**JAVA基础**

# 数组

1、引用类型

2、在任何数据类型后面加上[]，便成了数组

int[] int类型的数组，只能存放int类型的数据

String[] String类型的数组 只能String类型的数据

int[][] 数组的数组，二维数组 第一位只能存放int类型的数据，第二位只能存放int类型数组的数据

3、数组一旦初始化之后，数组长度不可再变更，length

4、数组声明

|  |
| --- |
| **public** **class** ArrayDemo {  **public** **static** **void** main(String[] args) {  // 数组声明,推荐使用将中括号在前面的声明 方式  **int**[] iArr;  String[] sArr;    **int** iArr1[];  String sArr1[];  }  } |
| **public** **class** ArrayDemo {  // 数组声明,推荐使用将中括号在前面的声明 方式  **static** **int**[] *iArr*;  **static** String[] *sArr*;  **public** **static** **void** main(String[] args) {  System.***out***.println(*iArr*);// null  System.***out***.println(*sArr*);// null  }  } |
| **static** **int**[] *iArr*;  **public** **static** **void** main(String[] args) {  // 数组有一个长度的属性，使用方式如下  System.***out***.println(*iArr*.length);// null  }  Exception in thread "main" java.lang.NullPointerException  at com.gongnizant.ArrayDemo.main(ArrayDemo.java:12)  出现这个空指针异常的原因，是代码的错误导致的。企图用一个空（null）的东西（*iArr*），去调用他（*iArr*）的属性（length），于是便导致了空指针异常，发生异常时，会使整个程序崩溃。 |

5、数组的初始化

|  |
| --- |
| **public** **class** ArrayInit {  **public** **static** **void** main(String[] args) {  // 静态初始化  **int**[] iArr = {1,2,3,4};  System.***out***.println(iArr);// [I@15db9742  System.***out***.println(iArr.length);// 4  // 数组的下标从0开始，访问数组的第一个元素  System.***out***.println(iArr[0]);// 1  // 访问数组最后一个元素  System.***out***.println(iArr[iArr.length-1]);// 4  System.***out***.println("======="); //System.***out***.println(iArr[iArr.length]);  System.***out***.println(iArr[5]);  }  }  Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5（发生下标越界的位置）  at com.gongnizant.ArrayInit.main(ArrayInit.java:17)  数组下标越界的异常，会导致程序崩溃，根据提示的位置，找到对应的代码，检查数组访问时下标的变化，自己推导或通过断点调试的方式，找出问题并修复。 |
| // 只要有new，就意味着在内存分配存储空间，新开辟一个存放5个整型元素的空间  // 动态创建并初始化  **int**[] iArr1 = **new** **int**[5];// 5代表的是length，所以最大下标为4  **for**(**int** i = 0; i < 5; i++){  System.***out***.print(iArr1[i] + " ");// 0  }  System.***out***.println();  // 为数组的每个元素赋一个[3,99]的随机数  // int a = (int) (Math.random() \* 97 + 3);  System.***out***.println("--------------------");  **for**(**int** i = 0; i < iArr1.length; i++){  iArr1[i] = (**int**) (Math.*random*() \* 97 + 3);  }  **for**(**int** i = 0; i < 5; i++){  System.***out***.print(iArr1[i] + " ");  } |
| String[] strArr = **new** String[5];  **for**(**int** i = 0; i < strArr.length; i ++) {  System.***out***.println(strArr[i]);// null  }  String[] str = {"多情自古空余恨，好梦由来最易醒。",  "春宵一刻值千金，花有清香月有阴。",  "有缘千里来相会，三笑徒然当一痴。",  "天长地久有时尽，此恨绵绵无绝期。",  "娶妻莫恨无良媒，书中自有颜如玉。"};    // 产生一个[1-5]的随机整数,根据随机数将str的值赋给strArr，  // 如若产生1，就将str[0]赋值给strArr[0],依次类推。  **for**(**int** i = 0; i < strArr.length; i ++) {  **int** x = (**int**) (Math.*random*()\*5+1);  strArr[i] = str[x - 1];  }  **for**(**int** i = 0; i < strArr.length; i ++) {  System.***out***.println(strArr[i]);  } |

## Foreach

|  |
| --- |
| **public** **class** ForEachDemo {  **public** **static** **void** main(String[] args) {  String[] str = {"多情自古空余恨，好梦由来最易醒。",  "春宵一刻值千金，花有清香月有阴。",  "有缘千里来相会，三笑徒然当一痴。",  "天长地久有时尽，此恨绵绵无绝期。",  "娶妻莫恨无良媒，书中自有颜如玉。"};  // 数组所装载数据的数据类型 一个变量（合法的标识符） 数组  // 一般只用来做遍历操作，遍历过程中不推荐修改数组的值，修改不起作用  **for**(String var:str){  var = "凭君莫问封侯事，一将功成万骨枯。";  }  **for**(String var:str){  System.***out***.println(var);  }  }  } |

不可变长度的数组

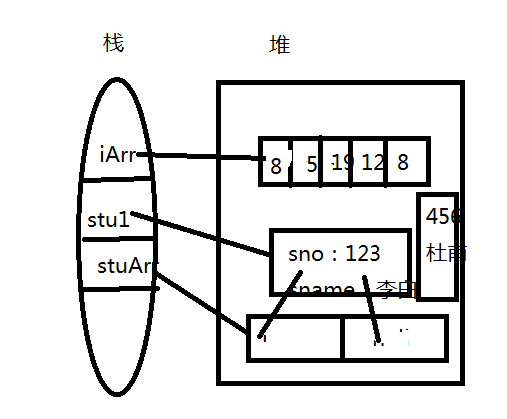
|  |
| --- |
| **public** **class** ImmutableLength {  **public** **static** **void** main(String[] args) {  **int**[] iArr = {1,2,3};  **int**[] iArr1 = **new** **int**[4];  iArr1 = iArr;  // 长度变化的假象，实际上在内存中一开始为iArr1分配的空间并没有发生变化  // 只是没有了指向，在下次gc回收时，会将其回收  System.***out***.println(iArr1.length);  }  } |

### 算法

|  |
| --- |
| **public** **class** AlgorithmDemo {  **static** **int**[] *iArr* = {3,50,30,80,33,61,45};  // 应用程序入口  **public** **static** **void** main(String[] args) {  // 方法调用，会执行该方法对应的方法体，执行完在回到调用的位置  *maxValue*();  System.***out***.println("上面的方法执行完后，执行这里");  }  } |
| // void 无返回值 maxValue方法名 ()内一般用来传参 {}内部方法体  // 如果有static修饰，在加载类时，先加载static修饰的东西，  // 在加载非static修饰的东西  **public** **static** **void** maxValue(){  **int** maxValue = *iArr*[0];  **for**(**int** i = 1;i < *iArr*.length; i++) {  **if** (maxValue < *iArr*[i]){  maxValue = *iArr*[i];  }  }  System.***out***.println(maxValue);  } |
| /\*  \* 冒泡排序（从找最大的开始）  \* 数组相邻元素，两两比较，若前面的大于后面的，则交换他们的位置  \* 每一轮比较会找到剩余最大的元素排在剩余剩余的后面  \* 3,50,30,80,33,61,45  \* 3,30,50,80,33,61,45  \* 3,30,50,33,80,61,45  \* 3,30,50,33,61,80,45  \* 3,30,50,33,61,45,80  \*  \* 3,30,33,50,61,45,80  \* 3,30,33,50,45,61,80  \* ....  \*/  **public** **static** **void** bubbleSort(){  // 多少轮比较可以将数组排序完成一共需要比较iArr.length-1轮才能排序完成  **for**(**int** i = *iArr*.length-1; i > 0 ; i--) {  // 每一轮比较的次数  **for**(**int** j=0; j < i; j++) {  **if** (*iArr*[j] > *iArr*[j+1]){  **int** temp = *iArr*[j];  *iArr*[j] = *iArr*[j+1];  *iArr*[j+1] = temp;  }  }  }  **for**(**int** iVal:*iArr*) {  System.***out***.print(iVal + " ");  }  } |
| /\*  \* 选择排序（从小到大找）  \* 3,50,90,80,33,61,45  \* 3,33,90,80,50,61,45  \* 3,33,80,90,50,61,45  \* 3,33,50,90,80,61,45  \* 3,33,45,90,80,61,50  \* ...  \*/    // i=0  // j=1 j<6 3 50  // j=2 j<6 3 90  // j=3 j<6 3 80  // j=4 j<6 3 33  // j=5 j<6 3 61  // j=6 j<6 3 45  **public** **static** **void** selectSort(){  // 多少轮比较，比较6轮，便可以实现排序  **for**(**int** i = 0; i < *iArr*.length-1; i++) {  // 每一轮比较多少次  **for**(**int** j = i+1; j < *iArr*.length; j ++) {  **if**(*iArr*[i] >*iArr*[j]) {  **int** temp = *iArr*[i];  *iArr*[i] = *iArr*[j];  *iArr*[j] = temp;  }  }  }    **for**(**int** iVal:*iArr*) {  System.***out***.print(iVal + " ");  }  } |
| /\*\*  \* 快速排序（改进的冒泡排序）  \* 70,50,3,80,33,61,45  \* 以数组中的某个值为参考，使其左边的值都小于他，右边的值都大于他，  \* 一般我们会使用第一个值作为参考，依次类推  \* 70 中间值70  \* 45,50,3,80,33,61,70  \* 45,50,3,70,33,61,80  \* 45,50,3,61,33,70,80  \* 中间值45  \* 33,50,3,61,45,70,80  \* 33,45,3,61,50,70,80  \* 33,3,45,61,50,70,80  \* 中间值3  \* 3,33,45,61,50,70,80  \* 中间值61  \* 3,33,45,50,61,70,80  \* 当两头比较的下标相遇时，就结束了  \*/ |

数组内存分析

|  |
| --- |
| **int**[] iArr = **new** **int**[5];  **for**(**int** i = 0; i < iArr.length; i ++) {  iArr[i] = (**int**) (Math.*random*()\*20);  }    **for**(**int** iVal:iArr){  System.***out***.println(iVal);  }    // 創建了一个学生对象，学生学号为123，姓名为李白  Student stu1 = **new** Student();  stu1.sno="123";  stu1.sname="李白";  // 創建了一个学生对象，学生学号为456，姓名为杜甫  Student stu2 = **new** Student();  stu2.sno="456";  stu2.sname="杜甫";      Student[] stuArr = **new** Student[2];  // null  stuArr[0] = stu1;  stuArr[1] = stu1; |



多维数组

|  |
| --- |
| **public** **class** MultiArray {  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **int**[] iArr = **new** **int**[2];  **int**[][] mulitArr1 = **new** **int**[2][];    **for**(**int** i = 0; i < mulitArr1.length; i++){  System.***out***.println(mulitArr1[i]);// null  //  If(null != mulitArr1[i]){  // 遍历第二维  }  System.out.println(mulitArr1[i].length);// nullpointerexception  }    **int**[][] mulitArr = **new** **int**[2][10];    **for**(**int** i = 0; i < mulitArr.length; i ++) {  **for**(**int** j = 0; j < mulitArr[i].length; j++) {  System.***out***.println(mulitArr[i][j]);  }  }    System.***out***.println("=========");  mulitArr1[0] = **new** **int**[5];  mulitArr1[1] = **new** **int**[10];  **for**(**int** i = 0; i < mulitArr1.length; i ++) {  **for**(**int** j = 0; j < mulitArr1[i].length; j++) {  mulitArr1[i][j] = 10;  System.***out***.println(mulitArr1[i][j]);  }  }  }  } |

#### 断点调试

执行到方法体内，快捷键F5

执行下一行代码，快捷键F6

返回到上次操作代码处，快捷键F7，一般对应与f5，f5进入方法，f7从方法出来，回到调用处的下一行

跳到下一个断点，快捷键F8

#### 作业：

1、定义一个20\*5的二维数组，用来存储某班级20位学员的5门课的成绩；这5门课，按存储顺序依次为：core C++，coreJava，Servlet，JSP和EJB。

（1）循环给二维数组的每一个元素赋0~100之间的随机整数。

（2）按照列表的方式输出这些学员的每门课程的成绩。

（3）要求编写程序求每个学员的总分，将其保留在另外一个一维数组中。

（4）要求编写程序求所有学员的某门课程的平均分。

2、验证哥德巴赫猜想：大于6的偶数能够表示成两个素数之和

3、使用数组实现以下功能，要求思路清晰，逻辑严谨。代码输出以下4\*4方阵，

\* \* \* \*

\* \* \* \*

\* \* \* \*

\* \* \* \* ， 之后接收键盘输入2个数字，用逗号隔开，如3,3，则输出方阵对应位置发生变化，如下所示

\* \* \* \*

\* \* \* \*

\* \* $ \*

\* \* \* \*,再输入1，3,如下图

\* \* $ \*

\* \* \* \*

\* \* $ \*

\* \* \* \*

循环输入变化操作，直到出现如下对角线的情形时即对角线的4个\*全部变为$，结束循环，停止操作

$ \* \* \*

\* $ \* \*

\* \* $ \*

\* \* \* $

提示：输入1,3可以通过使用String的split方法，对该该数据进行分割，得到string在数组，第一个元素为1，第二个元素为3

4、8皇后问题，在8\*8的棋盘上，某皇后所在列与行，对角线均不能有其他皇后存在，一共有多少种排列方法

5、10\*10矩阵，如上第三题，实现五子棋的功能，白子#表示，黑子$表示，\*表示棋盘