Arrays:

Array is an indexed collection of fixed number of homogenous data elements.

the main advantage of arrays is we can represent multiple values with the same name. so that readability of the code will be increased.

But the main disadvatage of arrays is:

Fixed in size that is once we created an array there is no chance of increasing or decreasing the size based on our requirement that is to use arrays concept compulsory we should know the size in advance which may not possible always.

we can resolve the problem by using collections.

Array declarations:

Single dimensional array declaration:

ex: int[] a; // recomended to use because name is clearly separated from the type.

int a[];

int []a;

at the time of the declaration we cant specify the size other wise we will get compile time error.

ex; int[] a; //valid

int [5] a; //invalid

Two dimentional array decleration:

ex:

int [][] a;

int [][]a;

int a[][];

int [] []a;

int [] a[];

int []a[];

above all are valid (6 ways).

Three dimensional array declaration :

ex:

int [][][] a;

int [][][]a;

int a[][][];

which of the following declarations are valid?

1) int[] a1,b1; //a-1,b-1 valid

2) int[] a2[],b2; //a-2,b-1 valid

3) int[] []a3,b3; //a-2,b-2 valid

4) int[] a,[]b; //compile time error (invalid)

note:

if we want to specify the dimension before the variable that rule is applicable only for 1st variable.

Second variable onwards we cant apply in the same declaration.

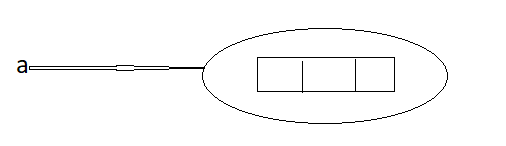
ex:

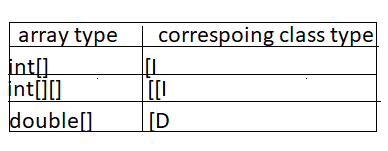
int[] []a,[]b;

first varable []a is valid but second variable b[] invalid.

ex

int[] a=new int[3];

 for every array type correspoding classes are available but these classes are part of java language and not available to the programmer level.



Rule 1:

At the time of array creation compulsory we should specify the size otherwise we will get compile time error.

example:

int[] a=new int[3];

int[] a=new int[]; //C.E : array dimension missing

Rule 2:

It is legal to have an array with size zero in java.

Example:

int[] a=new int[0];

System.out.println(a.length); //0

Rule 3:

If we are taking array size with -ve int value the we will get runtime exception saying

NegativeArraySizeException

Ex:

int[] a=new int[-3]; //Runtime Exception : NegativeArraySizeException

Rule 4:

The allowed data types to specify array size are byte,short, char,int.

By mistake if we are using any other type we will get compile time error.

Example:

int[] a=new int['a']; //valid

byte b=10;

int[] a=new int[b]; //valid

short s=20;

int[] a=new int[s]; // valid

int[] a=new int[101];//C.E : possible loss of precision//(invalid)

int[] a=new int[10.5];//C.E: possible loss of precision//(invalid)

Rule 5 :

The maximum allowed array size in java is maximum value of int size [2147483647]

Ex:

int[] a1=new int[2147483647]; (valid)

int[] a2=new int[2147483648];

//C.E : integer number too large: 2147483648(invalid).

In the first case we may get RE : OutOfMemoryError.

Multi dimensional array creation:

In Java multi dimensional arrays are implemented as array of arrays approach but not matrix form.

the main advantage of this approach is to improve memory utilization.

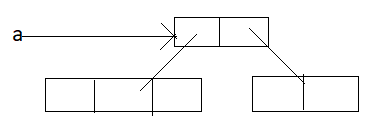
Ex 1:

int[][] a=new int[2][];

a[0]=new int[3];

a[1]=new int[2];

Memory representation



Example 2:

int [][][] a=new int[2][][];

a[0]=new int[3][];

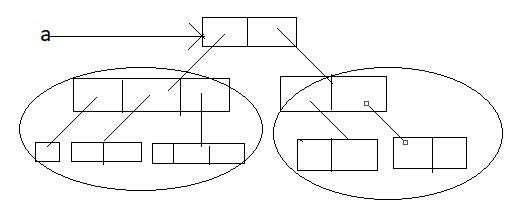
a[0][0]=new int[1];

a[0][1]=new int[2];

a[0][2]=new int[3];

a[1]=new int[2][2];

Memory representation



which of the following declarations are valid?

1) int[] a=new int[] //C.E array dimension missing(invalid)

2) int[][] a=new int[3][4]; (valid)

3) int[][] a=new int[3][]; (valid)

4) int[][] a=new int[][4];//C.E : ']' expected (invalid)

5) int[][][] a=new int[3][4][5]; valid

6) int[][][] a=new int[3][4][]; valid

7) int[][][] a=new int[3][][5]; //C.E : ']' expected (invalid).

Array Initialization:

whenever we are creating an array every element is initialized with default value automatically.

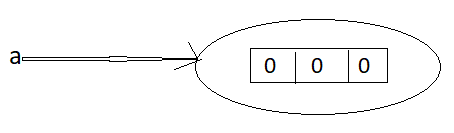
example 1:

int[] a=new int[3];

System.out.println(a); //[I@3e25a5

System.out.println(a[0]);//0

Diagram:



Note: Whenever we are trying to print any object reference internally toString() method will be executed which is implemented by default to return the following.

classname@hexadecimalstringrepresentationofhashcode.

Example 2:

int[][] anew int[2][3];

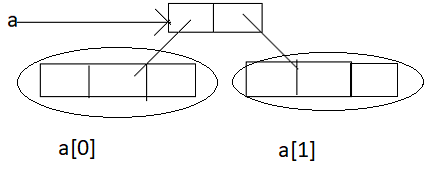
in above array base size of array is [2]

System.our.println(a);//[[I@3e25a5

System.our.println(a[]); //[I@19821f

System.our.println(a[0][0]); //0

Memory representaion:



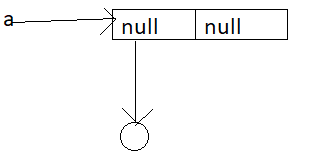
Example 3:

int[][] a=new int[2][];

System.out.println(a);//[[I@3e25a5

System.out.println(a[0]);//null

System.out.println(a[0][0]);//R.E: NullPointerException



Once we created an array all its elements by default initialized with default values. If we are not satisfied with those defaults values then we can replays with our customized values.

ex:

int[] a=new int[4];

a[0]=10;

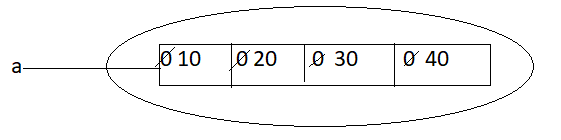
a[1]=20;

a[2]=30;

a[3]=40;

a[4]=50;//R.E: ArrayIndexOutOfBoundsException: 4

a[-4]=60;//R.E: ArrayIndexOutOfBoundsException: -4

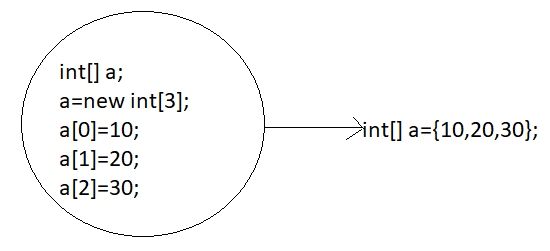


Note: if we are trying to access array element with out of range index we will get Runtime

Exception saying ArrayIndexOutOfBoundsException.

Declaration,construction and initialization of an array in a single line:

We can perform Declaration,construction and initialization of an array in a single line.



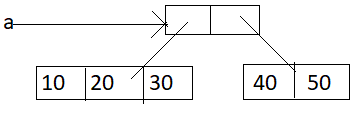
char[] ch=('a', 'e', 'i', 'o', 'u'); valid

String[] s=("chiru","balayya","venki","nag"); valid

We can extend this short cut even for multi dimensional arrays also.

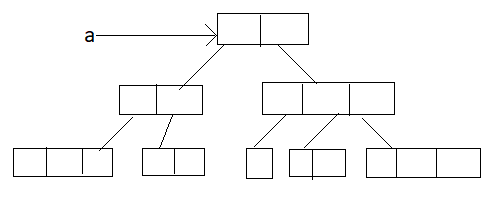
ex:

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



example:

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



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

System.out.println(a[0][1][1]);//50 (valid)

System.out.println(a[1][0][2]); //R.E: ArrayIndexOutOfBoundException: 2(invalid)

System.out.println(a[1][2][1]); //100(valid)

System.out.println(a[1][2][2]); //110(valid)

System.out.println(a[2][1][0]; //R.E: ArrayIndexOutOfBoundException: 2(invalid)

System.out.println(a[1][1][1]); //80 valid

.If we want to use this short cut compulsory we should perform declaration, construction and initialization in a single line.

.if we are trying to divide into multiple lines then we will get compiletime error.

length Vs length():

length:

1.It is the final variable applicable only for arrays

2.It represents the size of the array

ex:

int[] x=new int[3];

System.out.println(x.length());//C.E : cannot fine symbol

System.out.println(x.length);//3

length() method:

1.It is a final method applicable for String Objects

2.It returns the no of charactors present in the String.

ex:

String s="bhaskar";

System.out.println(s.length);//C.E : cannot find Symbol

System.out.println(x.length());//7

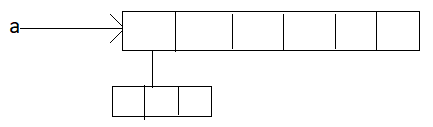
In multidimentional arrays length variable represents only base size but not total size.

ex:

int[][] a=new int[6][3];

System.out.println(a.length);//6

System.out.println(a[0].length);//3



length variable applicable only for arrays where as length() method is applicable for String objects.

There is no direct way to find total size of multi dimentional array but indirectly we can find as follows.

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

Anonymous Arrays:

1.Some times we can create an array without name such type of nameless arrays are called anonymous arrays.

2.the main objective of anonymous arrays is"just for instant use".

3.we can create anonymous array as follows.

4. new int[] {10,20,30,40} ; (valid)

5. new int[][] {{10,20},{30,40}}; valid

At the time of anonymous array creation we cant specify the size otherwise we will get compile time error.

ex:

new int[3] {10,20,30,40}; //C.E : ' ;' expected (invalid)

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

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

Ex:

int[] a=new int[]{10,20,30,40}; valid

ex:

class Test {

public static void main(String [] args) {

System.out.println(sum(new int[]{10,20,30,40})); //100

}

public static int sum(int[] x) {

int total=0;

for(int x1:x)

{

total =total+x1;

}

return total;

}}

Array elements assignments:

case 1:

In the case of primitive as array element any type is allowed which can be promoted to declared type.

Ex 1:

For the int type arrays 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]=101; //C.E possible loss of precision

example 2:

for float type arrays the allowed element types are byte ,short,char,int,long,float.

byte---->short ------->int---->long------>float----->double

char------> int---->long------>float----->double

case 2:

In the case of Object type arrays as array elements we can provide either declared type objects or its child class objects.

Ex 1:

Object[] a=new Object[10];

a[0]=new Integer(10); //valid

a[1]=new Object(); //valid

a[2]=new String ("bhaskar"); //valid

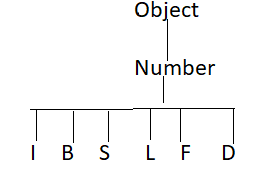
Ex2:

Number[] n=new Number[10];

n[0]=new Integer(10); //valid

n[1]=new Double(10.5); //valid

n[2]=new String("bhaskar'); //C.E : incompatible types . //invalid



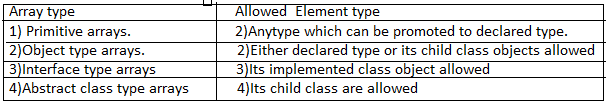
Case 3:

In the case of interface type arrays as array elements we can provide its implemented class objects.

Ex: Runnable[] r=new Runnable[10];

r[1]=new Thread();

r[1]=new String("bhaskar"); //C.E : incompatable types



Array variable assignments:

case 1:

. Element level promotions are not applicable at array object level

.Ex: A char value can be promoted to int type but char array cannot be promoted to int array.

Ex:

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

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

int[] b=a; //valid

int[] c=ch; //C.E imcompatible types(invalid)

which of the following promotions are valid?

1)char-----------------int (valid)

2)char[]-------------int[] (invalid)

3)int------------------long (valid)

4)int[]---------------long[] (invalid)

5)double-----------float (invalid)

6)double[]---------float[] (invalid)

7)String-----------Object (valid)

8)String------------Object[] (valid)

Note: In case of object type arrays child type array can be assign to parent type array variable.

Ex:

String[] s={"A", "B"};

Object[] o=s;

Case 2:

whenever we are assigning one array to another array internal elements wont be copy just reference variables will be reassigned hence sizes are not important but types must be matched.

example:

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

int[] b={80,90};

a=b; //valid

b=a; //valid

case 3:

whenever we are assigning one array to another array dimensions must be matched that is in the place of one dimensional array we should provide the same type only otherwise we will get compile time error.

Ex:

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

a[0]=new int[4][5]; //C.E : incompatable types (invalid)

a[0]=10; //C.E : incompatable types (invalid)

a[0]=new int[4]; //valid

Note: whenever we are performing array assignments the types and dimensions must be matched but sizes are not important.

Ex:

int[][] a=new int[3] [2];

a[0]=new{3];

a[1]=new int[4]; ,