# **Java数组**

通常，数组是具有连续内存位置的类似类型的元素的集合。Java数组是一个包含类似数据类型的元素的对象。 它是一个数据结构，我们存储类似的元素。 只能在java数组中存储固定的元素集合。

java中的数组是基于索引的，数组的第一个元素存储的索引为：0。



****Java Array的优点****

****代码优化：**** 它使代码优化，可以轻松地检索或排序数据。  
****随机访问：**** 可以获取任何位于任何索引位置的数据。

****Java Array的缺点****

****大小限制：**** 只能在数组中存储固定大小的元素。 它在运行时不会增长其大小。 为了解决这个问题，在java中使用了集合框架。

****java中的数组类型****

有两种类型的数组。

* 一维数组
* 多维数组

****java中的单维数组****

在java中声明一个数组的语法。

dataType[] arr; (or)

dataType []arr; (or)

dataType arr[];

Java

****在java中实例化数组****

arrayRefVar=new datatype[size];

Java

****一维java数组的示例****

让我们来看看java数组的简单例子，下面声明，实例化，初始化和遍历数组。

class Testarray {

public static void main(String args[]) {

int a[] = new int[5];// declaration and instantiation

a[0] = 10;// initialization

a[1] = 20;

a[2] = 70;

a[3] = 40;

a[4] = 50;

// printing array

for (int i = 0; i < a.length; i++)// length is the property of array

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

}}

Java

执行上面代码的得到下面的结果 -

10

20

70

40

50

Shell

****Java数组的声明，实例化和初始化****

可以通过以下方式声明，实例化和初始化java数组：

int a[]={33,3,4,5};//declaration, instantiation and initialization

Shell

让我们来看看打印数组的简单例子。

class Testarray1 {

public static void main(String args[]) {

int a[] = { 33, 3, 4, 5 };// declaration, instantiation and

// initialization

// printing array

for (int i = 0; i < a.length; i++)// length is the property of array

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

}}

Java

执行上面代码，得到以下结果 -

33345

Java

****在Java中将数组传递给方法****

我们可以将java数组传递给方法，以便可以在数组上重复使用相同的逻辑。

让我们来看看获取使用方法的数组的最小数的一个简单的例子。

class Testarray2 {

static void min(int arr[]) {

int min = arr[0];

for (int i = 1; i < arr.length; i++)

if (min > arr[i])

min = arr[i];

System.out.println(min);

}

public static void main(String args[]) {

int a[] = { 33, 3, 4, 5 };

min(a);// passing array to method

}}

Java

执行上面代码，得到以下结果 -

3

Java

****java中的多维数组****

在这种情况下，数据存储在基于行和列的索引(也称为矩阵形式)中。在Java中声明多维数组的语法。

dataType[][] arrayRefVar; (or)

dataType [][]arrayRefVar; (or)

dataType arrayRefVar[][]; (or)

dataType []arrayRefVar[];

Java

****在java中实例化多维数组的示例****

int[][] arr=new int[3][3];//3 row and 3 column

Java

****在java中初始化多维数组的示例****

arr[0][0]=1;

arr[0][1]=2;

arr[0][2]=3;

arr[1][0]=4;

arr[1][1]=5;

arr[1][2]=6;

arr[2][0]=7;

arr[2][1]=8;

arr[2][2]=9;

Java

****多维java数组示例****

让我们来看看一个简单的例子来声明，实例化，初始化并打印二维数组。

class Testarray3 {

public static void main(String args[]) {

// declaring and initializing 2D array

int arr[][] = { { 1, 2, 3 }, { 2, 4, 5 }, { 4, 4, 5 } };

// printing 2D array

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

for (int j = 0; j < 3; j++) {

System.out.print(arr[i][j] + " ");

}

System.out.println();

}

}}

Java

执行上面代码，得到以下结果 -

1 2 32 4 54 4 5

Java

****java数组的类名是什么？****

在java中，数组是一个对象。 对于数组对象，创建一个代理类，其名称可以通过对象上的getClass()。getName()方法获取。

class Testarray4 {

public static void main(String args[]) {

int arr[] = { 4, 4, 5 };

Class c = arr.getClass();

String name = c.getName();

System.out.println(name);

}}

Java

执行上面代码，得到以下代码 -

I

Java

****复制java数组****

可以通过System类的arraycopy方法将数组复制到另一个数组。

****arraycopy方法的语法****

public static void arraycopy(

Object src, int srcPos,Object dest, int destPos, int length )

Java

****arraycopy方法的示例****

class TestArrayCopyDemo {

public static void main(String[] args) {

char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f', 'e', 'i', 'n', 'a', 't', 'e', 'd' };

char[] copyTo = new char[7];

System.arraycopy(copyFrom, 2, copyTo, 0, 7);

System.out.println(new String(copyTo));

}}

Java

执行上面代码，得到以下代码 -

caffein

Java

****在java中添加2个矩阵****

让我们来看看一个简单的例子，添加两个矩阵。

class Testarray5 {

public static void main(String args[]) {

// creating two matrices

int a[][] = { { 1, 3, 4 }, { 3, 4, 5 } };

int b[][] = { { 1, 3, 4 }, { 3, 4, 5 } };

// creating another matrix to store the sum of two matrices

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

// adding and printing addition of 2 matrices

for (int i = 0; i < 2; i++) {

for (int j = 0; j < 3; j++) {

c[i][j] = a[i][j] + b[i][j];

System.out.print(c[i][j] + " ");

}

System.out.println();// new line

}

}}

Java

执行上面代码，得到以下代码 -

2 6 86 8 10