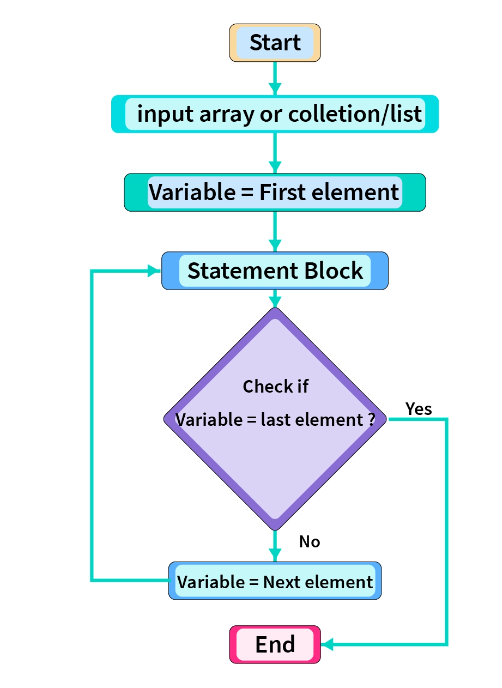
For Each Loop



Eg: Array\_1\_Dimensional\_Eg2

// this type of array declaration is used much.

Eg: Array\_2\_Dimensional\_jagged\_Eg3

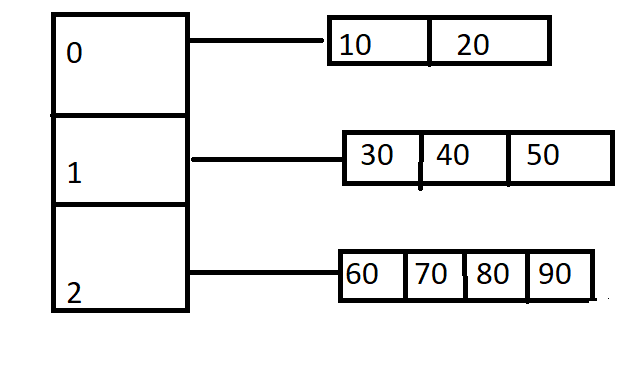
// go through the code



2d array can be written as row column

int[][] arr = {{ 10,20 },{ 30,40,50 },{ 60,70,80,90 }}

0 1 2



In for each loop for a 2-dimensional jagged array

for(int a[] : arr){

for(int b :a){

System.out.print(b)

}}

Here outer for each loop stores the elements in the row eg : 0th consists of elements 10,20 they are stored in outer for each loop array ( a[] )

And inner for each loop fetches each element in the a[] ,stores them in the variable b and further do operations based on the code .

After all the elements in 0th row gets printed the inner for each loop terminates and again the outer for each loop gets executed stores 1sth row elements 30,40,50 in a[] , and inner for each loop fetches each element and prints them one by one, after all the elements oin the 1st row executed inner loop terminates and again the same process continues for 2th row.

Eg: Array\_3\_Dimensional\_Jagged\_Eg3

Before going through the code go through this example

int[][][] arr = { { { 1,2 },{ 3,4 } } , { { 5,6 },{ 7,8 } } }

arr[0][0][0] = 1

arr[0][0][1] = 2

arr[0][1][0] = 3

arr[0][1][1] = 4

arr[1][0][0] = 5

arr[1][0][1] = 6

arr[1][1][0] = 7

arr[1][1][1] = 8

for(int[][] a : arr){

for(int[]b : a){

for(int c :b){

System.out.print(c)

}

}

}

Step-1 : Here in 1st outer for each loop , array int[][] a stores { {1,2}, {3,4} }

Step-2 : In 2nd outer for each loop , array int[] b stores {1,2}

Step-3 : In 3rd inner for each loop elements are stored in c and are printed one by one . (i.e first one 1 is stored and printed and then 2 is stored and printed)

Step-4 : After that control again comes to 2nd outer loop this time {3,4} is stored in array int[]b , and then

Step-5 : Now 3rd inner for each loop takes elements 3,4 one by one stores them and prints them(i.e first 3 is stored and printed and then 4 is stored and printed)

Step-6 : now control comes to 1st outer loop and this time elements { {5,6} , {7,8} }

Repeat the steps from 1 to 5.

Limitations of for each loop :

1. We do not have access of index

Eg: if we want to store the element only in the particular index it is not possible (customization is not possible)

1. It will traverse / iterate only in forward direction

Eg: if i want to traverse array in reverse direction or traverse from particular element to particular element it is not possible (iteration is possible only from array starting element to ending element )

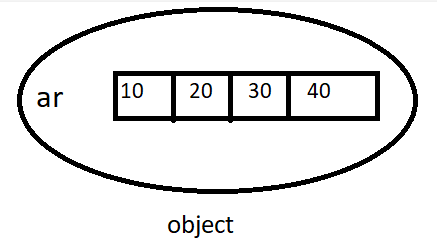
Different types of array declarations

Eg: Valid\_Array\_Declarations

// go through the code

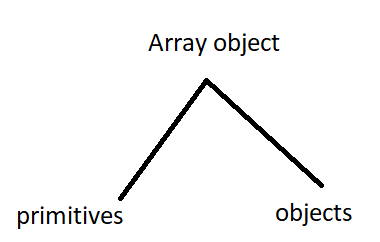
Note : Whenever you want to store similar type of collection and know the size , that to primitive values, in that case we can use array because array is fast and efficient than collections.

* Array is treated as object in java



Whole array is treated as object, and we can have primitives in that array .

* Array object can store both primitives and objects



* We can declare size in byte, short, int, char as size

Eg: byte b = 10

int[] arr = new int [b]

int to char is implicit type casting

int(4 bytes) <- char(2 bytes)

int[] arr = new int[‘a’] // c is converted to ascii value (97)

We cannot declare array size declaration in the of long, float, double, boolean types.

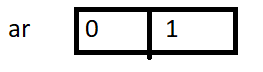
Exception in arrays

1. Exception is an error that occurs art the time of runtime .

Eg: int[] arr = new int[2]

arr[2] = 10

this leads to ArrayIndexOutOfBounds exception ,because the size given is 2 and we are trying to access method 3rd location



1. Size cannot be given as negative number it leads to NegativeArraySizeException at runtime

Eg: int[] arr = new int[-2]

It is not compile time error ,because the negative number may be of type byte , short , int , char

1. If you are giving array input and it is of other type (you have declared ) it leads to ArrayStoreException

Eg: int[] arr = new int[2]

If we give string type data to store in the array it leads to the array ArrayStoreException because we have declared int type array and given String type data , so runtime exception.

1. Array size should not exceed the datatype , if it happens it leads to OutOfMemory runtime error.

byte b = 180

Eg: int[] arr = new int[b]

Leads to runtime error , since byte range is only 128

if there is no class we cannot create object

Eg: Suppose if there is a class sample and I have created an object which is not of sample class it leads to compile time error .

class Sample{

------

-----

public static void main(String args[]){

Demo obj = new Demo(); // compile time error

}

}

If there is no class we cannot create object for that class.

Similarly Every type of array will have specific class , and that class is not for programmer , it is only for java ,so we are able to create object for array .

Eg: Array\_Class

// go through the code

To perform operations on array collection (data on arrays like {10,20,30} )we cannot use arrays class directly (type of array class) , so there is a utility class (or) support class called “Arrays”

In arrays there is a utility class “Arrays” . This class can be used along with array ,to perform some operations like sort, fill etc

All the methods which are inside “Arrays” class are static methods .

You can call static methods with object or with ClassName.methodname

Eg: Array\_Sum\_Of\_Elements

// go through the code

Variable written inside loops are treated as local variables .

Eg: Array\_Max\_Element

// go through the code.

To use the variable in the local method / loops first initialize them , other wise it leads to compile time error

Eg: Array\_Min\_Element

//go through the code

Eg: Copy\_1\_Array\_To\_Another

// go through the code

Eg: Array\_Eg1

// go through the code

Eg : Array\_Eg2

// go through the code

Eg: Array\_Eg3

// go through the code

Eg: Array\_Eg4

//go through the code