一. 数组变量的声明；

声明数组：

1) 一个存放同一类型数据的集合

a. 即可以是基本类型，也可以是对象类型；

b. 数组中的每个数据为元素；

2) 数组是一个对象，成员是数组长度和数组中的元素;

3) 声明了一个数组变量并不是创建了一个对象;

4) 声明数组的方式：

数组也是一种数据类型，它本身是一种引用类型

int是一个基本类型，int[]是一种引用类型

int[] IArray 或者 int IArray[] 基本数据类型数组，数组中存放的是基本数据类型。

Teacher[] tArray 或者 Teacher tArray[] 类类型数组，数组中存放的是Teacher类创建的若干个的对象。

type[] arrayName;推荐使用这种方式

type arrayName[];可读性差

JAVA的模仿者c#就不再支持 type arrayName[]这种语法，它只支持第一种定义数组的语法。

越来越多的语言不再支持第二种数组定义语法。

注意：1) 声明数组变量的时侯，不能指定数组的长度，以下声明方式是非法的。

int x[1];

int[2] x;

2) 数组是一种引用类型的变量，因此使用它定义一个变量时，仅仅表示定义了一个引用变量（也就是定义了一个指针），这个引用变量还未指向任何有效内存，因此定义数组时不能指定数组的长度。没有内存，这个数组就不能使用，必须初始化后才可以使用。

二. 初始化

初始化：自变量创建后首次赋值的过程；

一旦数组的初始化完成，数组在内存中所占的空间将被固定下来，因此数组的长度将不可改变。

1. 创建数组对象；

数组对象和其他Java对象一样，也用new语句创建；

int[] iArray = new int[2];

new int[2]语句执行以下步骤：

a. 在堆区中为数组分配内存空间，以上代码创建了一个包含2个元素的int数组

b. 为数组中的每个元素赋予其数据类型的默认值。

整数类型byte/short/int/long 0

浮点类型float 0.0f

浮点类型double 0.0d

字符类型char '\u0000'

布尔类型boolean false

引用类型 类、接口、数组 null

c. 返回数组对象的引用

在用new语句创建数组对象时，需要指定数组长度。数组长度表示数组中包含的元素数目。数组长度可以用具体的数值表示，也可以用变量表示。如：

int[] x = new int[10];

或

int size=10;

int[] x = new int[size];

数组的长度可以为0，此时数组中一个元素也没有。例如：

int[] x = new int[0];

对于Java类的程序入口方法main(String args[]),如果运行时这个类没有输入参数，那么main()方法的参数args并不是null, 而是一个

长度为0的数组。例如：

public class Sample {

public static void main(String[] args) {

System.out.println(args.length); //打印0

}

}

数组对象创建后，它的长度是固定的。数组对象的长度是无法改变的，但是数组变量可以改变所引用的数组对象。

int[] x = new int[3];

int[] y = x;

x = new int[4];

2. 初始化数组对象；

数组中的每个元素都有一个索引，或者称为下标。数组中的第一个元素的索引为0，第二个元素的索引为1, 依次类推。

通过索引可以访问数组中的元素或者给数组中元素内容赋值。

1) 声明、创建、初始化分开：

//声明

int[] iArray;

//创建

iArray = new int[2];

//初始化

iArray[0] = 0;

iArray[1] = 1;

2) 声明、创建的同时并初始化数组；

int[] iArray = {0, 1};

int[] iArray = new int[]{0, 1};

Student sArray[] = new Student[] { new Student(“George”, “Male”, 20), new Student()};

Student[] stArray = { new Student(), new Student()} ;

注意：a.下面这些都是非法的数组初始化方式：

int[] x = new int[5]{5,4,3,2,1}; //编译出错，创建数组对象的同时并初始化的时候就不能指定数组长度了；

int[] x;

x = {5,4,3,2,1}; //{5,4,3,2,1}必须在声明数组变量的语句中使用，不能单独使用

使用数组

为数组元素赋值、访问数组元素值、获得数组长度等

for(起始值;循环判断条件;迭代){}

for(int i=;i<arrayName.length;i++){ arrayName[i]}

JAVA5之后新增foreach循环的语法格式：

for(type variableName:arrayName | collection){variableName}

foreach中的循环变量variable相当于一个临时变量，并不是数组元素，并不能改变数组元素的值，因此不要对foreach的循环变量进行赋值。虽然赋值在语法上是允许的，但没有太大的实际意义，而且极易引起错误。

int[] a={5,7,8}; //a.length=3

int[] b= new int[4]; //b.length=4

b=a; //b.length=3

b的长度可变的原因：a与b都是引用变量，实际内存地址发生了改变，画栈、堆内存图。

引用类型数组的数组元素是引用，它指向另一块内存，这块内存里存储了有效数据。

引用指向同一内存区，访问和修改相互影响。

3. 多维数组；

Java支持多维数组。假定某个宾馆有三层楼，第一层有4个房间，第二层有3个房间，第三层有5个房间。某一天客户人住宿情况如

下图所示:

第三层: | |Tom |Jerry| |Rose|

第二层： |Mary| |Kevin|

第一层： |Mike|Jane|Duke | |

可以用两维数组来存储各个房间的客人信息。

String[][] room = new String[3][];

room[0] = new Stirng[]{"Mike","Jane","Duke",null};

room[1] = new String[]{"Mary",null,"kevin"};

room[2] = new String[]{null,"Tom","Jerry",null,"Rose"}

以上代码等价于：

String[][] room = {

{"Mike","Jane","Duke",null},

{"Mary",null,"kevin"},

{null,"Tom","Jerry",null,"Rose"}

};

多维数组本质上是数组的数组，数组的元素内容还是数组；

写一个类MultiArray.java来创造一个二维数组并输出其中元素

4. 数组的边界：

1) 一个数组的下标从0开始，数组通过数组的对象引用变量的下标访问数组。

数组中第一个元素的索引为0, 第二元素的索引为1，依次类推。如果一个数组长度是5，要访问最后一个数组元素可以通过

下标4来访问，如果通过下标5访问，超出了数组的边界，在运行时会抛出ArrayIndexOutOfBoundsException。

2) 通过调用数组的length方法可以获得一个数组的元素个数（数组长度）。

所有Java数组都有一个length属性，表示数组的长度. 该属性只能读取，但是不能修改。

以下修改数组的length属性，这是非法的。

int[] x = new int[4];

x.length = 10; //编译出错，length属性不能被修改。

注：a. 数组变量必须在指向一个数组对象之后，才能访问其元素。

public class Sample {

private int[] x;

public static void main(String[] args) {

Sample s = new Sample();

System.out.println(s.x); //打印null

System.out.println(s.x[0]); //运行时抛出NullPointerException

System.out.println(s.x.length); //运行时抛出NullPointerException

}

}

b. 当数组的元素为引用类型时，数组中存放的是对象的引用，而不是对象本身；

练习：1) 求一组值的平均值：

public class ArrayAvgTest {

public double avg(int[] n) {

double result = 0.0;

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

result += n[i];

}

result /= n.length;

return result;

}

public static void main(String[] args) {

ArrayAvgTest a = new ArrayAvgTest();

int[] n = {100,60,80,90,75,38};

System.out.println("Avg score: " + a.avg(n));

}

}

2) 求一组值的最大值：

public class ArrayMaxTest {

public int max(int[] n) {

int result = n[0];

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

if(result<n[i])

result = n[i];

}

return result;

}

public static void main(String[] args) {

ArrayMaxTest a = new ArrayMaxTest();

int[] n = {100,60,80,90,75,38};

System.out.println("Max score: " + a.max(n));

}

}

3) 数组内容排序

冒泡排序：值较小的数逐渐向数组的顶部(即朝第一个元素)冒上来，就像水中的气泡上升一样，同时，值较大的数据逐渐向数组的底部(即朝最后一个元素)沉下去。这种算法用嵌套的循环对整个数组进行数次遍历，每次遍历都要比较数组中相邻的一对元素，如果这对元素以升序(或者值相等)的顺序排列，那么保持它们的位置不变；如果这对元素以降序的顺序排列，那么交换它们的值。