解题思路

1、 画图分析需求
2、 技术选型（用什么技术来开发设计）
3、 像结构、会出现的问题、技术难点
4、 构思解题方法（这样解决）
5、 开始编写代码

数组

案例统览

```
package com.haoyu;

import java.util.Arrays;

/*
* 2，一个城市有10个街道
*          每个街道有一个街道号 1001-1010
*    一个街道有3百户人家
*    每个人家一个门牌号0-++
*/
public class Demo12 {

     public static void main(String[] args) {
          City city=new City();
          city.initCity();
          Street[] streets=city.getStreets();
          for(Street street:streets) {
               Home[] homes=street.getHomes();
```

```
                System.out.println("-------
>"+street.blockId);
                for(Home home:homes) {
                     System.out.println(home.doorId);
                }
           }
     }

}
//城市
class City{
     Street[] streets=new Street[10];
     //初始化城市功能
     //init 初始化
     void initCity() {
          for(int i=0;i<10;i++) {
                Street street=new Street();
                streets[i]=street.initStreet(i+1,
this);
           }
     }

     public Street[] getStreets() {
          return streets;
     }

     public String toString() {
          return "City [streets=" +
Arrays.toString(streets) + "]";
     }
}
//街道
class Street{
     String blockId;
     Home[] homes=new Home[300];
     //
     City masterCity;
     //街道的初始化
     Street initStreet(int id,City masterCity) {
          this.blockId=IdGenerator.streetId(id);
```

```
            this.masterCity=masterCity;
            //每个街区创建300个家
            homesInit();
            return this;
        }
        //
        void homesInit() {
            for(int i=0;i<300;i++) {
                Home home=new Home();

        homes[i]=home.initHome(i+1,this);//this--new
Street
            }
        }
        City getStreetMasterCity() {
            return this.masterCity;
        }
        public Home[] getHomes() {
            return homes;
        }
        public String toString() {
            return "Street [blockId=" + blockId + ",
homes=" + Arrays.toString(homes) + ", masterCity=" +
masterCity + "]";
        }
}
//家
class Home{
        String doorId;
        //master 主人
        Street masterStreet;
        //
        City masterCity;
        //TODO 思考为什么只传两个
        Home initHome(int id,Street masterStreet) {
            //健壮性判断
            if(0!=id&&null!=masterStreet) {
                this.doorId=IdGenerator.doorId(id);
                this.masterStreet=masterStreet;
//          this.masterCity=masterStreet.masterCity;
```

```
        this.masterCity=masterStreet.getStreetMasterCity(
);//oop
                    return this;
               }else {
                    return null;
               }
          }
          public String toString() {
               return "Home [doorId=" + doorId + ",
masterStreet=" + masterStreet + ", masterCity=" +
masterCity + "]";
          }
}
class IdGenerator{
          //思考什么是static
          static String streetId(int id) {
               int streetId=id+1000;
               return ""+streetId;//int+string=string -->
type
          }
          static String doorId(int id) {
               return ""+id;
          }
}
```

1，创建数组
2，初始化数组
3，排序
4，增强 for 循环
5，赋值数组
6，二维数组
7，Arrays

数组的定义：

数组是一个固定长度的，包含了相同类型数据的容器

```
//数组是一个固定长度的，包含了相同类型数据的容器
final int length=10;
//相同类型      基本数据类型     类类型
byte[] a;
short[] a1;
int[] a2;
long[] a3;
float[] a4;
double[] a5;
boolean[] a6;
char[] a7;

Cat[] a8;

int[] arrays=new int[length];
```

创建数组

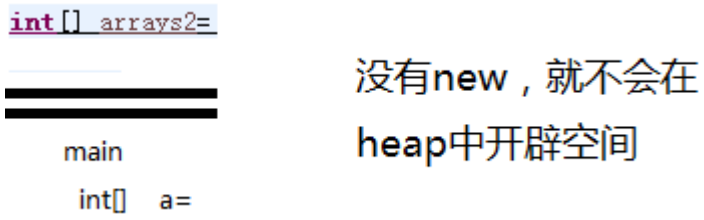
步骤 **1** :

声明数组

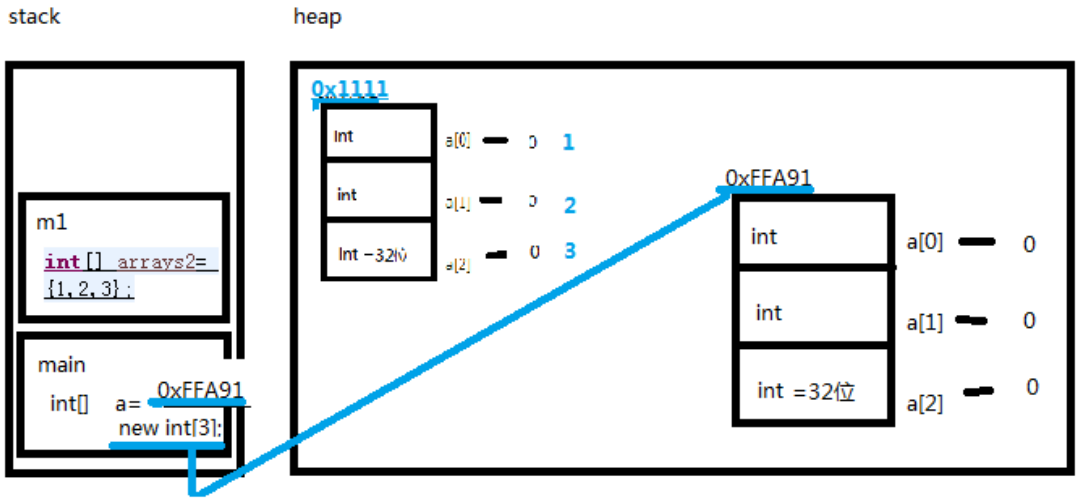
int[] a; 声明了一个数组变量。
[]表示该变量是一个数组
int 表示数组里的每一个元素都是一个整数
a 是变量名
但是，仅仅是这一句**声明，不会创建数组**

有时候也会写成 int a[]; 没有任何区别，就是你看哪种顺眼的问题

```
public class HelloWorld {
    public static void main(String[] args) {
        // 声明一个数组
        int[] a;
    }
}
```



对于要 new 出来的原理如下




```
main(){
      int[] arrays=new int[length];
      m1();
}
static void m1() {
      int[] arrays2= {1,2,3};
}
```

步骤 **2** :

创建数组

创建数组的时候，要指明数组的长度。

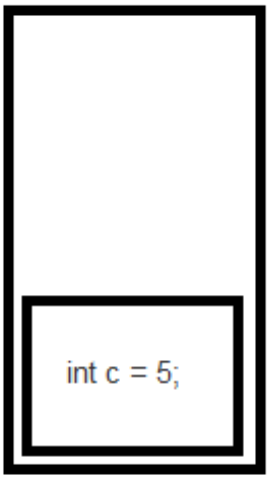
new int[5]，只要出现了 new 关键字，一定要对其赋予空间大小

引用概念：

如果变量代表一个数组，比如 a,我们把 a 叫做**引用** ——堆空间开辟的数组空间起始地址

与基本类型不同

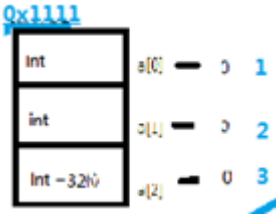
int c = 5; 这叫给 c **赋值**为 5



声明一个引用 int[] a;

a = new int[5];

让 a 这个引用，**指向**数组



Or

0	1	2	3	4

```
public class HelloWorld {
    public static void main(String[] args) {
```

```
        //声明一个引用
        int[] a;
        //创建一个长度是 5 的数组，并且使用引用 a 指向该数组
        a = new int[5];

        int[] b = new int[5]; //声明的同时，指向一个数组

    }
}
```

步骤 **3** :

访问数组

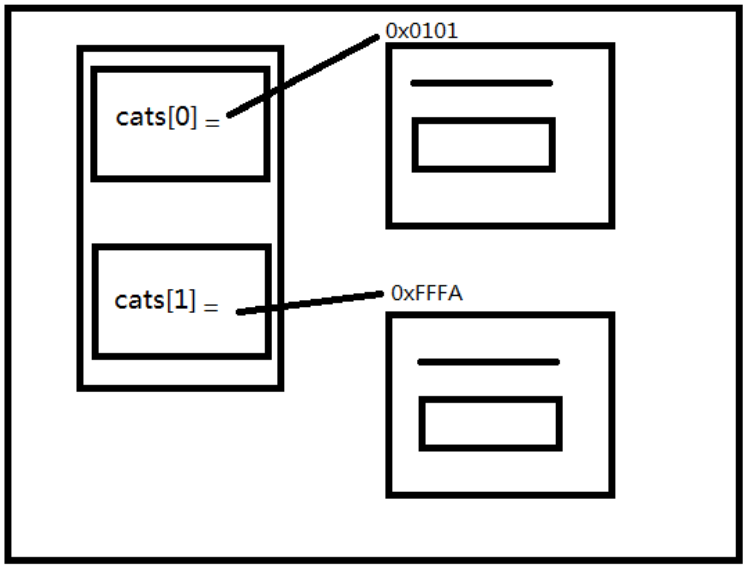
数组下标**从 0 开始**
下标 0，代表数组里的第一个数

int[] a={1,2,3};

角标值是：0,1,2 ——— 数组元素的身份证号
代表顺序：1,2,3

```
public class HelloWorld {
    public static void main(String[] args) {

        int[] a;
        a = new int[5];

        a[0]= 1;  //下标0，代表数组里的第一个数
        a[1]= 2;
        a[2]= 3;
        a[3]= 4;
        a[4]= 5;

         int[] b=a;
```

```
            b[0]=8;

            System.out.println(a[0]);//1  8

            Cat[] cats=new Cat[2];
            System.out.println(cats[0]);//空   null

    }
}
```




```
Cat[] cats=new Cat[2];
```

步骤 4 :

数组长度

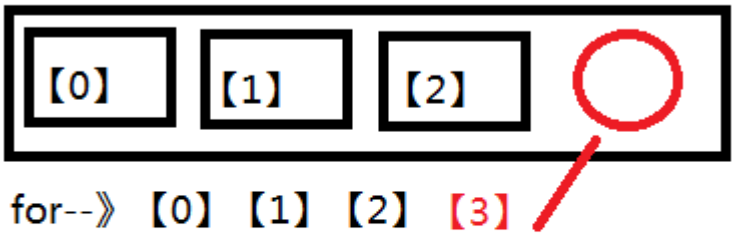
.length 属性用于访问一个数组的长度

数组访问下标范围是 0 到长度-1

一旦超过这个范围,就会产生数组下标越界异常

```
5
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
        at HelloWorld.main(HelloWorld.java:9)
```

int[] a=new int[3];



for--》【0】【1】【2】【3】

对于没有的元素，强行角标访问，会报错

indexoutofboundsexception

数组角标 值 超出定义长度

```
public class HelloWorld {
    public static void main(String[] args) {
        int[] a;
        a = new int[5];

        System.out.println(a.length); //打印数组的长度

        a[4]=100; //下标 4，实质上是“第 5 个”，即最后一个
        a[5]=101; //下标 5，实质上是“第 6 个”，超出范围 ,产生数组下标越界异常
    }
}
```

步骤 **5** :

练习-数组最小值

首先创建一个长度是 5 的数组
然后给数组的每一位赋予随机整数
通过 for 循环，遍历数组，找出最小的一个值出来

0-100 的 随机整数的获取办法有多种，下面是参考办法之一:

```
(int) (Math.random() * 100)
```

Math.random() 会得到一个 0-1 之间的随机浮点数，然后乘以 100，并强转为整型即可。

```
public class HelloWorld {
    public static void main(String[] args) {
        int[] a = new int[5];
        a[0] = (int) (Math.random() * 100);
        a[1] = (int) (Math.random() * 100);
        a[2] = (int) (Math.random() * 100);
        a[3] = (int) (Math.random() * 100);
        a[4] = (int) (Math.random() * 100);

        System.out.println("数组中的各个随机数是:");
        for (int i = 0; i < a.length; i++)
            System.out.println(a[i]);

        System.out.println("本练习的目的是，找出最小的一个值: ");
    }
}
```

方法一（推荐）

```
            int length=5;
            int[] arrays=new int[length];//{0,0,0,0,0}
arrays=0xAAAA01
            for(int i=0;i<arrays.length;i++) {
                arrays[i]=(int) (Math.random() * 100);
            }
            for(int data:arrays) {
                System.out.print(" "+data);
            }
            System.out.println();

            Arrays.sort(arrays);
            for(int data:arrays) {
                System.out.print(" "+data);
            }
            System.out.println();
            System.out.println(arrays[0]);
```

方法二

```
        int length=5;
        int[] arrays=new int[length];//{0,0,0,0,0}
arrays=0xAAAA01
        for(int i=0;i<arrays.length;i++) {
            arrays[i]=(int) (Math.random() * 100);
        }
        for(int data:arrays) {
            System.out.print(" "+data);
        }
        System.out.println();
        int min=arrays[0];
        int count=0;
        for(;;count++) {//while(true)
            if(count==arrays.length) {//6次循环
                break;
            }
            if(min>arrays[count]) {
                min=arrays[count];
            }
        }
        System.out.println("min--->"+min);
```

初始化数组

步骤 1：分配空间与赋值分步进行

步骤 2：分配空间，同时赋值

步骤 **1** :

分配空间与赋值分步进行

分配空间与赋值分步进行

```
public class HelloWorld {
    public static void main(String[] args) {
        int[] a = new int[5]; //分配了长度是 5 的数组，但是没有赋值

        //没有赋值，那么就会使用默认值
```

```
        //作为 int 类型的数组，默认值是 0
        System.out.println(a[0]);

        //进行赋值

        a[0] = 100;
        a[1] = 101;
        a[2] = 103;
        a[3] = 120;
        a[4] = 140;
    }
}
```

步骤 **2** :

分配空间，同时赋值

分配空间，同时赋值

```
public class HelloWorld {
    public static void main(String[] args) {
        //写法一： 分配空间同时赋值
        int[] a = new int[]{100,102,444,836,3236};

        //写法二： 省略了 new int[],效果一样
        int[] b = {100,102,444,836,3236};

        //写法三：同时分配空间，和指定内容
        //在这个例子里，长度是 3，内容是 5 个，产生矛盾了
        //所以如果指定了数组的内容，就不能同时设置数组的长度
        int[] c = new int[3]{100,102,444,836,3236};

    }
}
```

步骤 **3** :

练习-数组反转

首先创建一个长度是 6 的数组,并填充随机数。
使用 for 循环，对这个数组实现反转效果

例如，1 2 3 4 5 6—》6 5 4 3 2 1

```
int[] a= {1,2,3,4,5,6};
for(int i=a.length-1;i>=0;i--) {
    System.out.println(a[i]);
}
```

基本排序

步骤 1 :

选择法排序

选择法排序的思路:
把第一位和其他所有的进行比较，只要比第一位小的，就换到第一个位置来。
比较完后，**第一位就是最小的** 。
然后再从**第二位**和剩余的其他所有进行比较，只要比第二位小，就换到第二个位置来 。
比较完后，**第二位就是第二小的** 。
以此类推。

未排序	18	62	68	82	65	9
第一个位置	9	62	68	82	65	18
第二个位置	9	18	68	82	65	62
第三个位置	9	18	62	82	68	65
第四个位置	9	18	62	65	82	68
第五个位置	9	18	62	65	68	82

```
public class HelloWorld {
    public static void main(String[] args) {
        int a [] = new int[]{18,62,68,82,65,9};
```

```
//排序前，先把内容打印出来
for (int i = 0; i < a.length; i++) {
    System.out.print(a[i] + " ");
}
System.out.println(" ");
//选择法排序

//第一步： 把第一位和其他所有位进行比较
//如果发现其他位置的数据比第一位小，就进行交换

for (int i = 1; i < a.length; i++) {
    if(a[i]<a[0]){
        int temp = a[0];
        a[0] = a[i];
        a[i] = temp;
    }
}
//把内容打印出来
//可以发现，最小的一个数，到了最前面
for (int i = 0; i < a.length; i++) {
    System.out.print(a[i] + " ");
}
System.out.println(" ");

//第二步： 把第二位的和剩下的所有位进行比较
for (int i = 2; i < a.length; i++) {
    if(a[i]<a[1]){
        int temp = a[1];
        a[1] = a[i];
        a[i] = temp;
    }
}
//把内容打印出来
//可以发现，倒数第二小的数，到了第二个位置
for (int i = 0; i < a.length; i++) {
    System.out.print(a[i] + " ");
}
System.out.println(" ");

//可以发现一个规律
//移动的位置是从 0 逐渐增加的
//所以可以在外面套一层循环

for (int j = 0; j < a.length-1; j++) {

    for (int i = j+1; i < a.length; i++) {
```

```
            if(a[i]<a[j]){

                int temp = a[j];
                a[j] = a[i];
                a[i] = temp;
            }
        }
    }

    //把内容打印出来
    for (int i = 0; i < a.length; i++) {
        System.out.print(a[i] + " ");
    }
    System.out.println(" ");
  }
}
```

步骤 **2** :

冒泡法排序

冒泡法排序的思路:

第一步:从第一位开始,把相邻两位进行比较

如果发现前面的比后面的大,就把大的数据交换在后面,循环比较完毕后,**最后一位就是最大的**

第二步: 再来一次,只不过不用比较最后一位

以此类推

未排序	18	62	68	82	65	9
倒数第一个位置	18	62	68	65	9	82
倒数第二个位置	18	62	65	9	68	82
倒数第三个位置	18	62	9	65	68	82
倒数第四个位置	18	9	62	65	68	82
倒数第五个位置	9	18	62	65	68	82

```
public class HelloWorld {
    public static void main(String[] args) {
        int a [] = new int[]{18,62,68,82,65,9};
        //排序前,先把内容打印出来
        for (int i = 0; i < a.length; i++) {
            System.out.print(a[i] + " ");
```

```
}
System.out.println(" ");
//冒泡法排序

//第一步：从第一位开始，把相邻两位进行比较
//如果发现前面的比后面的大，就把大的数据交换在后面

for (int i = 0; i < a.length-1; i++) {
    if(a[i]>a[i+1]){
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
    }
}
//把内容打印出来
//可以发现，最大的到了最后面
for (int i = 0; i < a.length; i++) {
    System.out.print(a[i] + " ");
}
System.out.println(" ");

//第二步： 再来一次，只不过不用比较最后一位
for (int i = 0; i < a.length-2; i++) {
    if(a[i]>a[i+1]){
        int temp = a[i];
        a[i] = a[i+1];
        a[i+1] = temp;
    }
}
//把内容打印出来
//可以发现，倒数第二大的到了倒数第二个位置
for (int i = 0; i < a.length; i++) {
    System.out.print(a[i] + " ");
}
System.out.println(" ");

//可以发现一个规律
//后边界在收缩
//所以可以在外面套一层循环

for (int j = 0; j < a.length; j++) {

    for (int i = 0; i < a.length-j-1; i++) {

        if(a[i]>a[i+1]){
            int temp = a[i];
            a[i] = a[i+1];
```

```
                a[i+1] = temp;
            }
        }
    }

    //把内容打印出来
    for (int i = 0; i < a.length; i++) {
        System.out.print(a[i] + " ");
    }
    System.out.println(" ");
  }
}
```

步骤 **3** :

练习-排序

首先创建一个长度是 5 的数组,并填充随机数。

首先用选择法正排序，然后再对其使用冒泡法倒排序

注 所谓的正排序就是从小到大排序，倒排序就是从大到小排序

增强 for 循环

注：增强型 for 循环只能用来取值，却不能用来修改数组里的值

代码比较复制代码

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

        for(int i=0;i<arrays.length;i++) {
            arrays[i]+=10;//arrays[i]=arrays[i]+10;
            System.out.println(arrays[i]);
        }

        int[] arrays2= {1,2,3,4,5};

        for(int data:arrays2) {
            //data  是arrays2中每个元素的临时变量，{}
结束后就消失
            data+=10;
        }
```

```
        for(int data:arrays2) {
            System.out.println(data);
        }
```

集合中的增强 for 循环

```
class Banana{
    public Banana(String name) {//创建banana模型的实例-对象的时候，默认调用的函数就是构造函数
        this.name=name;
    }
    String name;

    public String toString() {
        return "Banana [name=" + name + "]";
    }
}
public class Demo16 {

    public static void main(String[] args) {
        List<Banana> bs=new ArrayList();
        bs.add(new Banana("b1"));
        bs.add(new Banana("b2"));
        bs.add(new Banana("b3"));
        for(Banana b:bs) {
            System.out.println(b);
        }
```

复制数组

数组的长度是不可变的，一旦分配好空间，是多长，就多长，不能增加也不能减少

步骤 1：

复制数组

把一个数组的值，复制到另一个数组中

```
System.arraycopy(src, srcPos, dest, destPos, length)
```

src: 源数组

srcPos: 从源数组复制数据的起始位置

dest: 目标数组

destPos: 复制到目标数组的起始位置

length: 复制的长度

```
public class HelloWorld {
    public static void main(String[] args) {
        int a [] = new int[]{18,62,68,82,65,9};

        int b[] = new int[3];//分配了长度是 3 的空间，但是没有赋值

        //通过数组赋值把，a 数组的前 3 位赋值到 b 数组

        //方法一:  for 循环

        for (int i = 0; i < b.length; i++) {
            b[i] = a[i];
        }

        //方法二: System.arraycopy(src, srcPos, dest, destPos, length)
        //src: 源数组
        //srcPos: 从源数组复制数据的起始位置
        //dest: 目标数组
        //destPos: 复制到目标数组的启始位置
        //length: 复制的长度
        System.arraycopy(a, 0, b, 0, 3);

        //把内容打印出来
        for (int i = 0; i < b.length; i++) {
            System.out.print(b[i] + " ");
        }
```

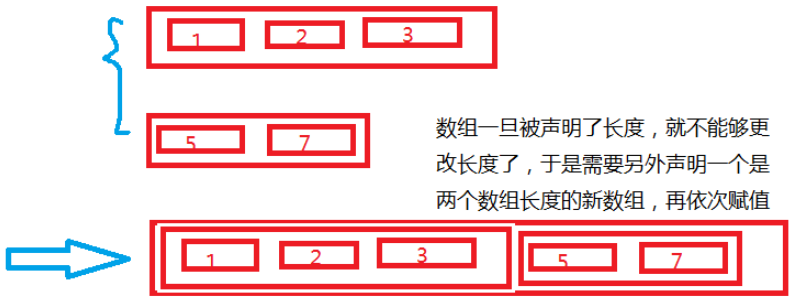
```
    }
}
```

练习合并数组

首先准备两个数组，他俩的长度是 5-10 之间的随机数，并使用随机数初始化这两个数组然后准备第三个数组，第三个数组的长度是前两个的和通过 System.arraycopy 把前两个数组合并到第三个数组中

```
数组a的内容:
24 49 23 28 26 67 89
数组b的内容:
65 70 86 77 43 10
数组c的内容:
24 49 23 28 26 67 89 65 70 86 77 43 10
```

结构思路:



案例思路

```
        int[] a= {1,2,3,4,5};//a[0]-a[4]
        int[] b= {6,7,8};
        int[] arrays=new int[a.length+b.length];
        for(int i=0;i<arrays.length;i++) {
            //copy a  a.length=5
            if(i<a.length) {
                arrays[i]=a[i];
            }else {
                //copy b
                //i=5  a[5]=b[0]
                arrays[i]=b[i-a.length];
            }
            System.out.print(arrays[i]+" ");
```

```
        }
```

二维数组

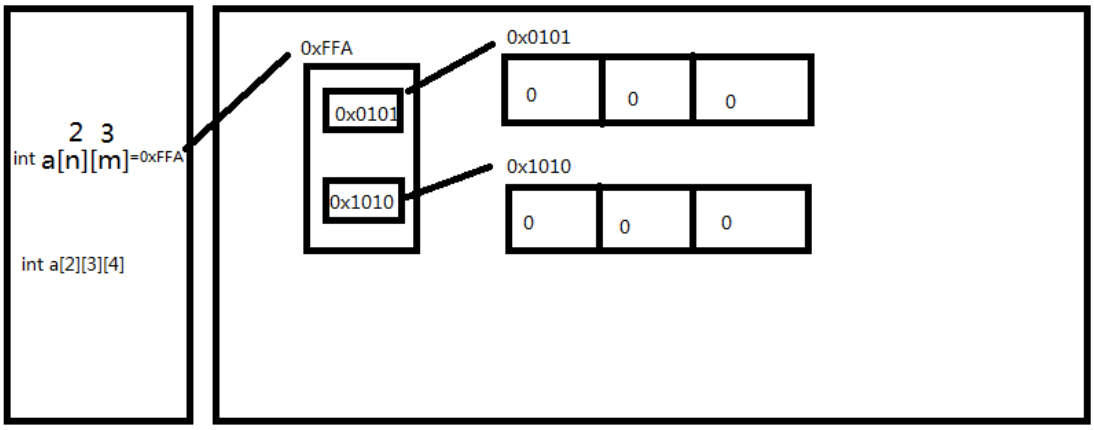
这是一个一维数组, 里面的每一个元素，都是一个基本类型 int

```
int a[] =new int[]{1,2,3,4,5};
```

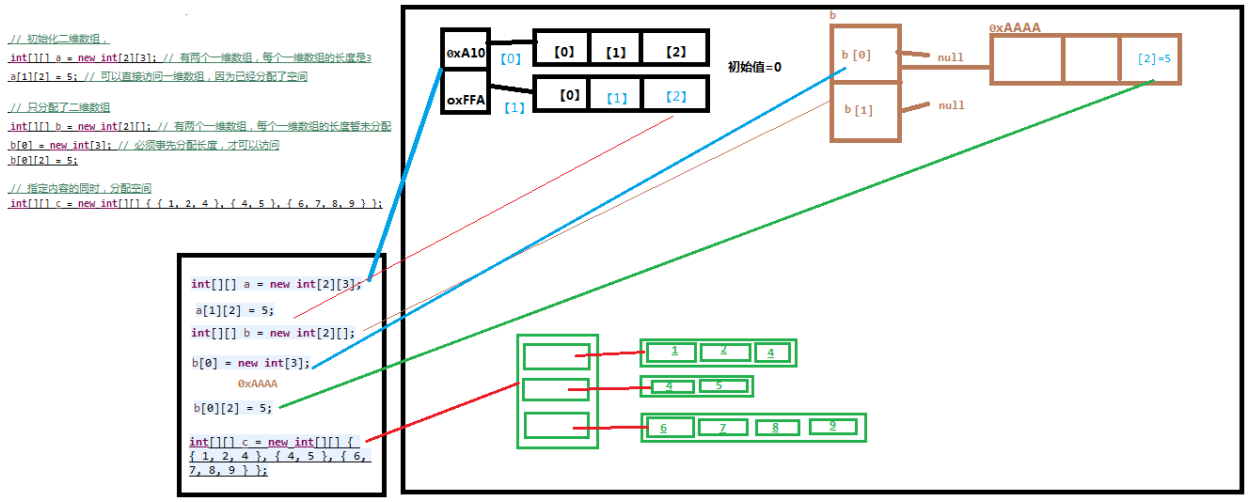
这是一个二维数组，里面的每一个元素，都是一个一维数组
所以二维数组又叫数组的数组

```
int b[][] = new int[][]{
  {1,2,3},
  {4,5,6},
  {7,8,9}
};
```

```
        /*
         1 2 3 4 5
         6 7 8 9 1
         2 3 4 5 6
         7 8 9 1 2
         */
        //二维数组
        int[][] arrays=new int[][]
{{1,2,3,4,5},{6,7,8,9,1},{2,3,4,5,6},{7,8,9,1,2}};

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



初始化二维数组




```
public class HelloWorld {
    public static void main(String[] args) {
        //初始化二维数组，
        int[][] a = new int[2][3]; //有两个一维数组，每个一维数组的长度是 3
        a[1][2] = 5;  //可以直接访问一维数组，因为已经分配了空间

        //只分配了二维数组
        int[][] b = new int[2][]; //有两个一维数组，每个一维数组的长度暂未分配
        b[0]  =new int[3]; //必须事先分配长度，才可以访问
        b[0][2] = 5;

        //指定内容的同时，分配空间
        int[][] c = new int[][]{
                {1,2,4},
                {4,5},
                {6,7,8,9}
```

```
    };

  }
}
```

练习题

定义一个 5X5 的二维数组。 然后使用随机数填充该二维数组。
找出这个二维数组里，最大的那个值，并打印出其二维坐标

0-100 的 随机整数的获取办法有多种，下面是参考办法之一:

```
(int) (Math.random() * 100)
```

Math.random() 会得到一个 0-1 之间的随机浮点数，然后乘以 100，并强转为整型即可。

```
<terminated> HelloWorld (1) [Java Application] E:\jdk\bi
60      3       44      53      7
80      35      68      42      83
81      71      57      51      96
46      79      54      27      0
59      33      89      32      14
找出来最大的是:96
其坐标是[2][4]
```

TODO

类与对象：oop 细节

对象数组

Arrays 工具类