Java数组

本站内容

- 数组类型与数组变量
- 数组创建
- 数组访问
- 二维数组
- 数组作为方法参数和返回值
- 数组操作的常用方法
- 数组应用举例
- 小结

概述

- 数组是相同类型数据的有序集合。
- 数组是对象。
- 一个数组包含一组变量，这些变量通常被称为数组元素，数组元素的数目称为数组长度。数组元素没有自己的名字，而是通过数组变量名和一个下标值来标识。
- 数组元素类型可以是Java中任意的数据类型，包括基本类型和引用类型。

1 数组类型与数组变量

1.1 数组类型

- 在Java中，数组被作为对象处理。
- 但是一个数组对象并不是某个类的一个实例。
- 数组类型（即数组对象的类型）用数组元素的类型跟随一对方括号来表示。
- 比如：

1 数组类型与数组变量

int[] // 数组元素为int型的数组对象类型

char[] // 数组元素为char型的数组对象类型

String[] // 数组元素为String型的数组对象类型

Point[] // 数组元素类型为类Point的数组对象类型

Object[] // 数组元素类型为类Object的数组对象类型

数组中的每个数组元素就如一个简单变量，可以存储一个基本型值或引用值。

注意：

数组类型不涉及到数组长度。数组长度在创建数组时指定。

1 数组类型与数组变量

1.2 数组变量

数组变量是指那些类型为数组类型的变量。

数组变量是引用型变量，其中存储着指向某个数组对象的一个引用值。定义数组变量的例子：

```
int[] ai;        // 定义一个类型为int[]的数组变量ai
String[] as;     // 定义一个类型为String[]的数组变量as
Point[] ap;      // 定义一个类型为Point[]的数组变量ap
Object[] ao;     // 定义一个类型为Object[]的数组变量ao
int ai[];        // 定义一个类型为int[]的数组变量ai
String as[];     // 定义一个类型为String[]的数组变量as
```

1 数组类型与数组变量

数组变量定义语句只是声明某个变量的类型是某种数组类型，并没有创建具体的数组对象。

如果一个数组变量v的类型是a[]，a是基本类型，那么v只能指向某个a[]型数组对象。如：

int[] ai; //ai只能存放数组对象为int[]型的引用值

如果一个数组变量v的类型是x[]，x是引用类型，那么v可以指向任何一个y[]型数组对象，只要y能够赋值转换成x。如： **Object[] ao;**

ao = 一个String[]型数组对象的引用值

注意区分数组变量的类型与其所指数组的类型。

int[]　　int

2 数组创建

创建的两种方式：

- 使用数组创建表达式
- 使用数组初始化块

用数组创建表达式创建数组时，数组各元素会有一个默认初始值。

用数组初始化块来创建数组时，需要为各数组元素指定初始值。

2 数组创建

2.1 数组创建表达式

格式如下：　　new <类型>[<表达式>]

　　数组创建表达式的类型为<类型>[]，其结果是创建一个<类型>[]型的数组，并返回对该数组对象的引用。

<表达式>的值用于指明数组的长度，即数组中元素的个数。<表达式>的值必须大于等于0。

表达式：常量，有值的变量，整型表达式

注意与C语言的区别

```
int arr[10];     //维数不能为变量，无初值
int[] arr=new int[len];
```

2 数组创建

一旦数组被创建，数组的长度是不能够改变的。

```
// 定义长度为10的int[]型数组并将引用值赋给变量ai
int[] ai = new int[10];
// 定义长度为3的Point[]型数组并将引用值赋给变量ap
Point[] ap = new Point[3];
// 定义长度为5的Point[]型数组并赋给Object[]型数组变量ao
Ojbect[] ao = new Point[5];
```

2 数组创建

在用数组创建表达式创建数组时，系统将为数组分配内存空间，并给每个数组元素赋以一个默认的初始值。数组元素的默认初始值与其类型有关，和相应类型的成员变量的默认初始值是相同的，如表所示。

元素类型	默认初始值	元素类型	默认初始值
byte	**0**	**short**	**0**
int	**0**	**long**	**0L**
Float	**0.0F**	**double**	**0.0**
Char	**'u0000'**	**boolean**	**false**
对象引用	**null**		

2 数组创建

2.2 数组初始化块

数组初始化块在创建数组的同时，能用花括号将一组表达式括起来形成，各表达式之间用逗号分隔。数组初始化块只能用在数组变量定义语句中，其格式如下：

<类型>[] <数组变量>={<表达式表>};

该语句将数组变量定义、数组创建以及数组元素初始化合并成一步完成。

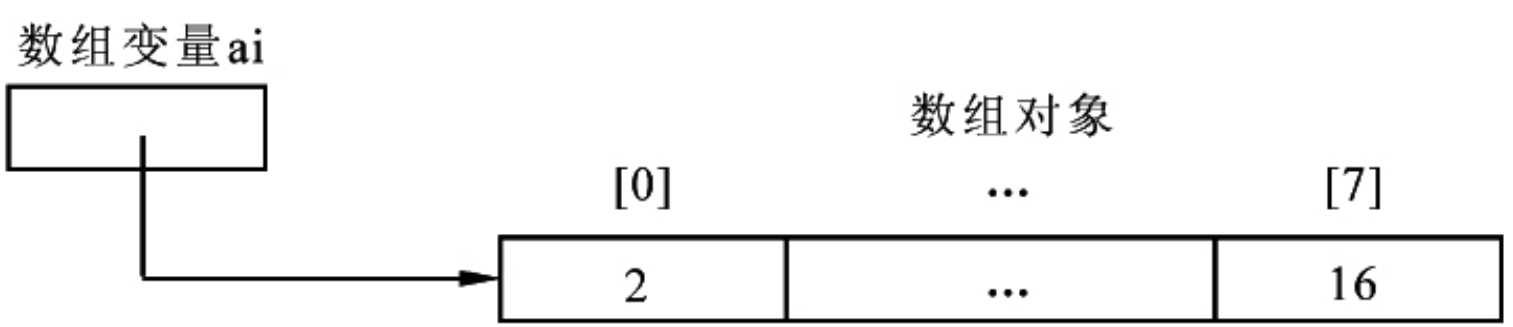
每个表达式按其次序从左到右依次被计算，计算结果作为对应数组元素的初值。

2 数组创建

下面是使用数组初始化块的例子：

- **int[] ai = {2, 4, 6, 8, 10, 12, 14, 16};**

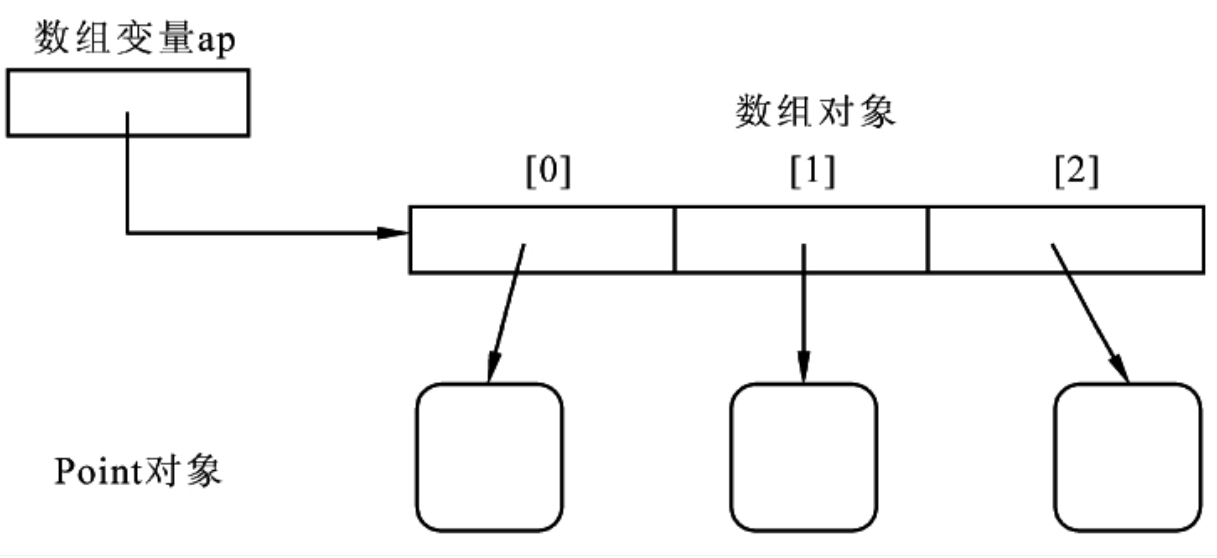
  创建了一个int[ ]型数组，数组的长度为8。



2 数组创建

- **Point[] ap = {new Point(), new Point(5,10), new Point(12,7)};**

创建了一个Point[]型数组，数组的长度为3，各数组元素分别引用一个Point类的实例。



2 数组创建

- **Point[] ar = {new Point(), new Point3D()};**

创建了一个Point[]型数组，数组的长度为2。其中，第1个数组元素引用一个Point类的实例，第2个数组元素引用一个Point3D类的实例。这里，Point3D类是Point类的一个子类。

3 数组访问

- 数组是一种对象，它也有自己的成员变量和方法。
- 数组一旦创建，就可以通过数组对象的引用访问数组中的每一个元素，或者访问数组的成员变量和方法。

<数组对象引用>[<下标>]

<下标>是一个表达式，其类型可以是byte、char、short或int型，但最终都会自动单目算术提升为int型。<下标>的类型不能是long型。

<下标>的取值从0开始，一直到数组的长度减1。如果<下标>值超出了允许的取值范围，将引发运行时例外ArrayIndexOutOfBoundsException。

3 数组访问

3.1 对数组元素的访问

int[] ai = new int[10];

访问数组ai各元素的表达式分别为：

ai[0]、ai[1]、ai[2]、ai[3]、ai[4]、ai[5]、ai[6]、ai[7]

Point[] ap = new Point[3];

下面循环语句利用数组访问表达式对数组ap的各元素进行实例化：

```
for (int i = 0; i < 3; i++)    ap[i] = new Point();
```

该循环语句使用不带形参的构造方法创建了3个Point对象，而数组ap的3个数组元素分别指向这3个Point对象。

3 数组访问

3.2 对数组成员的访问

数组类型可以看作是Object类的直接子类，其成员包括如下几类。

- 一个公共的有名常量length，用以记录数组对象的长度。利用该有名常量可以将上面的循环语句改写为：

```
Point[]  ap = new Point[3];
for (int i = 0; i < ap.length; i++)
{     ap[i] = new Point();     }
```

- 从Object类中继承的方法，如equals、toString等方法。

3 数组访问

```
a == b : false
a == c : true
```

```
class Test
{    public static void main(String[] args) {
      int[] a = {10, 11, 12},  b = {10, 11, 12};
      int[] c = a;
      System.out.println(″ a == b : ″  + a.equals(b));
      System.out.println(″ a == c : ″  + a.equals(c));
}    }
```

利用equals方法判断两个数组变量是否指向同一个数组对象(与String不同)。数组a和数组b虽然各元素的值都相同，但它们是两个不同的数组对象。数组变量a和c指向同一个数组。

4 二维数组

在Java语言中，所谓二维数组是指数组的嵌套，即数组的数组。

一个数组的数组元素类型既可以是基本类型，也可以是引用类型。其中，引用类型就包括数组类型。当一个数组的数组元素类型本身是数组类型时，就形成了二维数组，甚至三维数组、四维数组等。

int[][]

表示一个**int**型的二维数组。其中： 第**1**个方括号可以理解为第一维（外层）数组，其元素类型为**int[]**型；第**2**个方括号可以理解为第二维（被嵌套的内层）数组，其元素类型为**int**型。

4 二维数组

下面语句定义了一个int型二维数组的数组变量：

```
int[][]  aai;
int  aai[][];
int[]  aai[];
```

这里，方括号既可以写在元素类型名后也可以写在数组变量名后，或者分而写之。

上面3条语句有完全相同的效果。

下面语句创建了一个int型二维数组：

```
aai = new int[3][4];
```

4 二维数组

注意： 在使用new运算符创建二维数组时，外层数组的长度一定要指定，而内层数组的长度既可以指定也可以留待以后再确定。比如：

```
aai = new int[3][];
```

该语句仅创建了二维数组的外层数组，各内层数组并没有被创建。外层数组的长度为3，各元素的取值为null。可以根据需要分别创建各内层数组、指定它们的长度。各内层数组的长度完全可以不同，比如：

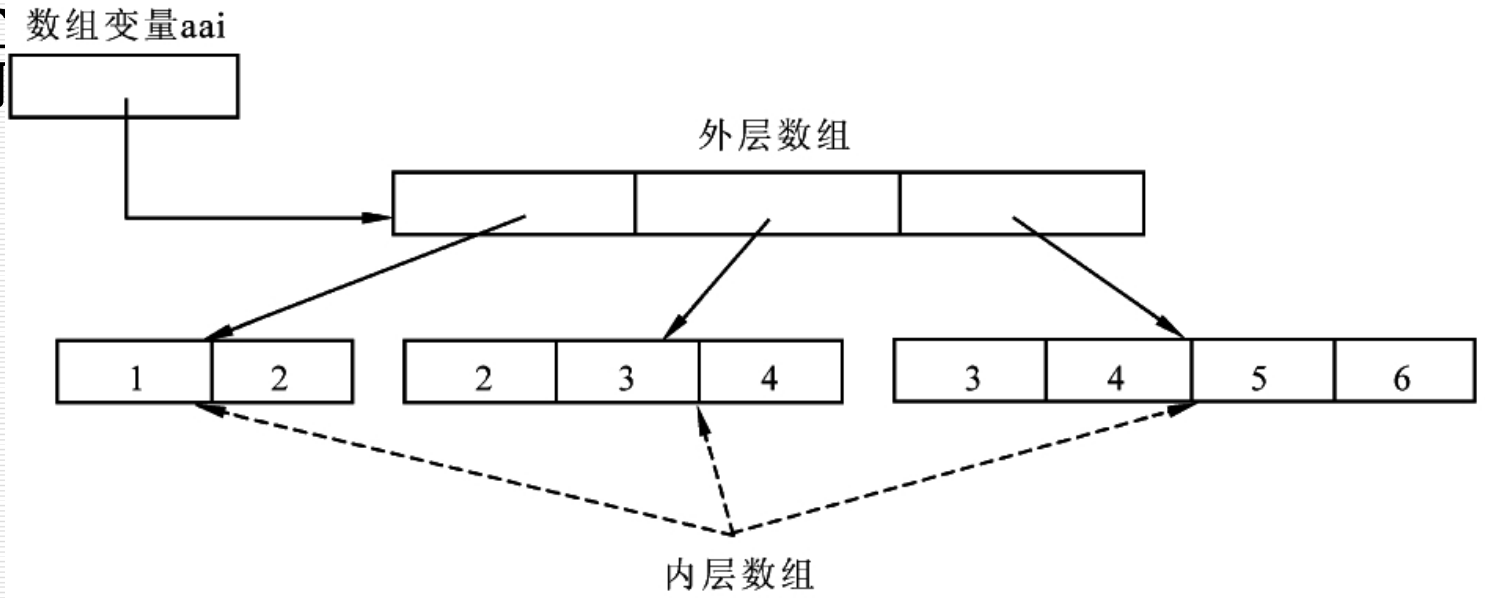
```
aai[0] = new int[2];
aai[1] = new int[3];
aai[2] = new int[4];
```

4 二维数组

可以用嵌套的数组初始化块来创建二维数组，并同时给二维数组各元素赋值。比如：

int[][] aai = {{1, 2},{2, 3, 4},{3, 4, 5, 6}};

该语句创建了一个与前面语句创建的数组有相同结构的二维数组，另外，它还用指定的值对数组各元素进行



4 二维数组

也可以通过数组访问表达式对二维数组各元素进行显式赋值：

```
for (int i = 0; i < aai.length; i++)
{
    for (int j =0; j < aai[i].length; j++)
    {
        aai[i][j] = i + j + 1;
    }
}
```

5 数组作为方法参数和返回值

数组是复合类型，数组变量存储的是数组存储区的引用，所以，传送数组或返回数组实际上在传送引用。

例:数组作为方法参数

```
class ArrayArgument
{    public static void main(String args[])
     {    int[] x = { 11, 12, 13, 14, 15 };
          display(x);
          change(x);
          display(x);                              }
     public static void change(int[] x)
     {    for(int i = 0;i < x.length; i++ )
               x[i] = x[i] + 10;                   }
     public static void display(int[] x)
     {    for(int i = 0; i < x.length; i++)
               System.out.print(x[i] + " ");
          System.out.println("");                  }
}
```

运行结果如下：

11 12 13 14 15

21 22 23 24 25

数组作为方法返回值

```
class Test
{   public int[] change(int x[])
    {  int y[] = { 1, 2, 3, 4, 5 };
       x=y;        return x;        }
    public void display(int x[])
    {   for (int i = 0; i < x.length; i++)
              System.out.print(x[i] + " ");
       System.out.println("");    }
    public static void main(String args[])
    {  Test tt=new Test();
       int x[] = { 11, 12, 13, 14, 15 };
       tt.display(x);
       x=tt.change(x);
       tt.display(x);    }        }
```

运行结果如下：

11 12 13 14 15

1 2 3 4 5

```
class ReturnArray
{    public static void main(String args[])
     {    double a[] = {1.1, 3.4, -9.8, 10};
          double b[] = max_min_ave(a);
          for(int i = 0;i < b.length;i ++)
               System.out.println("b[" + i +"]= " + b[i]);    }
      static double[] max_min_ave(double a[])
     {    double res[] = new double[3];
          double max = a[0], min = a[0], sum = a[0];
          for(int i = 1;i < a.length;i ++)
          {       if(max < a[i])
                       max = a[i];
                  if(min > a[i])
                       min = a[i];
                  sum += a[i];     }
          res[0] = max;                    res[1] = min;
          res[2] = sum/a.length;     return res;        }        }
```

6 数组应用举例

```
class ArraySort
{   static void sort(int[] a)  //排序算法？
    {     int t;
          for (int i=0;i<a.length-1;i++)
          {  int k=i;
             for (int j=i+1; j<a.length;j++)
                if (a[k] > a[j])   {   k = j;   } //找最小值位置
             if(k!=i)     //一次遍历若最小值不是i位置数值
             {   t=a[k];    a[k]=a[i];   a[i]=t;   }
          }
     }
```

```
public static void main(String args[])
   {  int[][] data = {{1,4,2,6}, {3,2}, {5,2,4}};
      sort(data[0]);    //对每一维排序
      sort(data[1]);
      sort(data[2]);
      for (int i = 0; i < data.length; i++)
      {   for (int j = 0; j < data[i].length; j++)
          {     System.out.print(data[i][j] + "");    }
         System.out.println();
       }
   } //end main
} //end class
```

程序运行的输出结果应该是：

```
1  2  4  6
2  3
2  4  5
```

例:方法返回类型为数组类型。定义方法method，其功能是创建一个指定大小的三角形二维数组，并用指定的值初始化数组各元素。

```
class ArrayCreate
{   static byte[][] method(int n, byte initialValue)
    {  byte[][] myArray = new byte[n][];
       for (int i=0; i<myArray.length;i++)
       {   myArray[i] = new byte[i+1];
           for (int j = 0; j < myArray[i].length; j++)
                 myArray[i][j] = initialValue;       }
       return myArray;
    }
```

```
public static void main(String args[])
{    byte[][] data = method(5, (byte)8);
     for (int i = 0;i < data.length; i++)
     {
        System.out.print("data[" + i + "] : ");
        for (int j=0; j<data[i].length;j++)
           System.out.print(data[i][j] + " ");
        System.out.println();
     }
  }
}
```

程序的运行结果是:

```
data[0] : 8
data[1] : 8  8
data[2] : 8  8  8
data[3] : 8  8  8  8
data[4] : 8  8  8  8  8
```

数组一旦建立，其长度是不能改变的。现在请定义一个IntArray类，作为可变长int[]型数组的一种实现。该类的软件接口如下：

```
public class IntArray
{    public IntArray();
     //向数组添加一个元素
     public void addElement(int value);
     //设置指定元素的值
     public void setElement(int n, int value);
     public int getElement(int n);//返回指定元素的值
     //返回数组当前长度
     public int getLength();
}
```

```
public class IntArray
{
   private static final int INCREMENT = 10;
   private int[] data;
   private int next;       //标记
   public IntArray()
   {   data = new int[INCREMENT];
       next = 0;    }
   public void setElement(int n, int value)
   {   data[n] = value;     }
   public int getElement(int n)
   {   return data[n];       }
   public int getLength()
   {   return next;          }
```

```
public void addElement(int value)
  {   if (next == data.length)    //?
      {
         int[] newData = new int[data.length +
                                        INCREMENT];
         for (int i = 0; i < data.length; i++)
         {   newData[i] = data[i];    } //将原数组复制过来
         data = newData;    //使原数组名指向新数组
      }
      data[next] = value;  //增加新元素
      next = next + 1;
  }
}
```

7 数组操作的常用方法

7.1 类System的静态方法arraycopy():

public static void arraycopy(Object src, int src_position, Object dst, int dst_position, int length)

从源数组src的src_position处，复制到目标数组dst的dst_position处，复制长度为length。

例4：用方法arraycopy()复制数组。

```
class ArrayCopy
{   public static void main(String args[])
    {   int array1[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 };
        int array2[] = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
        System.arraycopy(array1, 0, array2, 0, 5);
        System.out.print("array2: ");
        for(int i=0;i<array2.length;i++)
        {   System.out.print(array2[i] + " ");    }
    }
}
```

程序运行结果如下：

array2: 0 1 2 3 4 0 0 0 0 0

7 数组操作的常用方法

程序运行结果如下：

3 4 5 6 7 8

7.2 java.util.Arrays类中静态方法

(1)void sort(Object[] a)

例：使用sort()方法对一整型数组递增排序

```
import java.util.*;
public class ArraySort
{    public static void main(String args[])
     {    int a[]={8,6,7,3,5,4}, i;
          Arrays.sort(a);
          for(i = 0;i < a.length;i ++)
               System.out.println(" " + a[i]);     }  }
```

7 数组操作的常用方法

(2)int binarySearch(Object[] a,Object key)

```
import java.util.*;
public class BinarySearch
{   public static void main(String args[])
    {      int a[]={3,4,5,6,7,8},   i;     // 数组要求已排序
           // 在数组中查找数据6
           i=Arrays.binarySearch(a,6);
           System.out.println(i);
    }
}
```

运行结果为：3

7 数组操作的常用方法

7.3 数组的比较：不能使用=

```
int[][] arr1={ { 1, 2, 3 }, { 4,
int[][] arr2={ { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };
int[][] arr3={ { 0, 1, 3 }, { 4, 6, 4 }, { 7, 8, 9 } };
System.out.println("arr1内容等于arr2？" +
Arrays.deepEquals(arr1, arr2));
System.out.println("arr1内容等于arr3？" +
Arrays.deepEquals(arr1, arr3));
System.out.println("arr1 deepToString()\n\t" +
Arrays.deepToString(arr1));
```

```
arr1内容等于arr2？true
arr1内容等于arr3？false
arr1 deepToString()
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

8 小结

（1）数组是数据的有序集合，通过数组可以将一组数据组织在一起，并有助于对这组数据进行有效处理。

（2）一个数组包含一组变量，称为数组元素。数组元素的数目称为数组长度。

（3）一个数组中的每个数组元素都有相同的数据类型。数组元素类型可以是Java中任意的数据类型，包括基本类型和引用类型。

（4）在Java中，数组是对象。数组的类型用数组元素的类型跟随一对方括号来表示。数组类型是一种系统内置的引用类型。

8 小结

（5）数组变量是指那些类型为数组类型的变量。数组变量是引用型变量，其中存储着指向某个数组对象的一个引用值。

（6）一个数组可以使用数组创建表达式（new <类型>[<表达式>]）来创建。数组创建后，各元素有默认的初始值。

（7）一个数组也可以使用数组初始化块来创建。用数组初始化块来创建数组时，需要为各数组元素指定初始值。

8 小结

（8）数组访问表达式（<数组对象引用>[<下标>]）的类型是数组元素的类型，其值是对应数组元素的值。数组访问表达式是左值表达式，可以出现在赋值操作符的左边。

（9）与其他引用类型一样，数组类型也有成员，包括有名常量length；从Object类中继承的equals、toString等方法；clone方法等。

（10）在Java中，多维数组是指数组的数组，即当一个数组的数组元素类型本身是数组类型时，就形成了二维数组，甚至三维数组、四维数组等。

下课！ 

Thank you!