ARRAYS

1. Introduction
2. Array Declaration
3. Array Creation
4. Array Initialization
5. Array Declaration, creation & initialization in a single line.
6. length vs length()
7. Anonymous Arrays
8. Array element assignments
9. Array variable assignments

Date ___/___/_____ Page No.: _____

An array is an indexed collection of fixed number of homogenous data elements.

→ The main advantage of an array is we can represent a huge number of values by using single variable so that the readibility of code will be improved.

→ But the main disadvantages of an array is —

* fixed in size i.e once we create an array there is no chance of ↑ or ↓ based on our requirement.

Hence to use array concept compulsary we should know the size in advantage, which may not possible always.

Array declaration

⊕ 1- Dimensional Array Declaration

Best one ← int [] x;
int []x;
int x[];
] All are valid

Date ___/___/_____ Page No.: _____

int [] x;
recommended bcoz name is clearly separated from type.

3

At the time of declaration we can't specify the size otherwise we will get compile time Error.

int [6] x; (X)

int [] x; (✓)

2- Dimensional Array Declaration

int [][] x;
int [][] x;
int x[][];
int [] [] x;
int [] x[];
int [] x[];

All are valid.

Which of the following are valid ?

✓ int [] a, b; a → 1 , b → 1

✓ int [] a[], b; a → 2 , b → 1

✓ int [] a[], b[]; a → 2 , b → 2

✓ int [] [] a, b; a → 2 , b → 2

✓ int [] [] a, b[]; a → 2 , b → 3

✗ int [] [] a, []b; → C.E

Date ___/___/_____ Page No.: _____

NOTE → If we want to specify dimension before the variable, that facility is applicable only for 1st variable in a declaration.

If we are trying to apply for remaining variable we will get compile time error.

eg → int [] [] a, [] b, [] c,
(b: ✗, c: ✗)

✓

3- Dimensional array declarations

```
int [][][]   a;
int    [][][] a;
int    a[][][];
int []    [][]a;
int []    a[][];
int [][]    a[];
int [][]   []a;
int    [][]a[];
int    []a[][];
```

All are valid.

Date ___/___/_____ Page No.: _____

Array Creation

Every array in ~~arr~~ JAVA is object only hence we can create arrays using new operator.

eg → int [] a = new int [3];
System.out.println(a.getClass().getName()); // [I

for every array type corresponding class are available & these classes are part of JAVA language & not available to the programmer level.

Array type	corresponding class name
int []	[I
int [][]	[[I
double []	[D
short []	[S
Byte []	[B
Boolean []	[Z

Related to array there are several loop holes -

(1) At the time of array creation, compulsary we should specify the size otherwise we will get compile time error.

eg → int [] x = new int [];(X)
int [] x = new int [6];(✓)

Date ___/___/_____ Page No.: _____

(2) It is legal to have an array with size 0 in JAVA.

eg → int [] x = new int [0];

(3) int [] x = new int [-3];
Runtime error : -ve array size exception.

If we are trying to specify array size with -ve int value, then we will get Runtime exception saying [-ve Array size exception]

(4) To specify array size the allowed datatypes are byte, short, char & int.

If we are trying to specify any other types, then we will get compile time error.

eg → ✓ int [] x = new int [10];
✓ int [] x = new int ['a'];
byte b = 20;
✓ int [] x = new int [b];
short s = 30;
✓ int [] x = new int [s];
(x) int [] x = new int [10L];

Date 3/12/24 Page No.:

(5) The maximum allowed array size in JAVA is 2147483647 which is the maximum value of int datatype.

eg → (✓) int [] x = new int [2147483647];

(✗) int [] x = new int [2147483648];

↓

C.E : integer too large

(#) Even in the 1st case, we may get Runtime exception if sufficient heap memory not available

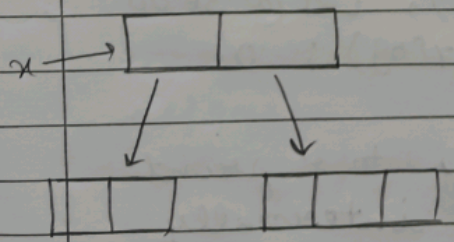
03/12/24

(#) Two dimentional Array (creation)

In JAVA, two dimensional array not implemented by using matrix representation, some people followed array of arrays approach for multi dimensional array creation.

The main advantage of this approach is memory ultization will be improved.

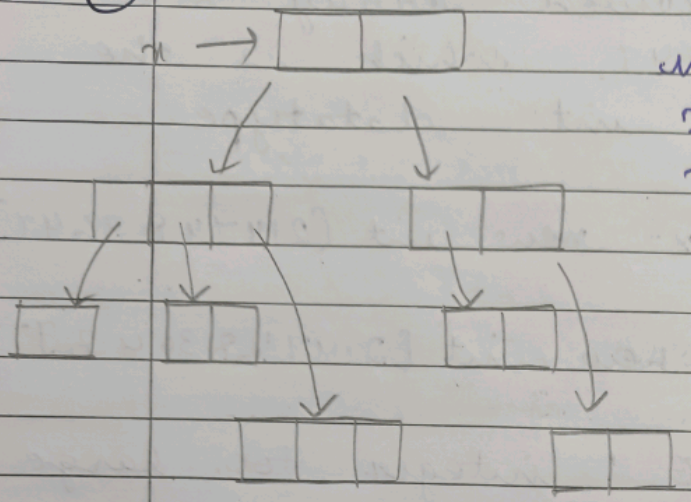
eg → (1) memory structure & Java code




```
int [][] x = new int [2][];
x[0] = new int [2];
x[1] = new int [3];
```

Date ___/___/_____ Page No.: _____

(2)



```
int[][][] x = new int[2][][];
x[0] = new int[3][];
x[0][0] = new int[1];
x[0][1] = new int[2];
x[0][2] = new int[3];
x[1] = new int[2][2];
```

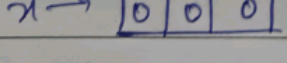
Which of the following are valid?

✗ `int[] a = new int[];`
✓ `int[] a = new int[3];`
✗ `int[][] a = new int[][];`
✓ `int[][] a = new int[3][4];`
✓ `int[][] a = new int[][4];`
✓ `int[][] a = new int[3][4];`
✓ `int[][][] a = new int[3][4][];`

Array Initialization

Once we creates an array, every initializer with default values.

eg →




```
int[] x = new int[3];
S.pln(x); [I@3e00...;
S.pln(x[0]); 0
```

Whenever we are trying to print any reference variable internally two string method will be called

Date ___/___/_____ Page No.: _____

which is implemented by default to return the string in the following form.

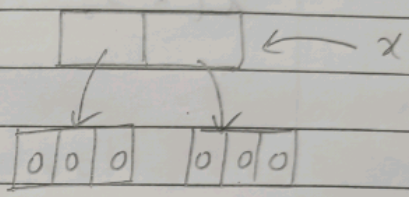
classname @ hashcode in hexadecimal form

eg → # int [][] x = new int [2][3];

S.pln (x); [[I@3e25

S.pln (x[0]); [I@4e45

S.pln (x[0][0]); 0



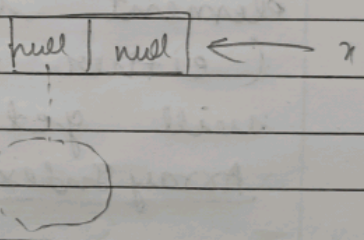
int [][] x = new int [2][];

S.pln (x); [[I@4e36

S.pln (x[0]); null

S.pln (x[0][0]);

R.E → NPE



NOTE

If we are trying to perform any operation on null then we will get Run time exception saying Null pointer exception.

→ Once we creates an array every array element by default initialized with default values if we are not satisfied with default values then we can override these values by our customized values.

Date ___/___/_____ Page No.: _____

eg → int [] x = new int [6];

x[0] = 10;

:

x [5] = 60;

x → | 10 (0) | 50 (8) | 20 (8) | 30 (8) | 40 (8) | 60 (8) |

x[6] = 70; RE: Array Index Out of Bounds Exception

x[-6] = 80; (" " " ")

x[2.5] = 90; [CE: Possible loss of precision
found : double
required : int]

NOTE

If we are trying to access array element with Out of Range Index (either +ve or -ve int value) then we will get Runtime exception saying Array Index Out of Bounds Exception.

Array declaration, creation and Initialization & in a single line

We can declare, create & initialize array in a single line (shortcut representation).

eg → int [] x;

int [] x = new int [3]; → int [*] x = {10, 20, 30};

x[0] = 10;

x[1] = 20;

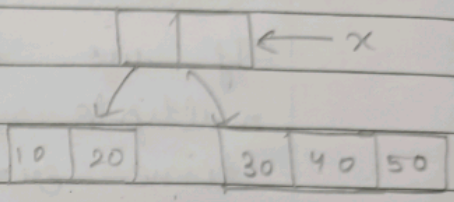
x[2] = 30;

char [] ch = {'a', 'b', 'c'};

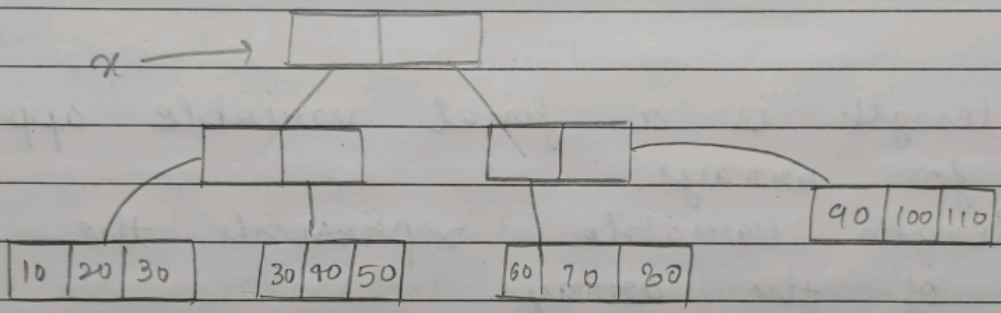
Date ___/___/_____ Page No.: _____

We can use this shortcut for multidimensional arrays also –

eg → `int [][] x = { {10,20}, {30,40,50}};`



eg → `int [][][] x = {{{10,20,30},{40,50,60}}, {{70,80}, {90,100,110}}};`



`S.pln (x[0][1][2]);` 60
`S.pln (x[1][0][1]);` 80
`S.pln (x[2][0][0]);` RE → AIOOBE
`S.pln (x[1][2][0]);` RE → AIOOBE
`S.pln (x[1][1][1]);` 100
`S.pln (x[2][1][0]);` RE → AIOOBE

`int x = 10;`
⇓
`int x;`
`x = 10;`

`int [] x = {10,20,30}`
`x = {10,20,30};`
↳ CE → Illegal start of Expression

Date ___/___/_____ Page No.: _____

If we want to this shortcut compulsory we should perform all activities in a single line.

If we are trying to divide into multiple lines we will get compile time error saying illegal start of Expression.

length vs length().

length is a final variable applicable for arrays.
length variable represents the size of the array.

eg → int [] x = new int [6];
S.pln (x.length()); → (C.E→ cannot find symbol
symbol → method length()
location : class int [])

S.pln (x.length); 6

length()

length() is a final method applicable for string objects.
length() returns no. of character present in the string.

eg → String s = "aarohi";
S.pln (s.length); (E→ cannot find symbol
Symbol → variable length
location → class

Date ___/___/_____ Page No.: _____

S.pln (x.length()); 5

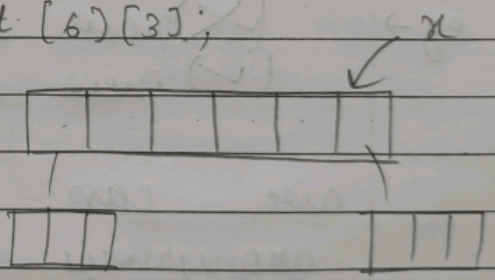
NOTE → length variable applicable for arrays but not for string objects whereas length() applicable for string objects but not for arrays.

eg → string [] s = {"A", "A,A", "A";"A";A" }

✓ S.pln (s.length); 3
✗ S.pln (s.length()); } compile time error
✗ S.pln (s[0].length);
✓ S.pln (s[0].length()); 1

In multidimensional arrays length variable represents only base size but not total size.

eg → int [][] x = new int [6][3];



S.pln (x.length); 6
S.pln (x[0].length); 3

There is no direct way to specify / find total length of multidimensional array but indirectly we can find as follows —

eg → x[0].length + x[1].length + x[2].length + ...

Date ___/___/_____ Page No.: _____

Anonymous Arrays

Sometimes, we can declare an array without name such type of nameless arrays are called anonymous arrays.

The main purpose of anonymous arrays is just for instance use (one time usage).

We can create anonymous arrays as follows

```
new int [] {10,20,30,40};
```

While creating anonymous arrays we can specify the size otherwise we will get compile time error.

eg→
(X) `new int [3] {10,20,30};`
(✓) `new int [] {10,20,30};`

We can create multidimensional anonymous arrays also.

eg→ `new int [][] {{10,20},{30,40,50}};`

→ Based on our requirement, we can give the name for anonymous array, then it is no longer anonymous.

eg→ `int [] x = new int [] {10,20,30};`

Date ___/___/_____ Page No.: _____

eg →

```
class Test
{
public static void main (String [] args)
{
   sum (new int [] {10,20,30});
}
  public static void sum (int [] x)
    {
       int total=0;
      for (int x1 : x)
        {
          total = total + x1;
        }
       Sopln ("The Sum:" + total);
    }
}
```

In above example, just to call some method we required an array but after completing some method call we are not using that array anymore.

Hence, for this one time requirement anonymous array is the best choice.

Date ___/___/_____ Page No.: _____

Array element Assignments

CASE 1

In the case of primitive type arrays as array elements we can provide any type which can be implicitly promoted to declare a type.

eg → #
```
int[] x = new int[5];
x[0] = 10;
x[1] = 'a';
byte b = 20
x[2] = b;
short s = 30;
x[3] = s;
(X) x[4] = 10l;   [ C.E→ PLP
                    found: long
                    required: int ]
```

→ In the case of float type arrays, the allowed data types are byte, short, char, int, long & float.

Date ___/___/_____ Page No.: _____

② Case 2

In case of object type arrays as array elements we can provide either declare type objects or its child class objects.

eg → ① Object [] a = new Object[10];

✓ a[0] = new Object();
✓ a[1] = new String("durga");
✓ a[2] = new Integer(10);

N ⟶ B, S, I, L, f, D

② Number [] n = new Number[10];

✓ n[0] = new Integer(10);
✓ n[1] = new double(20.5);
✗ n[2] = new String("nonu");
↓
C.E → incompatible types
found: Java.lang.String
required: java.lang.number

③ Case 3

In case of Interface type arrays as array elements its implementation class objects are allowed.

Date ___/___/_____ Page No.: _____

eg → Runnable [] r = new runnable [10];

Runnable (I)
Thread

✓ r[0] = new thread();
✗ r[0] = new String("AA");

↓

compile time : incompatible types error

Array Type	Allowed element Type
primitive arrays	Any type which can be implicitly promoted to declared type.
Object type arrays	either declared type or its child class objects.
Abstract class type arrays (number)	its child class objects.
Interface type arrays	its implementation class objects are allowed.

Date 4/12/24 Page No.:

Array variable assignments

Case 1 :-

Element level promotion are not applicable at array level.

eg → ch element ⟶ int type
whereas
Char [] cannot be promoted to int [].

```
int [] x = {10, 20, 30};
char [] ch = {'a', 'b', 'c'};

✓ int [] b = x;
X int [] c = ch; → CE: incompatible types
                    found: char []
                    required: int []
```

Que :- which of the promotions will be performed automatically?

char → int ✓
char [] → int [] X
int → double ✓
int [] → double [] X
float → int X
float [] → int [] X
String → object ✓
String [] → object [] ✓

Date ___/___/_____ Page No.: _____

But in the case of object type arrays child class type array can be promoted to parent class type array.

eg → string [] s = {"A", "B", "C"}

object [] a = s;

Case 2 :

Whenever we are assigning one array to another array internal elements won't be copied, just reference variables will be reassigned.

eg → int a = {10, 20, 30, 40}
int b = {50, 60}

① ✓ a = b
② ✓ b = a
} Both are valid

Case 3 : Whenever we are assigning one array to another array ~~and~~ the dimensions must be matched.

for example → in the place of 1-dimensional int [] we should provide 1-dimensional array only, if we are trying to provide any other dimension, then we will get compile time error.

Date ___/___/_____

int [][] a = new int [3][];

✗ a[0] = new int [4][3];
CE: incompatible types
found: int [][]
required: int []

✗ a[0] = 10;
CE: incomptabile types
found: int
required: int []

✓ a[0] = new int [2];

NOTE:
Whenever we are assigning one array to another array both dimensions & type must be matched but sizes are not required to match.

Examples:

① class Test
{
public static void main (String [] args)
{
for (int i=0; i<= args.length; i++)
{
SopIn (args[i]);
}
}
}

Date ___/___/_____ Page No.: _____

Java Test A B C ↵

A
B
C

for (int i=0; i<=3 ; i++)

RE: AIOOBE

args[0]
args[1]
args[2]
args[3] → (X)

JAVA Test A B ↵

A
B

RE: AIOOBE

JAVA Test ↵

RE: AIOOBE

(2)

```
class Test
{
    P S V main (String [] args)
    {
String[] argh = {"X","Y","Z"}

        args = argh;

    for( String s: args)
        {
            Sopln (s);
        }
    }
}
```

Java Test A B C ↵

X
Y
Z

Date ___/___/_____ Page No.: _____

Java Test A B

X
Y
Z

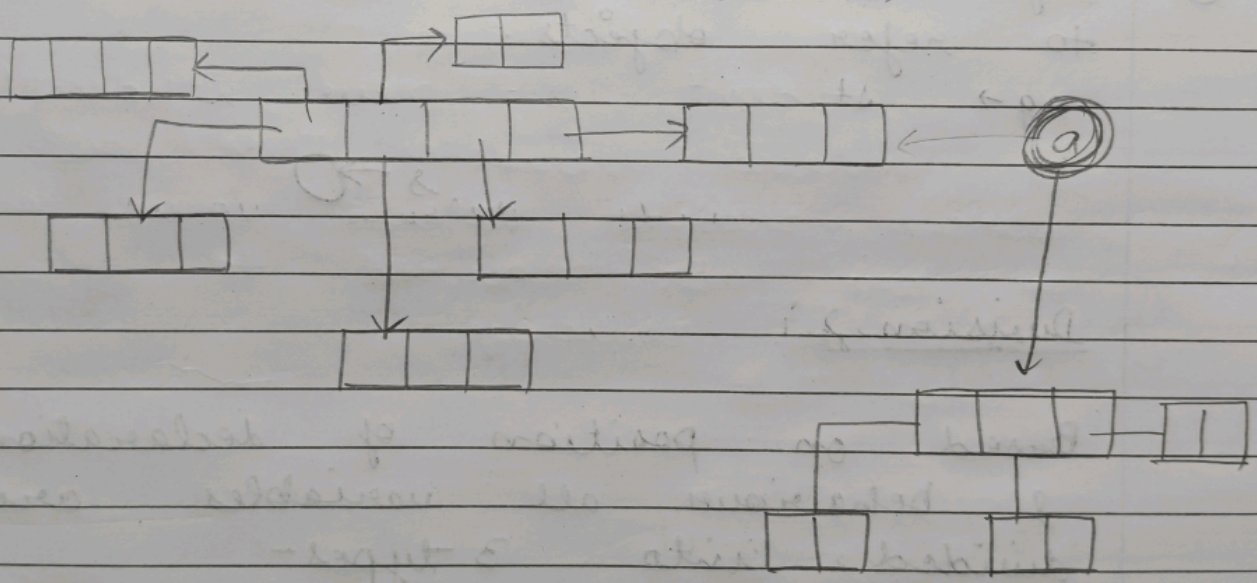
args : | A | B | C |

Java Test

X
Y
Z

args → | X | Y | Z |

③ int [][] a = new int [4][3]; → 5
a[0] = new int [4]; → 1
a[1] = new int [2]; → 1
a = new int [3][2]; → 4



① Total how many objects are created?
11.

② How many objects are eligible for garbage collection?
7.