Arrays

Date

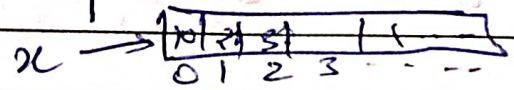
1. introduction
2. Array declaration
3. Array Creation
4. Array initialisation
5. Array declaration, creation & intialisation in a single line.
6. length vs length()
7. Anonymous Array
8. Array element assignments
9. Array Variable assignments

Introduction

```
int x=10;
int y=20;
int z=30;
```

> Not posible

```
int[] x = new
          int[10000]
```



Array is an Indexed collection of fixed no. of Homogeneous data element

the main advantage:

We can represent huge number of values by using single variable. so that readibility improved.

disadvantage

No chance of decreasing or increasing the size, hence to use array concept compulsory we should know size in advance

int [] [] a, b; — b >1

Date

Which may not possible always.

One dimensional array declaration

int [] x;
int [] x;
int x[];
} Valid

Recommended Name is Clearly Seperated from Type

At the time of declaration we can't specify the size.
other will compiler give error

* int [6] x;
//invalid

int [] x; //valid;

Two dimensional array

int [][] x;
int [][] x;
int x[][];
} valid.

int [] [] x;
int [] x[];
int [] x[];
} valid.

Date

```
int [] a,b;          a → 1
                     b → 1   } both dimension is 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;   CE
```

Note:
if we add array before variable after first variable then compilation error comes.

eg: `int[] [] a, [] b, [] c;`

~~Three~~
Three dimensional array declaration.

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

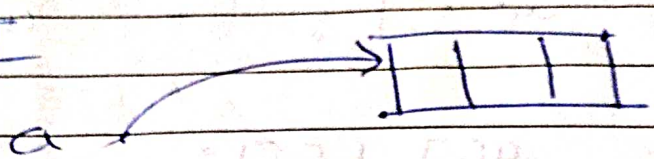
Spiral

Array Creation

Date

```
int [] a = new int [3];
Sop(a.getClass().getName())
```

Output [I



Every array in java is an object only Hence we can create array by using 'New' operator

For array Type Corresponding classes are available and these classes are part of Java language and not available to programmer level.

Array Type	Corresponding class name
int []	→ [I
int [] []	→ [[I
double []	[D
short []	[S
byte []	[B
boolean []	[Z

Loopholes of array

Date

① `int[] x = new int[];` → C.E (compile time error)

JVM will not reserve memory

At the time of Creation.

* At the time of Array Creation, compulsory we should specify the size, otherwise we will get compile time error.

② An array with zero size is perfectly available.

```
int[] x = new int[0];
```

best example public static void main (String[] args)

we are not

```
public static void main (String[] args)
{
    System.out.println (args.length);
}
```

output 0

③ `int x = new int[-3]`

→ At compile time → No error

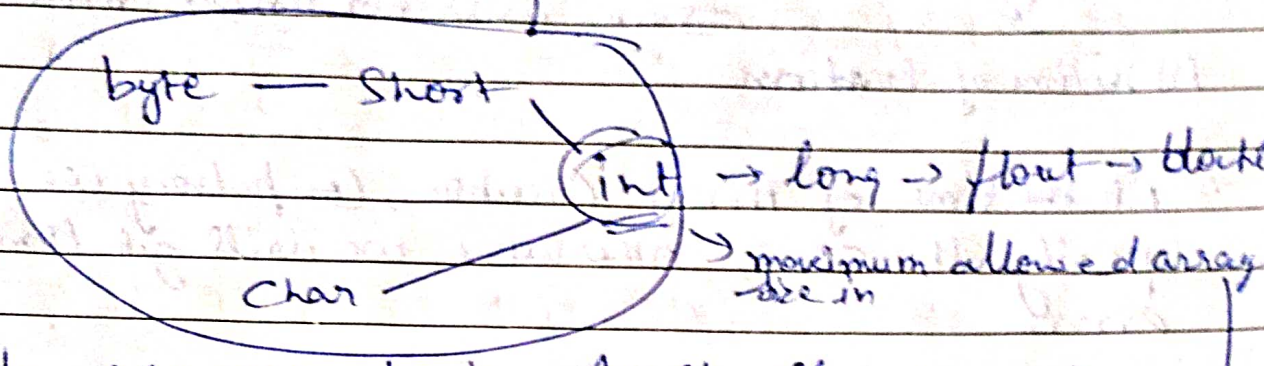
→ Run time → Exception

Java.lang. NegativeArraySizeException : -3

if we are trying to specify with negative size value

Date

4. To specify array size the allowed data types are:



if we are trying to specify any other type then we will get compile time error.

✓ 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];
✗ int [] x = new int [10l];
Compile error

int [] x = new int [2147483647];

My system's JVM is not much size to execute. Runtime error exception.

[2147483648]
↳ compile error

Program:- To Convert the characters into Unicode Value:

Date

```
public class char_Unicode: {
  public static void main (String[] arg) {
    int x[] = new int ['v'];
    int CountZeros = 0;
  for (int i=0 ; i< x.length ; i++) {
    if (x[i]==0) {
        Countzeros ++
    }
  }
  System.out.println ("number of zeroes in the
  array: "+ Countzeros);
  }
}
```

2 dimensional array Creation

Date

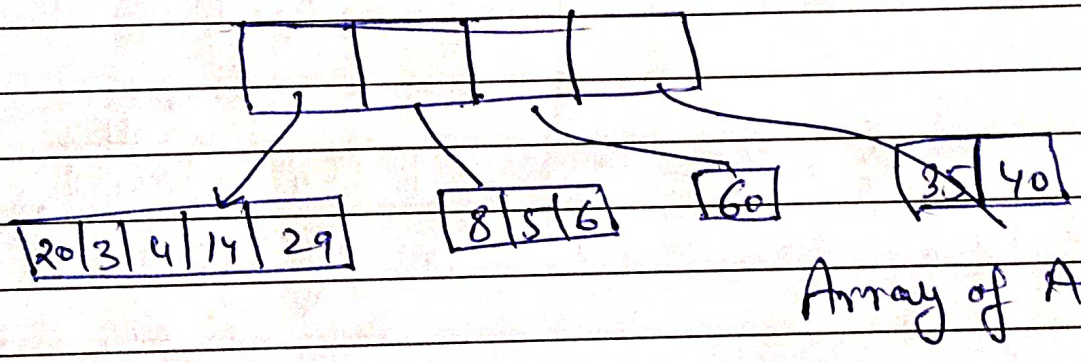
In java 2 dimensional array not implemented by using matrix style, some people followed "Array of Arrays" approach for Multidimensional array Creation.

⇒ Advantage of this Approach is Memory Utilisation will be improved.

eg

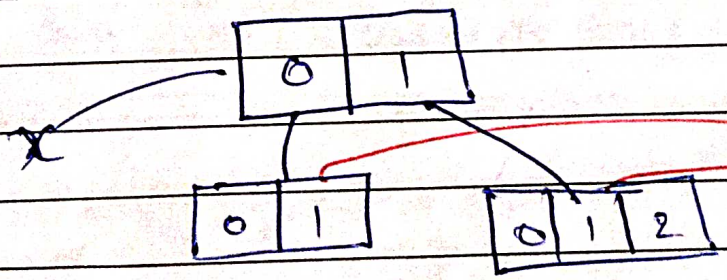
S_1	20	3	4	14	29
S_2	8	5	6	x	x
S_3	60	x	x	x	x
S_4	35	40	x	x	x

matrix style



Array of Array

Example 1.



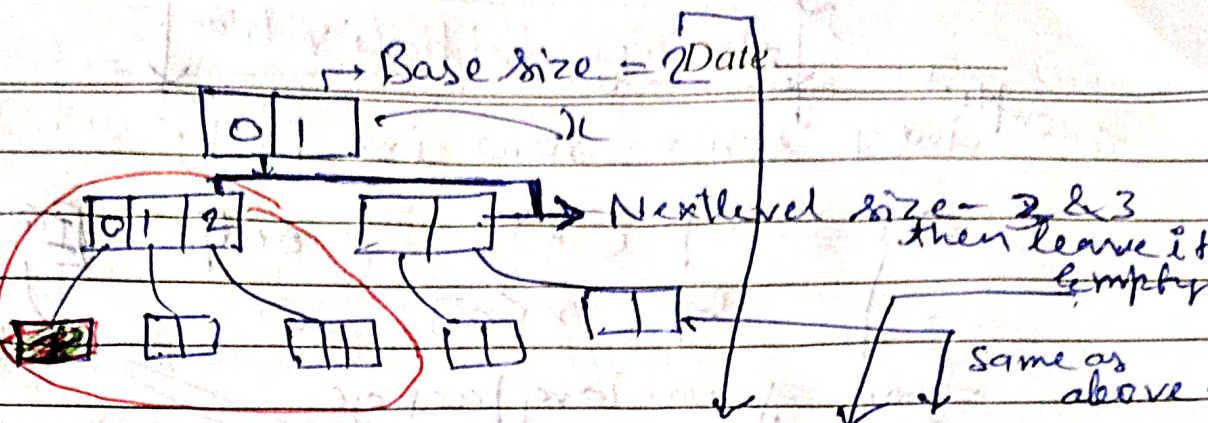

```
int[][] x = new int[2][];
```

leave it because next level array is not fixed

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

Sometimes 2 or 3

Example-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];
```

Q. Which of the following Array declarations are Valid?

```
int[] a = new int [];      invalid (base size we have to specified)
int[] a = new int [3];     valid
int [][] a = new int [][];  invalid
int [][] a = new int [3][];  valid
int [][] a = new int [][4];  invalid
int [][] a = new int [3][4];  valid
int [][][] = new int [3][4][5];  valid
int [][][] = new int [3][4][];  valid
int [][][] = new int [3][][5]  invalid
int [][][] = new int [][4][5]  invalid
```

Array initialisation

Date

Example 1

```
int [] x = new int [3];
sop(x);   // output [I@8087c
sop(x[0]); 0
```

from java.lang package.

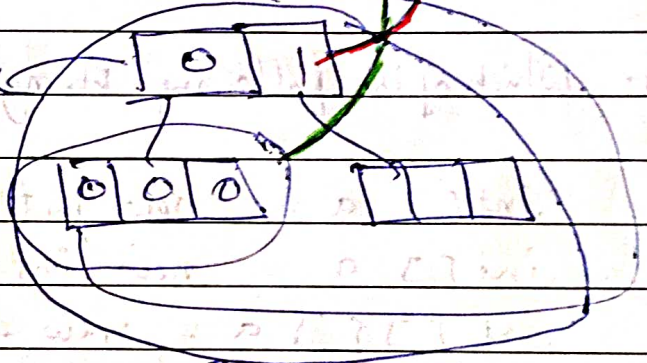
class name @ hash code in hexa decimal form

Example 2

```
int [][] x = new int [2][3];
sop(x);      [[I@3ae25
sop(x[0]);   [I@19211f
sop(x[0][0]); 0
```

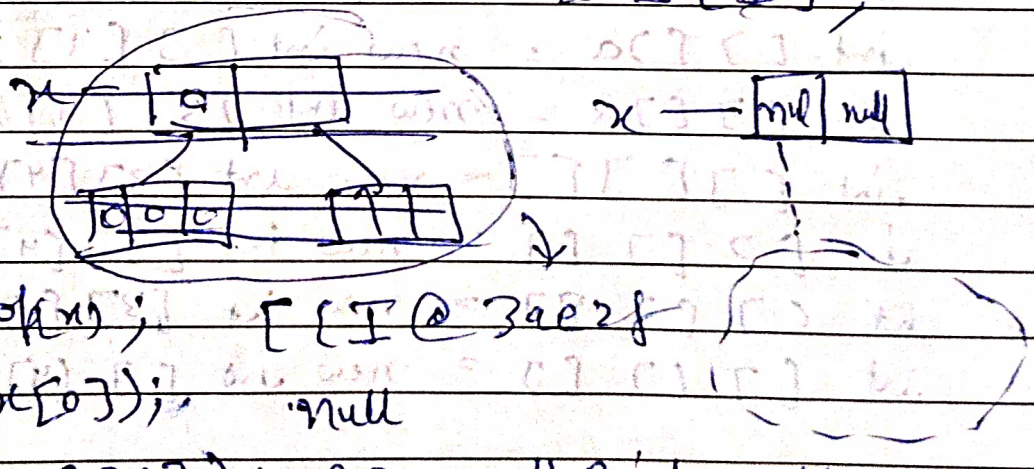
Output

x is reference variable to 2 dimension array



Example 3

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




```
sop(x);      [[I@3ae2f
sop(x[0]);   null
sop(x[0][0]); R.E - null pointer exception
```

(because null of null)

Note: if we are trying to perform any operation on null then we will get R.E

Array initialisation

Date

Example 1

```
int [] x = new int [3];
sop(x);   // output [I@8oefc
sop(x[0]); 0
```

from java.lang package.

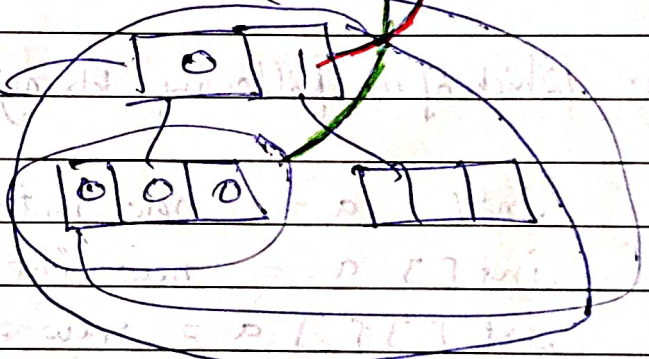
classname @ hashcode in hexadecimal form

Example 2

```
int [][] x = new int [2][3];
sop(x);      [[I@3ae25
sop(x[0]);   [I@19213f
sop(x[0][0]); 0
```

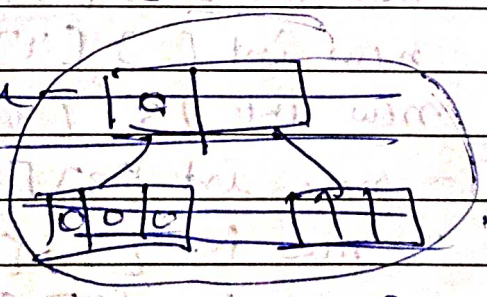
Output

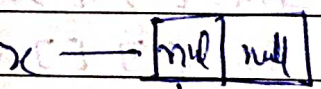
x is reference variable to 2 dimension array



Example 3

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






```
sop(x);      [[I@3ae2f
sop(x[0]);   null
sop(x[0][0]); R.E - null pointer exception
```

(because null of null)

Note: if we are trying to perform any operation on null then we will get R.E

```
int [] x = new int [6]
x[0] = 10;
x[1] = 20;
x[2] = 30
x[3] = 40;
x[4] = 50;
x[5] = 60;
```

6 - [0 to 5] + 6

0 1 2 3 4 5

x → | 10 | 20 | 30 | 40 | 50 | 60 |

Date

once we create an array every array element by default initialised with default value.

if we are not satisfied with default value then we can override these value with our customised values

if x[6] = 70; // {immediately Run time exception, array out of bound exception}

if x[-6] = 80; R.E Array Index Out of Bound Exception

if x[2.5] = 90; CE Possible found of precision

(Compiler will check if it is valid int value)

found : double

Required : int

Note: if we are trying to access Array element with out of Range Index (either positive value or negative int value) then we will Runtime exception saying Array Index Out of bound exception

Array declaration, Creation and Initialisation in a single line

```
int [] x;
x = new int [3];
x[0] = 10;
x[1] = 20;
x[2] = 30;
```

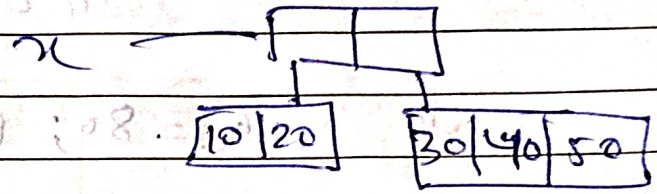
```
int [] x = {10, 20, 30};
char [] ch = {'a', 'e', 'i', 'o', 'u'};
String [] s = {"A", "AA", "AAA"};
```

Date

We can use this shortcut for multi-dimensional array also

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

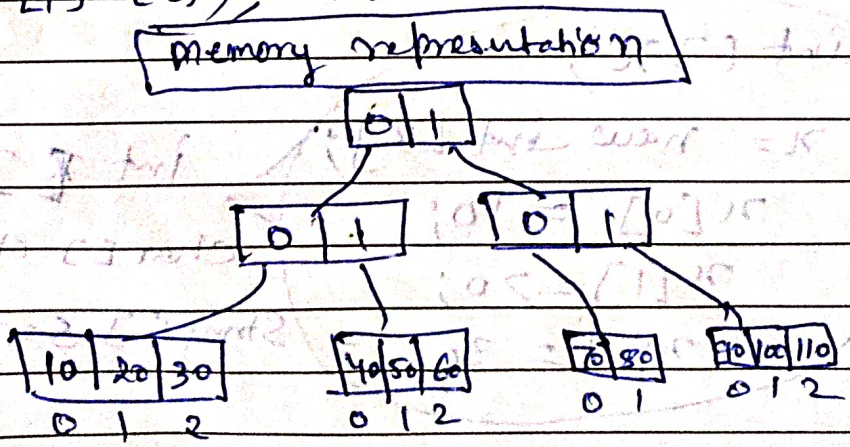
Memory representation



3 dimensional short cut

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

```
Sop(x[0][1][2]);  60
Sop(x[1][0][1]);  80
Sop(x[2][0][0]);  R.E
Sop(x[1][2][0]);  R.E
Sop(x[1][1][1]);  100
Sop(x[2][1][0]);  RE
```



int x=10;

valid
int x;
x=10;

int [ ] x = {10, 20, 30};

Date

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

if we want to use this shortcut Compulsory we should perform all activities in a single line if we are trying to divide into multiple line then we will compile time error.

length vs length()

length is a final variable applicable for Arrays.

length variable represent the size of the Array

ex: int [ ] x = new int [6];

Sop (x.length()); C.E Cannot find Symbol
Symbol : method length()
location : class int [ ]

Sop (x.length); output 6

ex: String s = "durga";

Sop (s.length); C.E Cannot find Symbol
Symbol : variable length
location : class java.lang.String

Sop (s.length()); 5 output

Date

length ( ) method is a final method applicable for String objects. it returns number of character present in the string

~~length~~ final class String

when final keyword is used its child cannot extend parent
∴ Not possible to create child because of final class, then override not possible

{
length ( )
{
}

String – length( )
~~String~~
Array[ ] – length

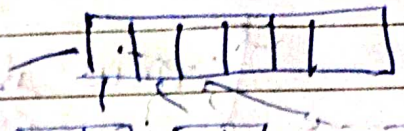
Note: length variable applicable for arrays but not for string objects whereas length ( ) method applicable for String object but not for Arrays.

```
         → Array
String [ ] s = {"A", "AA", "AAA"}

Sop (s.length);        3 output
Sop (s.length());      CE: Cannot find symbol
                           symbol: method length()
                           location class String[ ]
Sop (s[0].length);     CE → cannot find symbol
                           symbol. variable length
                           location: class java.lang.String
Sop (s[0].length());
                       → 1 output
```

and [][] x = new int [6][3];

Sop (x.length) ; 6

Sop (x[0].length); 3



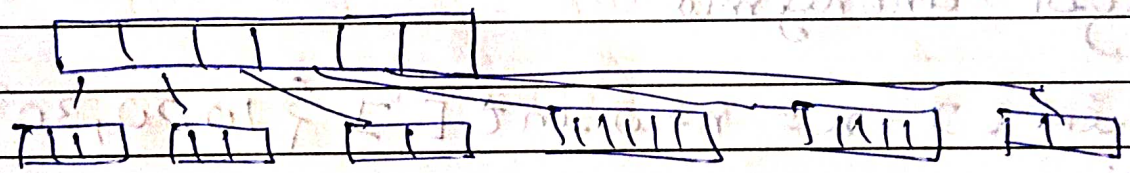
{Equal}

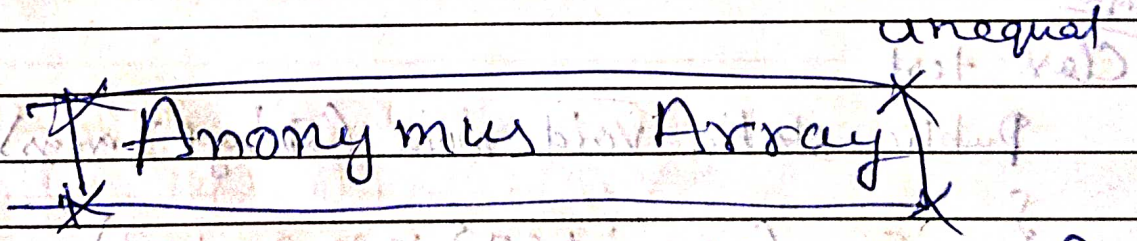
in Multi-dimensional array length Variable represent only base size

There is no direct way to ~~specify~~ find total length of Multi dimension Array but indirectly we can find as follows:

x[0].length + x[1].length + x[2].length - - -

memory representation





An array without name is Anonymous Array

* ~~just for instant use~~
* or One time use

} We declare array without name

name less array are called

We can create anonymus array as follows:

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

Common mistake:

```
new int[3] {10,20,30}
```

✗ C.E

because while creating anonymus array can't specify the size, because we are already providing no. of elements

Multi dimensional array (Anonymus) Creation:

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

based on our requirement, we can give the name for anonymus then it is no longer anonymus

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

Example:

```
Class test
{
    Public static void main (String[] args)
    {
        Sum (new int[] {10,20,30,40});
    }
    P s v Sum (int[] x)
    {
        int total = 0;
        for (int x1 : x)
        {
            total = total + x1;
        }
        Sop ln ("The Sum:" + total);
    }
}
```

Spiral

in the Above example just to call some method() we required an array but after completing some method() call, we are not using that array any more Hence, for One time requirement anonymous array is the best choice

Array Element assignments

Case:1 In the case of primitive type array as array Element, we can provide any type which can be implicitly promoted declared type.

Example 1. for the int type array the allowed array Element types are byte, short, Char, int.

```
int[] a = new int[10];
a[0] = 97;          Valid
a[1] = 'a';         Valid
byte b = 10;
a[2] = b;           Valid
short s = 20;
a[3] = s;           Valid
a[4] = 10l;         // C.E : possible loss of precision
```

byte — short — int — long → float → double
Char

Example 2: For float type arrays, the allowed Element types are byte, short Char, int, long, float.

Case-2 In the case of object type array as array Element, we can provide either declared type object or its child class object.

Date

reference: Part-8 || Array part-3

```
Object [] a = new object [10];
a[0] = new Integer (10);
a[1] = new object ();
a[2] = new String ("bhaskar");
```

~~Public static void~~ | Public abstract class Number

```
Number [] n = new Number [10];
n[0] = new Integer (10);
n[1] = new Double (10.5);
n[2] = new String ("bhaskar");
```

C.E Java.lang.Error : Unresolved Compilation Problem:

Type mismatch : Cannot convert from String to Number

String is not a sub class of Number

Number

Byte short int long Float double.

Case 3 In the case of interface type arrays elements we can provide its implemented class object.

public class Thread implements Runnable

Whereas Runable is an interface

Runnable [ ] r = new Runnable [10];

r [0] = new Thread ( );

✗ r [1] = new String ("durga");

↳ C.E : incompatible types
found : j.l. String , required j.l. Runnable

for interface type arrays as array elements it's implementation class objects are allowed

Array Type	Allowed element type
Primitive type Array	Any type which can be implicitly promoted to declared type.
Object type Array	Either declared type or its child class objects
Abstract class Type Array	Its child class objects
interface Type Array	its implementation class objects are allowed

Date

Array Variable Assignments

Case I : Element level promotions are not applicable at array level

Ex: Char element can be promoted to int type. Whereas char Array can not be promted to int Array

Example

```
int[] x = {10,20,30,40};
char[] ch = {'a','b','c','d'};
int[] b = x;
int[] c = ch;
```

→ C.E : incompatible type
found : char[]
required : int[]

char[] → [C

int[] → [I

Which of the following promotion will be performed automatically ?

✓ Char → int

Date

× Char [] → int[]

✓ int → double

× int[] → double[]

× float → int

× float[] → int[]

✓ String → Object

✓ String[] → Object[]

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

example

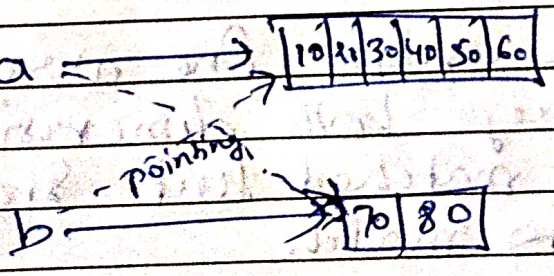
```
String [] s = {'A', 'B', 'C'};

object [] a = s;
```

Case II

```
int [] a = {10,20,30,40,50,60};
int [] b = {70,80};
```

1. a=b; } both valid perfectly
2. b=a }

Whenever we are assigning one array to another array, internal element won't be copied just reference variable will be re-assigned.



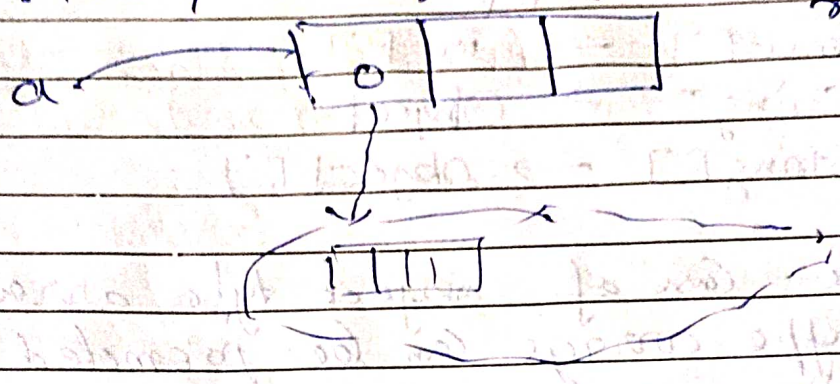
Date

Case 3: `int[][] a = new int[3][];`

`a[0] = new int[4][3];` ✗

↓

CE : incompatible type. found: int[][] required: int[]



`a[0] = 10;` → CE

incompatible type

found : int

required : int[]

`a[0] = new int[2]`

Whenever we are assigning one array to another array internal elements the dimension must be matched

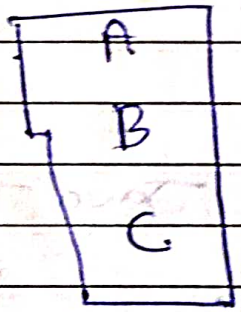
For example: In the place of one dimensional int array we should provide one dimensional array only. if we are trying to provide any other dimension, then we will get compile time error

Note: When we are assign one array to another array both dimension & types must be matched but sizes are not required to match

Date

```
class Test
{
    p s v main (String [] arg)
    { for (int i=0; i<= args.length ; i++)
        { sopln (args[i]);
        }
    }
}
```

Java test A B C ↵



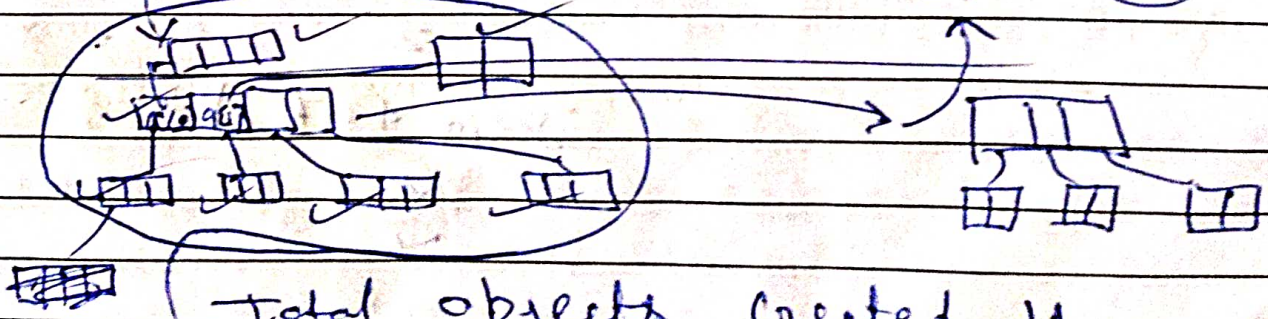
R.E. ArrayIndexOutOfBoundsException

Q. int [][] a = new int [4][3]; → (5) object created

a[0] = new int[4] → (1)

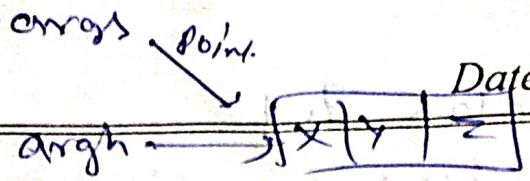
a[1] = new int[2]; → (2)

a[2] = new int[3][2]; → (4)



Total objects created 11

Garbage Collection 7



Date

```
class Test
{ p s v main (String [] arg)
  {
    String [] argh = {"x", "y", "z"};
    args = argh;
    for (String s : args)
    {
      Sop.ln (s);
    }
  }
}
```

Command line Argument pass

Java Test A B C ↵

x
y
z

Java Tes A B ↵

x
y
z

Java test ↵

x
y
z