# Arrays

* Disadvantage: fixed in size, can't be increased / decreased
* 'int array can hold only int type of array - homogeneous type of elements only
* An array is indexed collection of fixed number of homogeneous data elements.
* Collections - global in size, all elements

### Array Declaration:

int[] x; // recommended. - readability vise

int x[];

int x[]; // dimension mixed with variable. (more space between int and [])

int a []; //space is ignored by compiler.

int[4] x; //invalid

At the time of declaration, we are not allowed to declaration of size.

At the time of array creation, size is required.

int[][] x; // valid

int [][]x; //valid

int x[][]; //valid

int[] x[]; //valid

int [] x[]; //valid

int []x[]; //valid

Below all are valid

int[][][] a;

int [][][]a;

int a[][][];

int[][] a[];

int[][] []a;

int[] [][]x;

int[] []x[];

int[] x[][];

int []x[][];

int[] a,b; // both are one dim array

int[] a, b[]; //a is one and b is two

int[] []a, b; // a is two, b is two // space is ignored by compiler

int[] []a,b[]; // a=2 b=3

int[] a,[]b; // not allowed, []a this facility applicable to first variable only. Second variable wont compile

int[] a,b,[]c,d[]; .//not allowed

int[] a, b[], c[], d[]//ok

 Array Construction

int[] a=new int[3];

|  |  |  |
| --- | --- | --- |
|  |  |  |

Sout( a.getClass().getName());

[I -1 dim

[[I - 2dim

[[[I - 3 dim

Byte[] [b

Long[] [j;

Boolean[] [z

new is used with class, and we know class name, but for array what will be the class name.

a is a reference variable and using x we can get class name and using class name we can get name of class

* [ I is the name of class
* All classed available at compiler level.

int[] a=new int[0] //valid in java

int[] a=new int[0]; // compilar ok

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

System.out.println(a[0]);//java.lang.ArrayIndexOutOfBoundsException: 0

int[] a=new int[-4]; //compilation ok, because -4 is a valid int type so compiler is happy compile time.

System.out.println(a.length);//java.lang.NegativeArraySizeException

System.out.println(a[0]);//java.lang.ArrayIndexOutOfBoundsException: 0

Compiler always take care of type only int[] x=new int[-9];

JVM is responsible to allocate memory, JVM will throw NegativeArraySizeException

Index start with 0 only, java kept the same convention like other language.

int[] j=new int['a'];

System.out.println(j.length); // 97 output

byte, short and char are promoted to int,

-- only these are allowed. [byte, short, char, int]

Long type size is not allowed. Size should always be in int type.

long x=10;

Int[] aa= new int[x]; // invalid

double[] d= new double[1];

double[] dd=new double[10];

System.out.println(dd.length); //ok 10

double[] ddd= new double[(int)11.3333]; // size is in int

System.out.println(ddd.length); //ok 11

Size should be in (int, byte, short, char)

Type could be anything..

In Java, max length allowed in Array size - Integer.MAX

Int max size is allowed.

double[] d = new double[Integer.max];

java.lang.OutOfMemoryError: Requested array size exceeds VM limit.

**2D Array= Matrix**

Matrix - memory wastage is more.

In java, Arrays of Array concept //JAGGED ARRAY

Int[][] marks= new int[5][6]; //5 row 6 column

marks[0]={2,3}

marks[1]={1}

marks[2]={1,3,4}

marks[3]={3,4}

marks[4]={5}

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 2 | 3 | waste | waste | waste |
| 1 | waste | waste | waste | waste |
| 1 | 3 | 4 | waste | waste |
| 3 | 4 | waste | waste | waste |
| 5 | waste | waste | waste | waste |

Int[][] ar= new[3][ ]; //leave the second line size, because you may have variable size, base size compulsory need to specify

ar[0] = new int[3];

ar[1] = new int[2];

ar[2] = new int[1];

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

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

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

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

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

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

Int[][][] xxx=new int[1][2][3]; //valid

Int[][][] xxx=new int[1][2][]; //valid

Int[][][] xxx=new int[1][][]; //valid

Int[][][] xxx=new int[][][]; //in valid

Int[][][] xxx= new int[3][][4]; //in valid

Int[][] xxx= new int[][5]; //in valid

Int[][][][][][] xxx= new int[1][][2][][4][]; // invalid

int[] x= new int[3];

Syso(x[0]); // will print default value

|  |  |  |
| --- | --- | --- |
| 0 | 0 | 0 |

Syso(x[3]); // not compile time error; array index out of bound array exception. Runtime exception.

Syso(x); // toString() method will be called. classname@hashcode\_in\_hexadecimalform [I@123

int[] x= {1,2,3,4}; // should be in one line only, multiple line wont compile.

int[] x= new int[] {2,3,4};

int[] x;

x={10,20,30}; // invalid compile time error

int[][] x= { {1,2}, {3,4}}; // should be single line.

Length vs length()

--------------------

Length variable applicable for Array.

Length() method applicable for String

String[] x={"A", "AA", "AAA"};

s.length; //valid

s.length(); // invlid

s[0].length; //invalid

s[0].length();//valid

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

x.length; //3

x[0].length;//2

Anonymous Arrays - Array just for instant use/temporary requirement

----------------------------

new int[] {12,23,34};

int int[][]{{12},{2,34,5}};

Sum(new int[]{45,56,67});

new int[3]{3,4,5}; // size not required, compiler error

4292 0704 4585

int x= new int[10];

x[0]='a'; //valid

byte c=123;

x[1]=c;//valid

short s=23;

x[2]=s;//valid - int can accept int and lower data types (byte, short, char)

double[] d=new double[3];

//all lower datatype can be allowed (byte,short,char,int,float,long,double)

String[] s= new String[3]; //only string type is allowed.

Object[] o=new Object[3];

o[1] = new Object(); //ok

o[2]=new String("23"); //allowed

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

int[] b={50,60};

a=b; // a is point to b array; // size matching not required to match, type and dimension must be matched.

b=a; // b is pointing to a array, internal array is not copied, means b is having 2 size, 4 size does not matter.

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

x[0]=new int[4];

Q1

input

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

System.out.println(x[0].length);

x[0] = new int[4];

System.out.println(x[0].length);

x[0]= new int[]{2,3,4,5,6,7,8};

x[1]= new int[]{2,3,4,5,6,7,8};

x[2]= new int[]{2,3,4,5,6,7,8};

System.out.println(x[0].length );

Q2:

String[] s=new String[2];

int i=0;

for(String s1: s) {

s[i].concat("Element: "+i);

}

for(i=0;i<s.length;i++) {

System.out.println(s[i]);

}

Output:

Exception in thread "main" java.lang.NullPointerException

at com.pune.it.ArrayResizeTest.main(ArrayResizeTest.java:11)

Q3:

int[][] x= new int[1][3];

for(int i=0;i<x.length;i++) {

for(int j=0; j>x[i].length;j++){

x[i][j]=10;

}

}

System.out.println(Arrays.deepToString(x));

[[0, 0, 0]]

|  |  |
| --- | --- |
| class Bank{  String bankName;  String ifscCode; .. }  class Vehical{  }  class Animal{  }  class User{  }  class Customer{  }  class Loan{  } | Write attributes and methods |
| //Bean , POJO = plain object  class User {  int id;  String name;  public User(int id, String name) {  this.id = id;  this.name = name;  }  int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getName() {  return name;  }  public void setName(String name) {  this.name = name;  }  }  public class ArraysTest {  public static void main(String args[]) {  User userObject = new User(1, "Shiv");  User userObject2 = new User(2, "Swapnil");  User userObject3 = new User(4, "Sumit");  System.out.println(userObject.getId());//1  System.out.println(userObject.getName());// Shiv  System.out.println(userObject2.getId());//2  System.out.println(userObject2.getName());//Swapnil  System.out.println(userObject3.getId());//4  System.out.println(userObject3.getName());//Sumit  }  } |  |
| byte[] a = new byte[5];  System.out.println(Arrays.toString(a));  short[] b = new short[5];  System.out.println(Arrays.toString(b));  int[] c = new int[5];  System.out.println(Arrays.toString(c));  long[] d = new long[5];  System.out.println(Arrays.toString(d));  float[] f = new float[5];  System.out.println(Arrays.toString(f));  double[] g = new double[5];  System.out.println(Arrays.toString(g));  boolean[] h = new boolean[5];  System.out.println(Arrays.toString(h));  String[] i = new String[5];  System.out.println(Arrays.toString(i));  class Person {}  Person[] j = new Person[5];  System.out.println(Arrays.toString(j)); | Default value in Array  //int, byte, short, long -> 0  //String, Person -> null  //float, double -> 0.0  //boolean -> true/false -> default value false |
| int[] array; //declare  int array[]; //declare | declare |
| int[] array1 = new int[4];  int[] array2 = {1,2,3,4};  int[] array3 = new int[]{1,2,3,4}; | Declare and initialize |
| int[] array2 = {1,2,3,4};  at index 0 the values is 1  at index 1 the values is 2  at index 2 the values is 3  at index 3 the values is 4 | Index starts from 0 |
| int[] array= new int[5]; //size is mandatory | size is mandatory |
| int[] array= new int[5];  array[0] = 0;  array[4] = 4;  //0,0,0,0,4 | Add values in array using index |
| int[] array= new int[5]; //size is mandatory  array[0] = 1;  array[4] = 4;  array[1] = 2;  array[2] = 2;  //1,2,0,0,4 | Add values in array using index |
| int[] numbers = new int[]{1,2,3}; //valid  System.out.println(numbers[0]); //1 System.out.println(numbers[1]); //2 System.out.println(numbers[2]); //3  System.out.println(numbers[3]); // compiles fine //java.lang.ArrayIndexOutOfBoundsException: // Index 3 out of bounds for length 3   //skip System.out.println(numbers[100]); //compiles fine | Get value beyond index range |
| int[] a = new int[-1] | Not allowed negative size |
| int[] numbers = new int[]{1,2,3}; //valid    for (int i = 0; i <2 ; i++) { //1<2  System.out.println(numbers[i]);//1 2  }  for(int n: numbers){ //read only, all values will be read  System.out.println(n);//1 2  } | for vs for each |
| int[] numbers = new int[]{1,2,3}; //valid    for (int i = 2; i >0 ; i--) {  System.out.println(numbers[i]);  } | Reverse order |
| int[][] multi = new int[1][]; | Only first value size is mandatory, second array size is optinal |
| int[][] multi = new int[1][]; //2D  int[][] multi1 = {{1,2,3},{4,5,6},{7,8,9},{7,8,9}};  int[][] multi2 = new int[][]{{1,2,3},{4,5,6},{7,8,9}};  //{{1,2,3},{4,5,6},{7,8,9}};  //1 2 3  //4 5 6  //7 8 9 | 2D array |
| int[][] multi = new int[1][2]; //2D  int[][] multi1 = {{1,2,3},{4,5,6,60},{7,8,9,90,900,900},{7,8,9}};  // 0 1 2 3  int[][] multi2 = new int[][]{{1,2,3},{4,5,6},{7,8,9}}; | Variable size |
| int[][] multi = new int[1][1]; //2D //0 -> {8} //1 -> {9}  int[][] multi = new int[1][]; //2D //0 -> {8,0} //1 -> {9,10,1,12,,23,4,34,4,34} | 2D |
| int[][] multi = new int[3][]; //2D //0 -> {8,0,1} //1 -> {9,10}  //2 -> {10} |  |
| int[][][] multi = new int[1][][]; //3D  int[][] multi1 = { { {1,2}, {3}, {4,5,6} }, {{1},{3,5,6}}} | 3D |
| int[][][][][][] multi = new int[1][][][][][]; | N D |
| int array[] = new int[]; | Valid but not recomonded. |
| [Arrays (Java Platform SE 8 ) (oracle.com)](https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html) | <https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html>  Visit Arrays class provided by Java |
| int[] array = new int[]{9,2,1,0,6,7,3,4,8,2,3,5}; //valid Arrays.sort(array); System.out.println(Arrays.toString(array)); | Sort |
| String[] array = {"Aaakaash","Aakaash","Aakash"};  Arrays.binarySearch(array);  System.out.println(Arrays.toString(array)); | Sort |
| %  2%2= 0  4%2=0  100%2=0  1%2 =1  3%2=1 | % returns the reminder number. |
| for (int x = 1; x <= 100; x++) {  if(x%2!=0){  System.out.println(x);  }  } | Even no. less than 100 |