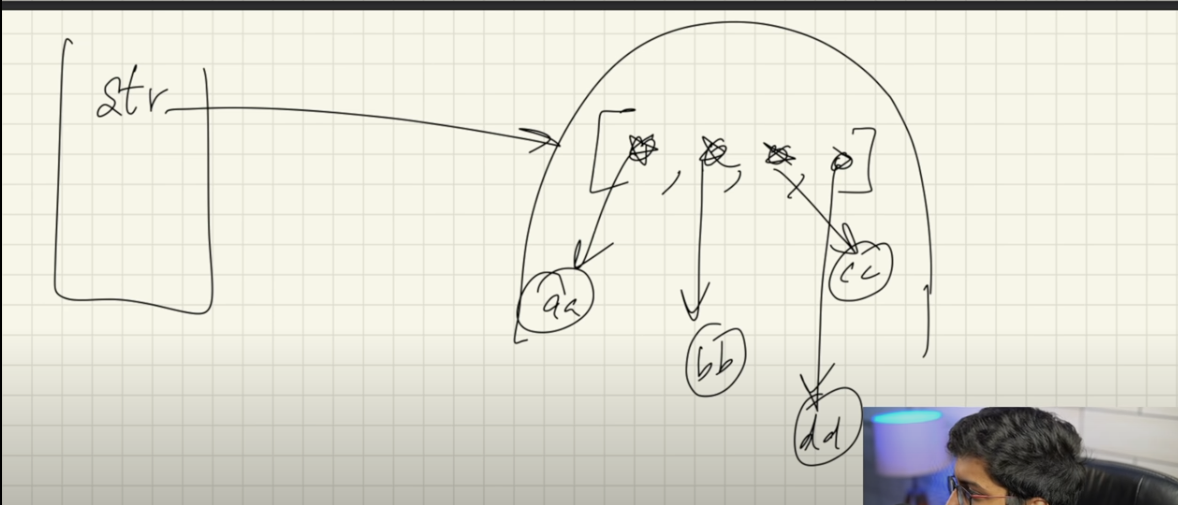


Here the LHS of the above is the declaration of the reference variable of the array

This is created during the compile time

The RHS of above is the actual memory allocation for the array in the heap which happens during the run time





here str is reference variable in the stack, pointing to the array living in the heap

Also the array is of String type

And in Java Strings are objects

So it is array of Objects

And objects are not primitive data types, they are reference data types

So here each index of the array is ( pointing to / referencing to ) the String object in the heap memory

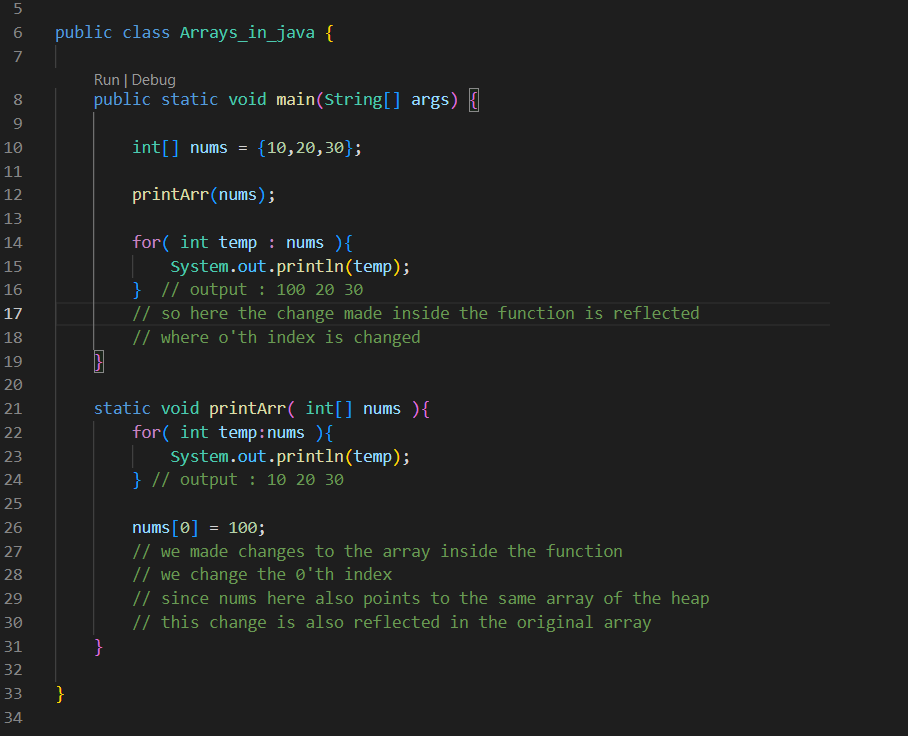
here str[0] , str[1], str[2], str[3] are the reference variables to the strings inside the array

**Passing array to a function**

As we know the name of array is the reference variable that is created during compilation, that lives in the stack and points to the array in the heap

So passing an array to a function is same as passing reference, where a copy of reference is passed

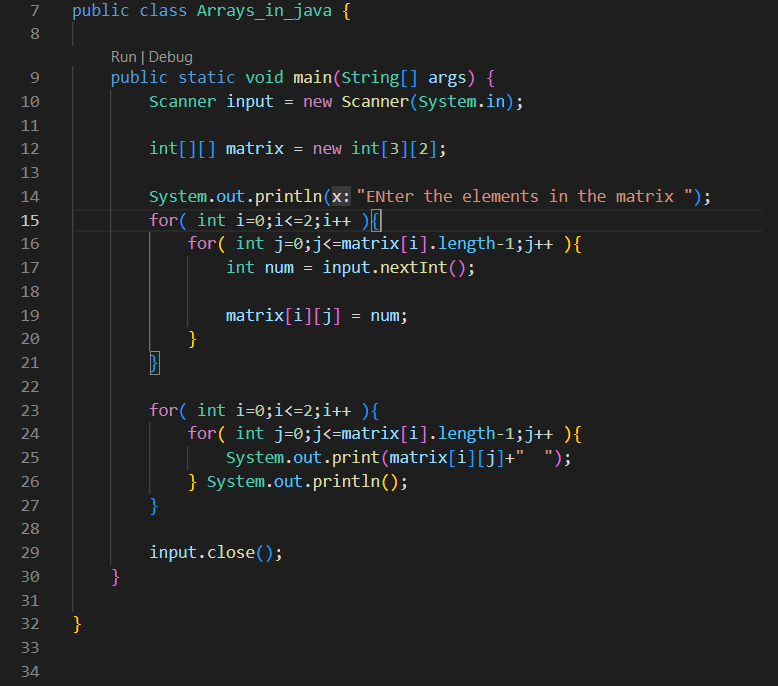
Inside the called function the passed copied reference of the array also points to the same array living in the heap, so any changes made to the array inside that function also reflected in the original array

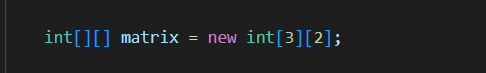


**Multidimensional arrays**

**2 Dimensional arrays**

A 2D array is just an array of arrays





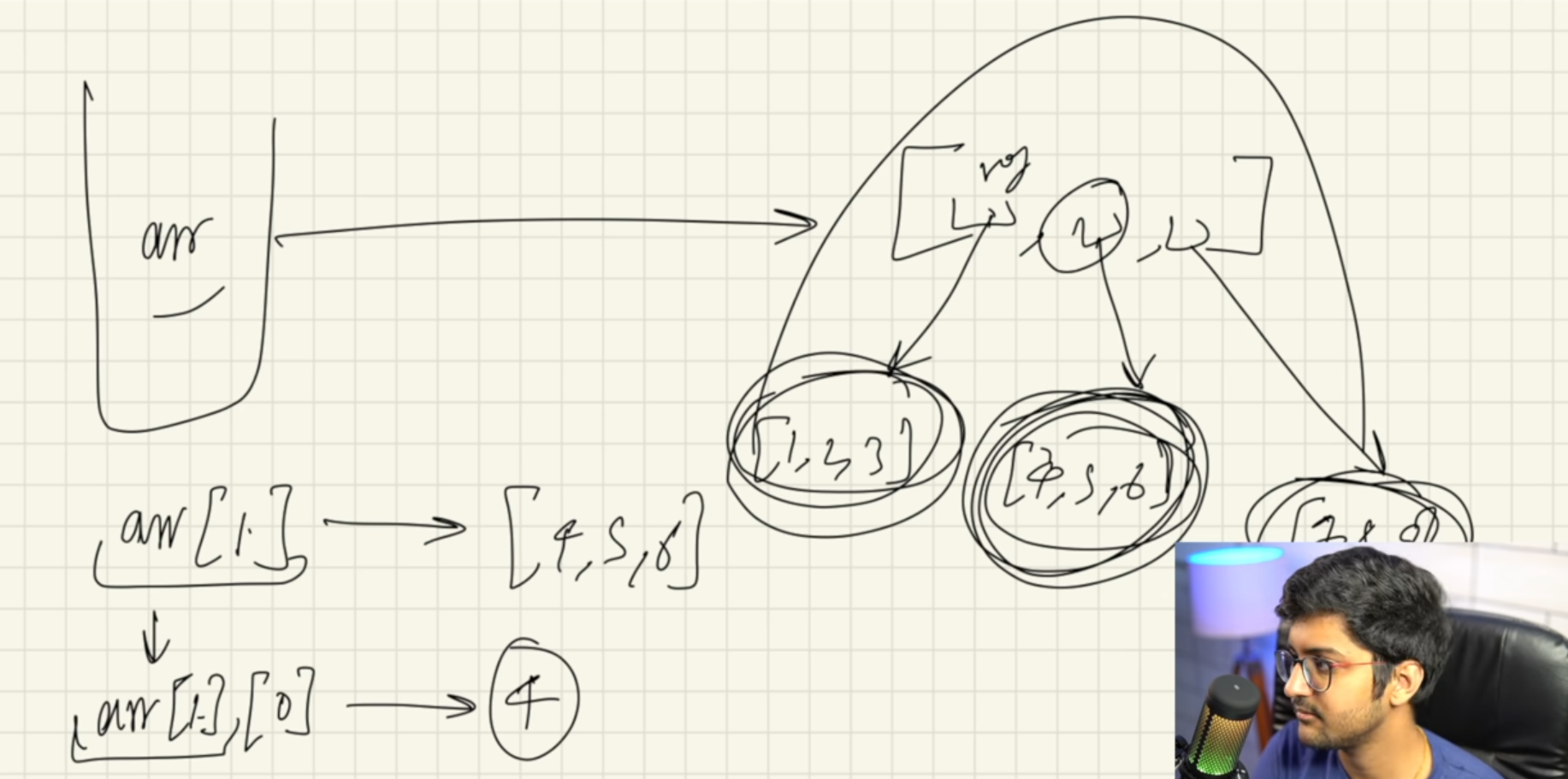
here inside matrix, we have three arrays each of size two

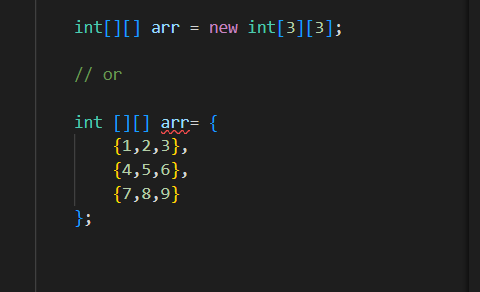
here each index of matrix array matrix[0], is pointing to another of length 2

**We can also view as rows and columns**

rows number is actually the number of arrays inside the big array

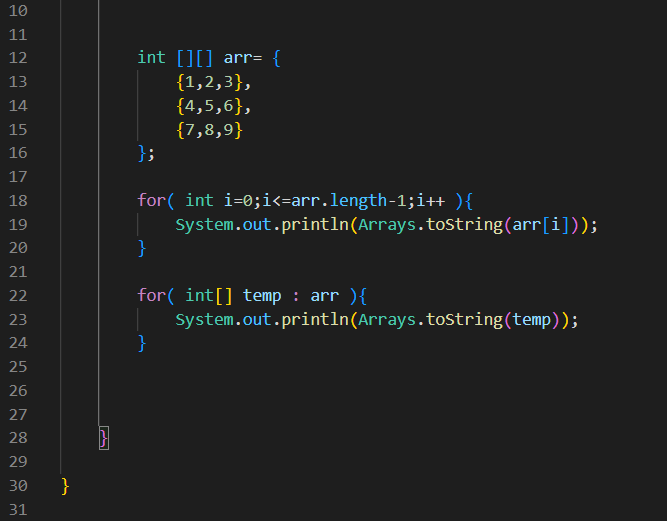
column number is the size of each arrays inside the big array



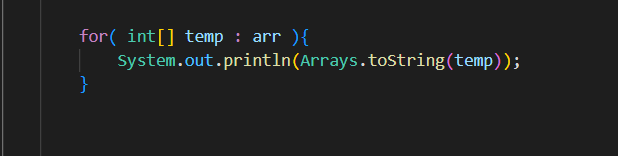


**Printing 2D array**

this is interesting, try to visualize it

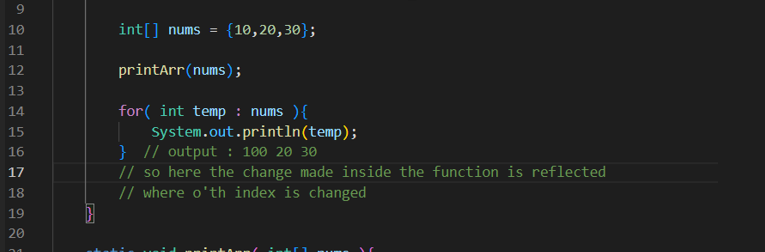


We have two methods od printing the 2D array



**Note : Yesmaa team vanaeko bhitra ko individual array ho**

previously we had



**Note : yesmaa temp vanaeko each item inside the nums array**