1. 写出一维数组初始化的两种方式

2. 谈谈你对数组的理解（提示：定义、分类、特点、使用场景等）

3. 自己定义一个一维 int 型数组，要求输出数组的最大值和最小值。

4. 不同类型的一维数组元素的默认初始化值各是多少

   整型 0

   浮点型 0.0

   字符型 0

   布尔型 false

```java
int[] arr = new int[]{1, 5, 6};
int[] arr1 = new int[5];
/*2. 谈谈你对数组的理解（提示：定义、分类、特点、使用场景等）
*/
/*3. 自己定义一个一维int型数组，要求输出数组的最大值和最小值。*/
int[] arr = new int [] {1, 2, 3, 4, 5};
int max =0;
int min =0;
for(int i = 0;i < arr.length-1;i++){
if(max < arr[i]){
max = i;
}else if(min > arr[i]){
min = i;
}
}
}
```

```java
/*3. 自己定义一个一维int型数组，要求输出数组的最大值和最小值。*/
int[] arr = new int [] {1, 2, 3, 4, 5};
int max = arr[0];
int min = arr[0];

for(int i = 1;i < arr.length;i++){
    if(max < arr[i]){
        max = arr[i];
    }

    if(min > arr[i]){
        min = arr[i];
    }
}
```

4. 不同类型的一维数组元素的默认初始化值各是多少

整型 0

浮点型 0.0

字符型 '\u0000'　0　'0'

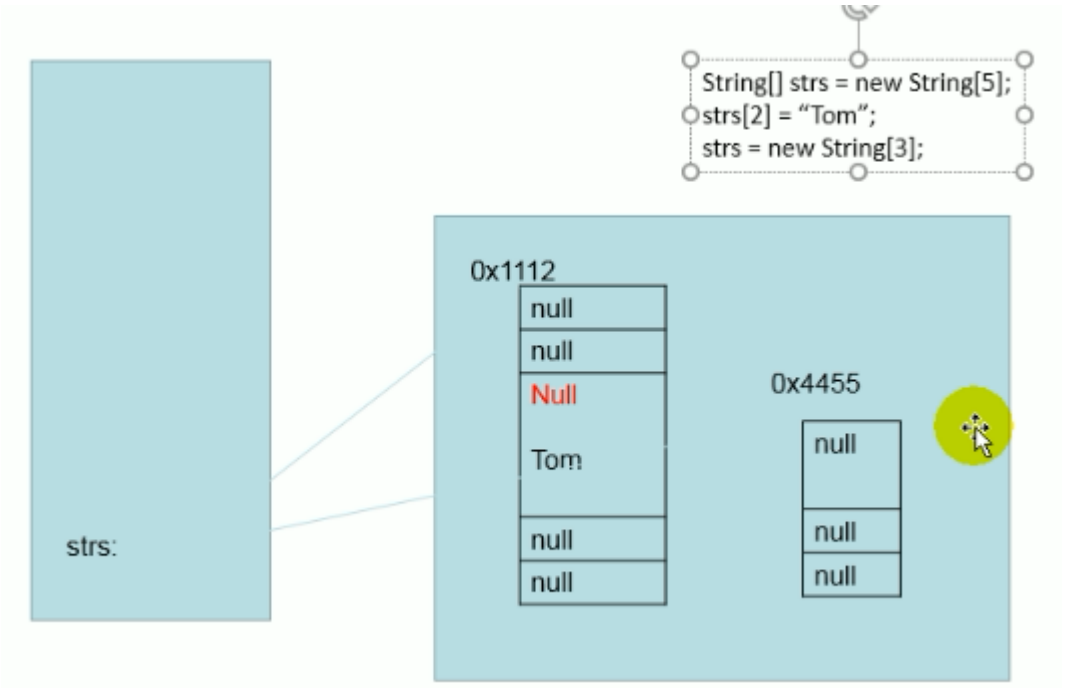
布尔型 false

引用类型 null

5. 一维数组的内存解析：

String[] strs = new String[5];

strs[2] = "Tom";



1.数组的说明：

1. 定义：是多个相同类型数据一定顺序排列的集合，并使用一个名字命名，并通过编号的方式对这些数据进行统一管理。
* 2. 相关概念：
* ① 元素
* ② 下标、角标、索引
* ③ 数组名
* ④ 数组的长度
* 3. 数组，属于引用数据类型。数组的元素，可以是基本数据类型，也可以是引用数据类型
* 4. 数组的分类：
* ①照维度：一维数组、二维数组、三维数组、...
* ②照元素的类型：基本数据类型变量的数组、引用数据类型变量的数组
*
* 5.数组一旦初始化完成，则数组的长度就确定了。长度一旦确定，就不可更改。
*
* 6.数组中的元素是在内存空间中连续存储的。

2.一维数组的声明与初始化

正确的方式：

```
int[] arr1;//声明
//静态初始化:数组的初始化和数组元素的赋值同时进行。
arr1 = new int[]{1,2,3,4};
//合并声明和初始化
int[] arr2 = new int[]{4,5,6,7};
//或：int arr2[] = new int[]{4,5,6,7};
//      或：int[] arr2 = {4,5,6,7}; //类型推断

//动态初始化：数组的初始化和数组元素的赋值分开进行。
String[] arr3 = new String[3];
```

错误的声明方式：

```
//        int[] arr4 = new int[4]{1,2,3,4};
//        int[4] arr4 = new int[]{1,2,3,4};
```

3.一维数组元素的引用：

通过角标的方式。角标从0开始，到长度 - 1结束。

4.数组的属性：使用length属性调用

```
System.out.println(arr3.length);//5
System.out.println(arr2.length);//4
```

5.一维数组的遍历

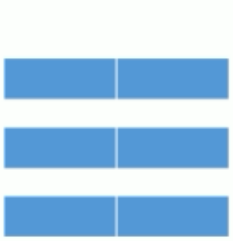
```
for(int i = 0;i < arr2.length;i++){
    System.out.println(arr2[i]);
}
```

6.一维数组元素的默认初始化值

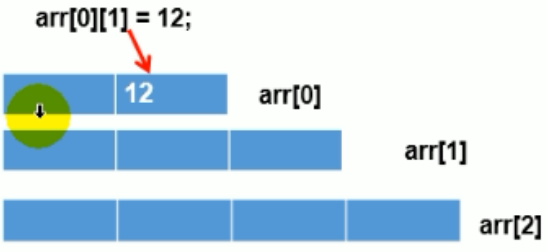
```
     整型：0
*    浮点型：0.0
*    字符型：0
*    布尔型：false
*    引用类型：null
*   此时的规范与后面类的属性的默认初始化值一致！
```

3.3 多维数组的使用

尚硅谷



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




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

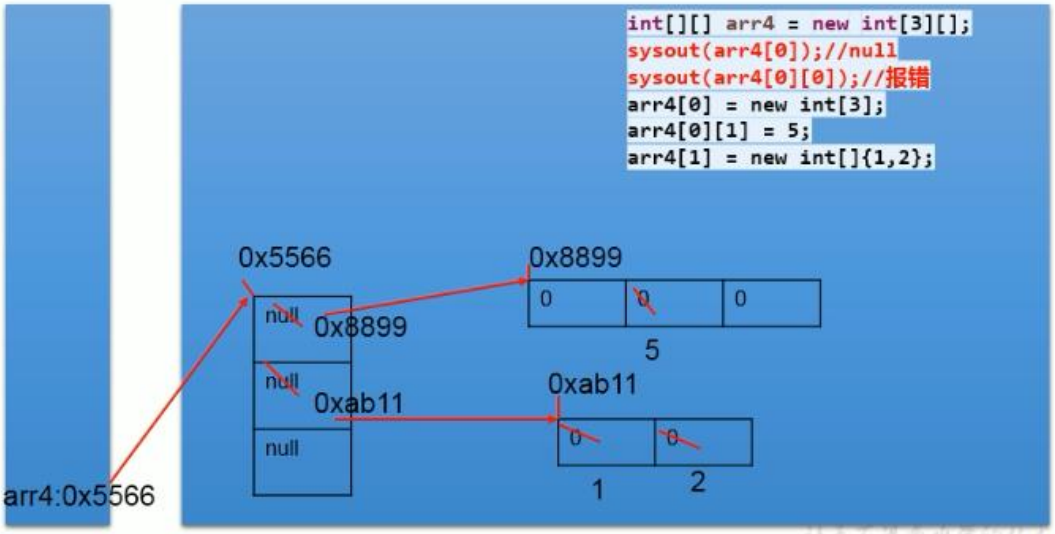
3.二维数组元素的默认初始化

以动态初始化方式来解释。

二维数组的外层元素的初始化值为：地址值 或 null
二维数组的内层元素的初始化值为：

① 如果外层元素初始化值为地址值时，此时内层元素默认初始化值与一维数组不同类型的默认初始化值相同。
即：整型：0；浮点型：0.0 ； 字符型：'\u0000' ；布尔型：false；引用类型：null
② 如果外层元素初始化值为null,则内层元素还不存在。试图调用的话，会报NullPointerException

补充：查看二维数组变量的值，只要初始化过二维数组，存储都是地址值。
比如如下问题的：arr1,arr2,..



3.4 数组中涉及到的常见算法

3.4 数组中涉及的常见算法

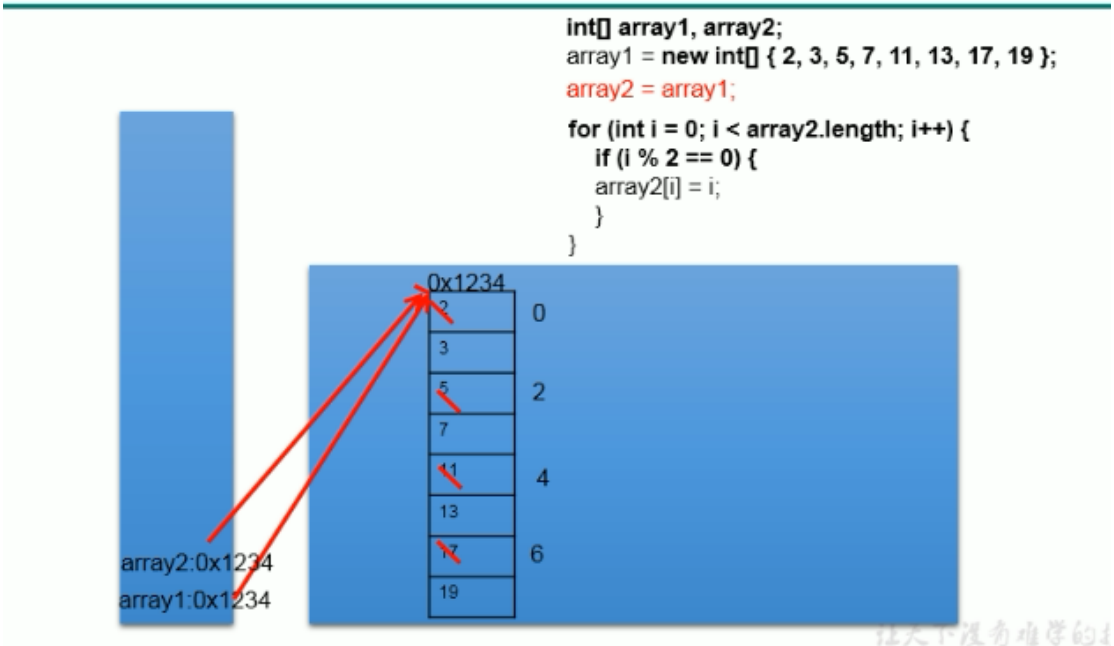
1.求数值型数组中元素的最大值、最小值、平均数、总和等

2.数组的复制、反转、查找(线性查找、二分法查找)

3.数组元素的排序算法

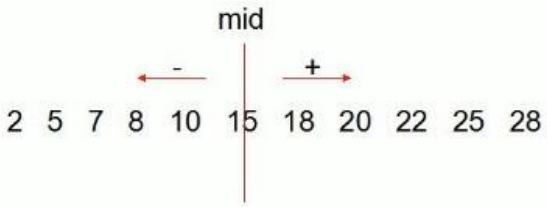
```java
package com.atguigu.exer;
/*
 * 定义一个int型的一维数组，包含10个元素，分别赋一些随机整数，然后求出所有元素的最大值，
 * 最小值，平均值，和值，并输出出来。
 * 要求：所有随机数都是两位数。[10,99]
 *
 *
 *  Math.random() : [0.0,1)
 *
 *  Math.random() * 90 : [0.0,90.0)
 *  (int)(Math.random() * 90) : [0,89]
 *  (int)(Math.random() * 90) + 10 : [10,99]
 *
 *  公式：获取[a,b]范围的随机数：(int)(Math.random() * (b - a + 1)) + a
 *
 *  比如：生成[1,30]范围的随机整数：
 *  Math.random() : [0.0,1)  ->
 *  Math.random() * 30 : [0.0,30.0) ->
 *  (int)(Math.random() * 30) : [0,29] ->
 *  (int)(Math.random() * 30) + 1 : [1,30]
 *
 */
```

3.4 数组中涉及到的常见算法

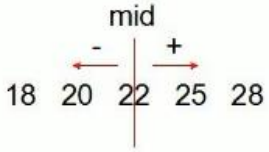


3.4 数组中涉及到的常见算法：二分法查找算法

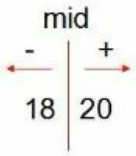
step 1:



step 2:



step 3:



让天下没有难学的技术

3.4 数组中涉及到的常见算法：二分法查找算法

```java
//二分法查找：要求此数组必须是有序的。
int[] arr3 = new int[]{-99,-54,-2,0,2,33,43,256,999};
boolean isFlag = true;
int number = 256;
//int number = 25;
int head = 0;//首索引位置
int end = arr3.length - 1;//尾索引位置
while(head <= end){
    int middle = (head + end) / 2;
    if(arr3[middle] == number){
        System.out.println("找到指定的元素，索引为：" + middle);
        isFlag = false;
        break;
    }else if(arr3[middle] > number){
        end = middle - 1;
    }else{//arr3[middle] < number
        head = middle + 1;
    }
}

if(isFlag){
    System.out.println("未找打指定的元素");
}
```

让天下没有

3.4 数组中涉及到的常见算法：算法介绍

尚硅谷

算法的5大特征

输入（Input）	有0个或多个输入数据，这些输入必须有清楚的描述和定义
输出（Output）	至少有1个或多个输出结果，不可以没有输出结果
有穷性（有限性，Finiteness）	算法在有限的步骤之后会自动结束而不会无限循环，并且每一个步骤可以在可接受的时间内完成
确定性（明确性，Definiteness）	算法中的每一步都有确定的含义，不会出现二义性
可行性（有效性，Effectiveness）	算法的每一步都是清楚且可行的，能让用户用纸笔计算而求出答案

说明：满足确定性的算法也称为：确定性算法。现在人们也关注更广泛的概念，例如考虑各种非确定性的算法，如并行算法、概率算法等。另外，人们也关注并不要求终止的计算描述，这种描述有时被称为过程（procedure）。

让天下没有难学的技术

3.5 Arrays工具类的使用

尚硅谷

- java.util.Arrays类即为操作数组的工具类，包含了用来操作数组（比如排序和搜索）的各种方法。

1	boolean equals(int[] a,int[] b)	判断两个数组是否相等。一系列重载的方法。
2	String toString(int[] a)	输出数组信息。一系列重载的方法。
3	void fill(int[] a,int val)	将指定值填充到数组之中。一系列重载的方法。
4	void sort(int[] a)	对数组进行排序。一系列重载的方法。
5	int binarySearch(int[] a,int key)	对排序后的数组进行二分法检索指定的值。

让天下没有难学的技术

3.4 数组中涉及到的常见算法：排序算法

尚硅谷

- 选择排序
  - 直接选择排序、堆排序
- 交换排序
  - 冒泡排序、快速排序
- 插入排序
  - 直接插入排序、折半插入排序、Shell排序
- 归并排序
- 桶式排序
- 基数排序

详细操作，见《附录：尚硅谷_宋红康_排序算法.pptx》

让天下没有难学的技术

3.4 数组中涉及到的常见算法：排序算法

尚硅谷

快速排序

介绍：

快速排序通常明显比同为O(nlogn)的其他算法更快，因此常被采用，而且快排采用了分治法的思想，所以在很多笔试面试中能经常看到快排的影子。可见掌握快排的重要性。

快速排序（Quick Sort）由图灵奖获得者Tony Hoare发明，被列为20世纪十大算法之一，是迄今为止所有内排序算法中速度最快的一种。冒泡排序的升级版，交换排序的一种。快速排序的时间复杂度为O(nlog(n))。

3.6 数组使用中的常见异常

尚硅谷

编译时，不报错！！

数组脚标越界异常(ArrayIndexOutOfBoundsException)

```
int[] arr = new int[2];
System.out.println(arr[2]);
```

访问到了数组中的不存在的脚标时发生。

空指针异常(NullPointerException)

```
int[] arr = null;
System.out.println(arr[0]);
```

arr引用没有指向实体，却在操作实体中的元素时。