22/11/22 (15) Array In Java (Part-3)

↳ Hyder Abbas Sir

(1) For Each loop or Enhanced for loop

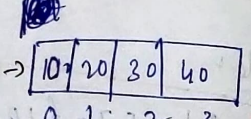
```
for(datatype Variable: arrayname){

    s.o.p(Variable);

}
```

Eg:

```
int a[] = {10,20,30,40};
for(int Elem: a){
    s.o.p(Elem);
}
```

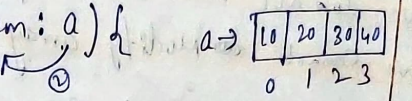


→ Traversing / iteration

① Traversing :- to traverse an array means to access Each Element (item) stored in the array so that the data can be checked or used as part of a process.

② iteration :- iterating over an array means accessing Each Element of an array one by one.

```
int a[] = {10,20,30,40};
for(int Elem: a){
    s.o.p(Elem);
}
```



(2) 2D Array (for Each loop)

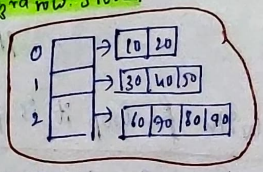
↳ Jagged array.

```
int[][] a = {{10,20}, {30,40,50}, {60,70,80,90}};
for(int ar[]: a) // getting 3 rows
{
    for(int Elem: ar)
    {
        s.o.p(Elem);
    }
}
```

① 1st row, 2nd row, 3rd row stored.

② 10,20

③ Elements in row are given to Elem one by one.



① Stores all rows.
② traverse through array and store values in array into Variables.
③ Print Values stored in Variable.

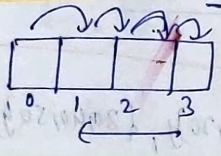
② Limitations of For Each loop :-

① We do not have access of Index. So we cannot modify array.

Eg:- Adding of value to some Index can't be done.

② It will Traverse (or Iterate only in forward direction.

Eg:-



→ forward (possible) iteration

Reverse (not possible) iteration

→ With for loop it is possible to Reverse iteration

```
Eg:- int [] ar = {10, 20, 30, 40};   (4-1 → 0)
                  0   1   2   3
for (int i = ar.length - 1; i >= 0; i--) {
    S.o.p(ar[i]);
}
```

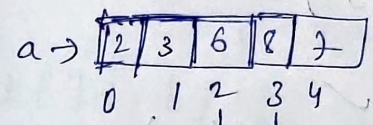
o/p :- 40, 30, 20, 10.

→ for Each loop Iterate from start to End.

③ Not Possible to get Specific Values in for Each Loop.

→ we can specify index directly and get values of it by for-loop.

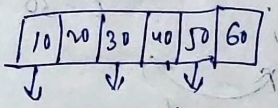
→ index control we don't have in for Each loop.



a[2] a[3] → for loop it is possible

→ not possible for for Each loop.

④ Printing alternate values in array :-



Printing alternative values

```
int [] ar = {10, 20, 30, 40};   X
for (int i = 0; i < ar.length; i++) {
    S.o.p(ar[i] + " ");
    ++i;
}
```

o/p :- 10, 30, 50

② Valid and invalid Syntax

① `int[][] a;` ✓ → good approach.

② `int a[][];` ✓

③ `int[] a[];` ✓

④ `int[] a[];` ✓

⑤ `int[] ar;` ✓

⑥ `int[5] ar;` ✗ (not valid.)

⑦ `int[] a, b;`

`a = new int[4];`

`b = new int[5];`

similar.

`int a, b, c;`

`a = 10`

`b = 20`

`c = 30`

⑧ `int[] x[], y;`

→ y is 1D array.

→ x is 2D array.

⑨ `int[][] x[], y;`

→ y is 2D array.

→ x is 3D array.

→ b is 1D array.

⑩ `int[][] a, b[];`

→ a is 2D array

⑪ `int[][] a, b;`

a, b are 2D arrays.

⑫ `int[] a, []b;` ✗ not allowed. (C.E)

→ After b, we cannot write [].

⑬ `int[] a, b[][];`

→ 1D array.

a → 1D array.

b → 3D array.

→ array → Sure about size
↳ Homogenous data.
↳ store Primitive Values

} go for array it is fast & Efficient than collection.

⇒ array is treated as object

`int[] ar = new int[4];`

ar → | 10 | 20 | 30 | 40 |

`a[0] = 10;`

`a[1] = 20;`

`a[2] = 30;`

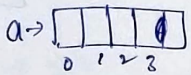
⇒

① arrays are treated as objects. primitive are present in it.

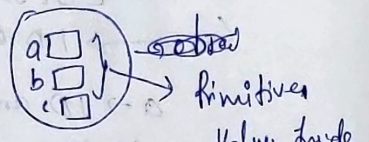
23/11/22

```
int [] a = new int[5];
```
↳ Primitive

a[10] = 10




```
Class Demo
{
int a;
int b;
int c;
}
```



② In array we can store objects and primitive

③ In Collections we can only store objects.

④ `int[] a = new int[5];` (5 → size)

↳ Size can be int, byte, short type but Cannot be long

↳ `int[] a = new int['a'];` ✓ allowed.
↳ 97 ASCII value

`int[] a = new int['A']`
↳ 65 (Size

⑤ Size of array can be byte, short, char
Size of array Cannot be float, double, String.

→ `int[] a = new int[10.5];`
↳ Invalid.

⑥ Exceptions associated with Array :-

① Array Index Out Of Bound :-

Eg: `int[] a = new int[2];` a → 

`a[2] = 10` → run time Error
↳ it throws exception
↳ ArrayIndex out of Bound.

② Negative Array Size Exception :-

`int[] a = new int[-2];`
↳ Size Cannot be nagative
↳ it throws NegativeArraySize Exception

→ Java Compiler will not any Error when we write like this
→ it only checks Syntax of program.
→ JVM throws Exception during run time.

③ Array store Exception :-

→ Arrays in Java is Homogenous

```
int[] a = new int[2];
a[0] = 10;
a[1] = "Ashish";
```

↳ String.

↳ it throws Array store Exception

④ OutOfMemoryError Exception :-

→ `int[] a = new int[999999];`

→ this Error is thrown when there is insufficient space to allocated an object in the JAVA heap.

`int[] a = new int[99999999];`

↳ it throws

↳ out of Memory Error Exception.

↳ in Runtime.

① if Class is not there we cannot create object.

② Whatever Classes associated with array, in java is there for Java internal purpose and not available for programmers.

③ Every type of array will have Specific Class. those classes for Java language but not for programmers.

④ these classe are present internally.

Class ← `int[] a = new int[];`

`Char[] a = new char['a'];`

⑤ In case of Scanner Class they are inbuilt and they also available for programmers.

```
Scanner S = new Scanner(System.in);
S.nextInt();
```

↳ Methods in Class

Code :-

Get class name of Array :-

⇒ `int[] a = new int[4];`

`S.o.p(a.getClass().getName());`

→ o/p :- @.[I → class name of
↳ 1D Integer array

[[I → for 2D Integer array.

[C → class for 1D char array.

→ Every Array type belongs to Spicific class but that class is not for programmers.

⑥ They have provided utility class by name Arrays.

↳ these class used along array

⑤ Arrays Class :- Created because We do not have class to perform operation easily.

↳ there class used along array.

↳ to Perform operations on collections easily.

↳ operations like sort, fill, binary search

↳ we have so many Methods in arrays class to perform operations. like sort, fill, binary search

→ all the Methods inside array's class are Static Methods.

→ the Methods are Called directly by class name.

Eg:①

```
Class Demo
{
    Void display()
    {
        ≡
    }
}
```

⇒ Creating Object

`Demo d = new Demo();`

`d.display();`
↳ Calling Method

⇒ if you use Static then Object Creation is optional.

```
CClass Demo
{
    Static Void display()
    {
        ≡
    }
}
```

→ `Demo.display();`

↳ to invoke Method no need to Creat object

↳ to invoke Method we can call directly Class name.

→ ~~you call method~~

→ you can call Method by Creating object (or) without Creating it. it is optional.

→ we can directly call Method by class name

Eg: `Demo.display();`

Conclusion :-

① Every type of array has specific class. there cannot used by programmer but it is for internal purpose.

② Arrays class is used to perform operation on arrays.
↳ we can use inbuilt Methods
↳ all Methods are static.
↳ Methods are invoked directly by class name.

Eg ① **to sort an array**

```
int a[] = {10, 60, 50, 30};

Array.sort(a); → it goes through array and it sorts it.
```

Eg ② **Adding values**

```
int a[] = new int[4];
```

a → | 0 | 0 | 0 | 0 | (default value)
0 1 2 3

→ If all 4 places we need same values.

alter-①

```
for (int i=0; i<a.length; i++)
{
    a[i]=5;
}
```

alter② **using utility class**

```
Array.fill(a, 5); → it fill value 5 in all indexs.
```

a → | 5 | 5 | 5 | 5 |
0 1 2 3

① write program to add elements in an array?

```
int[] a={10,20,30};
int sum=0;
for(int i=0; i<a.length; i++)
{
    sum=sum+a[i]; (or) sum+=a[i];
}
S.o.p(sum);
}
```

② find max and min element in an array?