第一章 数组的定义和访问

案例分析：

现在需要统计某公司员工的工资情况，例如计算平均工资、找到最高工资等。假设该公司有50名员工，用前面所学的知识点，程序需要声明50个变量来分别记住每位员工的工资，然后再操作，这样做回显得很麻烦，而且错误率会很高。因此我们使用**容器**来进行操作，将所有的数据存储到一个容器中，统一操作。

- **容器**：将多个数据存储到一起，每个数据称为该容器的元素
- **数组**：数据就是存储数据长度固定的容器，保证多个数据的数据类型要一致

1.1 数组的定义

1.1.1 格式一

```
数组存储的数据类型[] 数组名 = new 数组存储的数据类型[长度];
```

- 数组的存储数据类型：创建的数组容器可以存储什么数据类型
- []：表示数组
- 数组名：为定义的数组起个变量名，满足标识符规范，可以使用该名字操作数组
- new：关键字，创建数组使用的关键字
- [长度]：数组的长度，表示数组容器中可以存储多少个元素
- **注意事项：数组有定长特性，长度一旦指定，不可更改**

例：

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

1.1.2 格式二

```
数组存储的数据类型[] 数组名 = new 数组存储的数据类型[]{元素1,元素2,元素3,...};
```

例：

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

1.1.3 格式三

```
数组存储的数据类型[] 数组名 = {元素1,元素2,元素3,...};
```

例：

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

1.2 数组的访问

访问格式：

```
数组名[索引]
```

- **索引**：每一个存储到数组的元素，都自动的拥有一个编号，从0开始，这个自动编号称为**数组索引（index）**，可以通过数组的索引访问到数组中的元素
- **数组的长度属性**：每个数组都具有长度，而且是固定的，Java中赋予了数组一个属性，可以获取到数组的长度，语句为：`数组名.length`，属性`length`de 执行结果是数组的长度，为int类型。由此可以推断出，数组的最大索引值为`数组名.length+1`

```
public static void main(String[] args) {
    int[] arr = new int[]{1,2,3,4,5};
    //打印数组的长度属性，结果为5
    System.out.println(arr.length);
}
```

- **索引访问数组中的元素**
  - `数组名[索引] = 数值;`：为数组中的元素赋值
  - `变量 = 数组名[索引];`：获取数组中的元素

```
public static void main() {
    int[] arr = {1,2,3,4,5};
    //为0索引元素赋值为6
    arr[0] = 6;
    //获取数组0索引上的元素
    int i = arr[0];
    System.out.println(i);
    //直接输出数组0索引元素
    System.out.println(arr[0]);
}
```

第二章 数组原理内存图

通过计算机组成原理我们知道，硬盘只是做存储功能使用，在硬盘中的程序是不会运行的，必须放进内存中才能运行，运行完毕后会清空内存。因此，JVM要运行程序，必须对内存进行空间的分配和管理。

2.1 JVM的内存划分

为了提高运行效率，就对空间进行了不同区域的划分，因为每一片区域都有特定的处理数据方式和内存管理方式

区域名称	作用
寄存器	给CPU使用，和我们开发无关
本地方法栈	JVM在使用操作系统功能的时候使用，和我们开发无关
方法区	存储可以运行的class文件
堆内存	存储对象或者数组，new来创建的都存储在堆内存
方法栈	方法运行时使用的内存，比如main方法运行，进入方法栈中执行

2.2 数组在内存中的存储

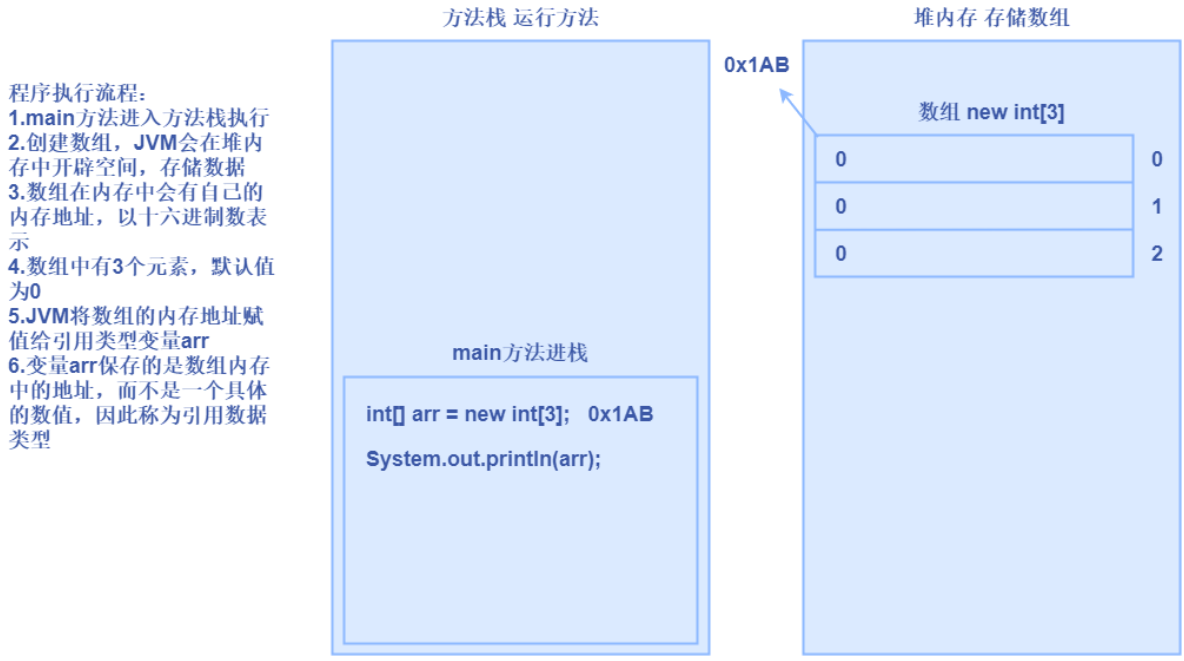
2.2.1 一个数组内存图

```
public static void main(String[] args) {
    int[] arr = new int[3];
    System.out.println(arr); //
}
```

运行结果：

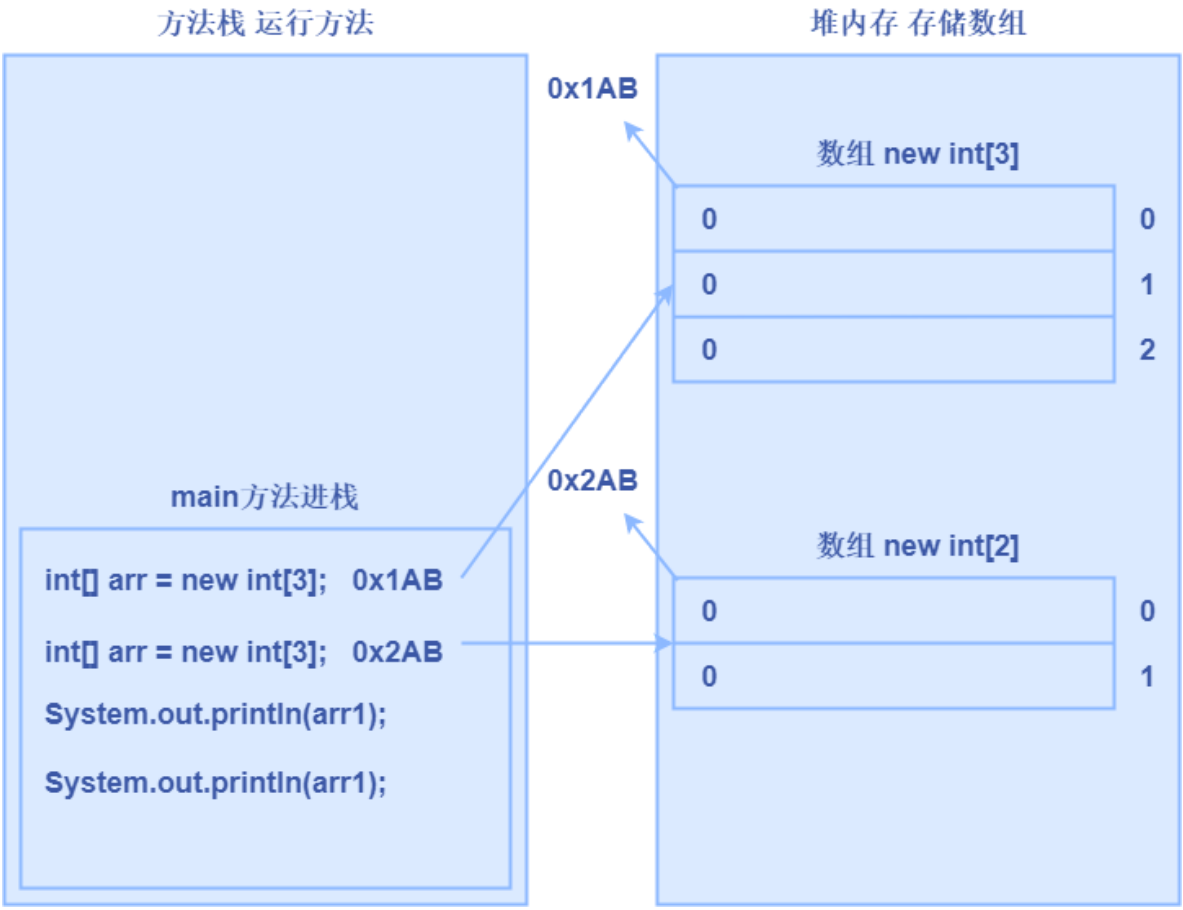
```
MethodSix
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files
[I@10f87f48

进程已结束，退出代码 0
```

- 运行结果为[I@10f87f48，这是数组在内存中的地址。new出来的内容，都是在堆内存中存储的，而方法中的变量arr保存的是数组的地址



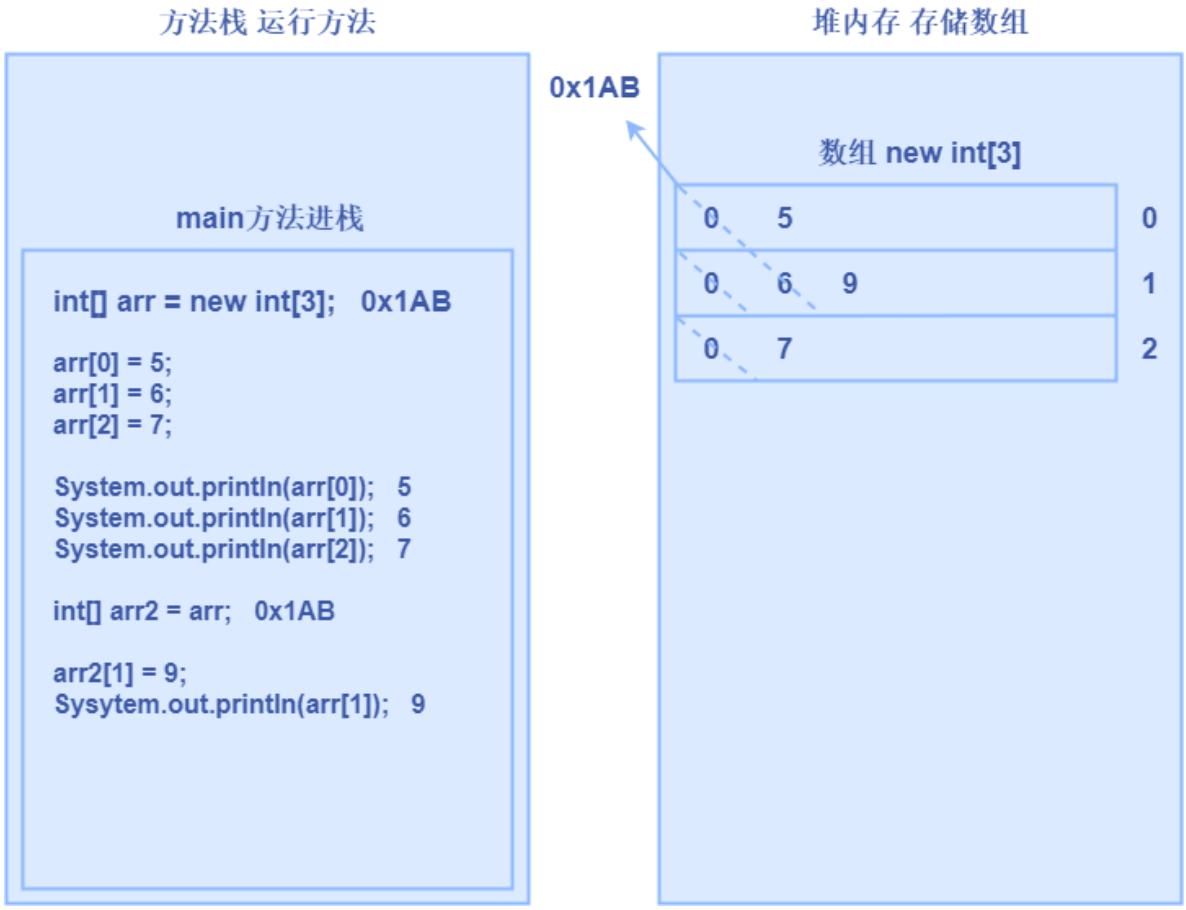
2.2.2 两个数组内存图

```
public static void main(String[] args) {
    int[] arr1 = new int[3];
    int[] arr2 = new int[2];
    System.out.println(arr1);
    System.out.println(arr2);
}
```



2.2.3 两个变量指向一个数组

```
public static void main(String[] args) {
    int[] arr = new int[3];
    arr[0] = 5;
    arr[1] = 6;
    arr[2] = 7;
    System.out.println(arr[0]);
    System.out.println(arr[1]);
    System.out.println(arr[2]);
    //定义数组变量arr2，将arr的地址赋值给arr2
    int[] arr2 = arr;
    arr2[1] = 9;
    System.out.println(arr[1]);
}
```



第三章 数组常见操作

3.1 数组越界异常

```
public static void main(String[] args) {
    int[] arr = {1,2,3};
    System.out.println(arr[3]);
}
```

```
MethodOne (1)
"D:\Professional Program Files\jdk-14\bin\java.exe" "-javaagent:D:\Professional Program Files\JetBrains\Intell
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
    at demoTwo.MethodOne.main(MethodOne.java:6)

进程已结束，退出代码 1
```

如上，程序运行后抛出`ArrayIndexOutOfBoundsException`数组越界异常，开发中，数组越界异常是不能出现的，一旦出现，必须修改代码

3.2 数组空指针异常

```
public static void main(String[] args) {
    int[] arr = {1,2,3};
    arr = null; //变量arr不会再保存数组的内存地址，也不再允许操作数组
    System.out.println(arr[0]);
}
```

如上，程序运行后抛出`NullPointerException`空指针异常，开发中，空指针异常也是不能出现的，一旦出现，必须修改代码

3.3 数组遍历【重点】

将数组中的每个元素分别取出来就是遍历，遍历是数组操作的基础，通常使用循环操作实现

```
public static void main(String[] args) {
    int[] arr = {1,2,3,4,5};
    for(int i = 0; i < arr.length; i++){
        System.out.println(arr[i]);
    }
}
```

3.4 获取数组最大值元素

基本思路：

- 定义变量，保存数组0索引上的元素
- 遍历数组，取出数组中的每个元素，并将元素与变量的值进行比较
- 若元素比变量的值大，则替换，反之略过
- 遍历结束，变量保存的就是数组的中的最大值

```
public static void main(String[] args) {
    int[] arr = {5,21,24,1,68,457,34};
    int max = arr[0];
    for(int i = 0; i < arr.length; i++) {
        if(arr[i] > max){
            max = arr[i];
        }
    }
    System,out,println("数组最大值是" + max);
}
```

3.5 数组的逆置

基本思路：

- 定义两个变量，分别保存数组最大和最小的索引
- 两个索引上的元素交换位置（引入第三方变量进行交换）
- 最小的索引加一，最大的索引减一，再次进行交换
- 最小索引大于最大索引，数组逆置操作结束

```
public static void main(String[] args) {
    int[] arr = {1,2,3,4,5};
    for(int max = 0, int min = arr.length - 1; min <= max; max--, min++) {
        //引入第三方变量完成数组元素交换
        int temp = arr[min];
        int arr[min] = arr[max];
        int arr[max] = temp;
    }
    //逆置后，进行数组遍历
    for(int i = 0; i < arr.length; i++) {
        System.out.println(arr[i]);
    }
}
```

第四章 数组作为方法参数和返回值

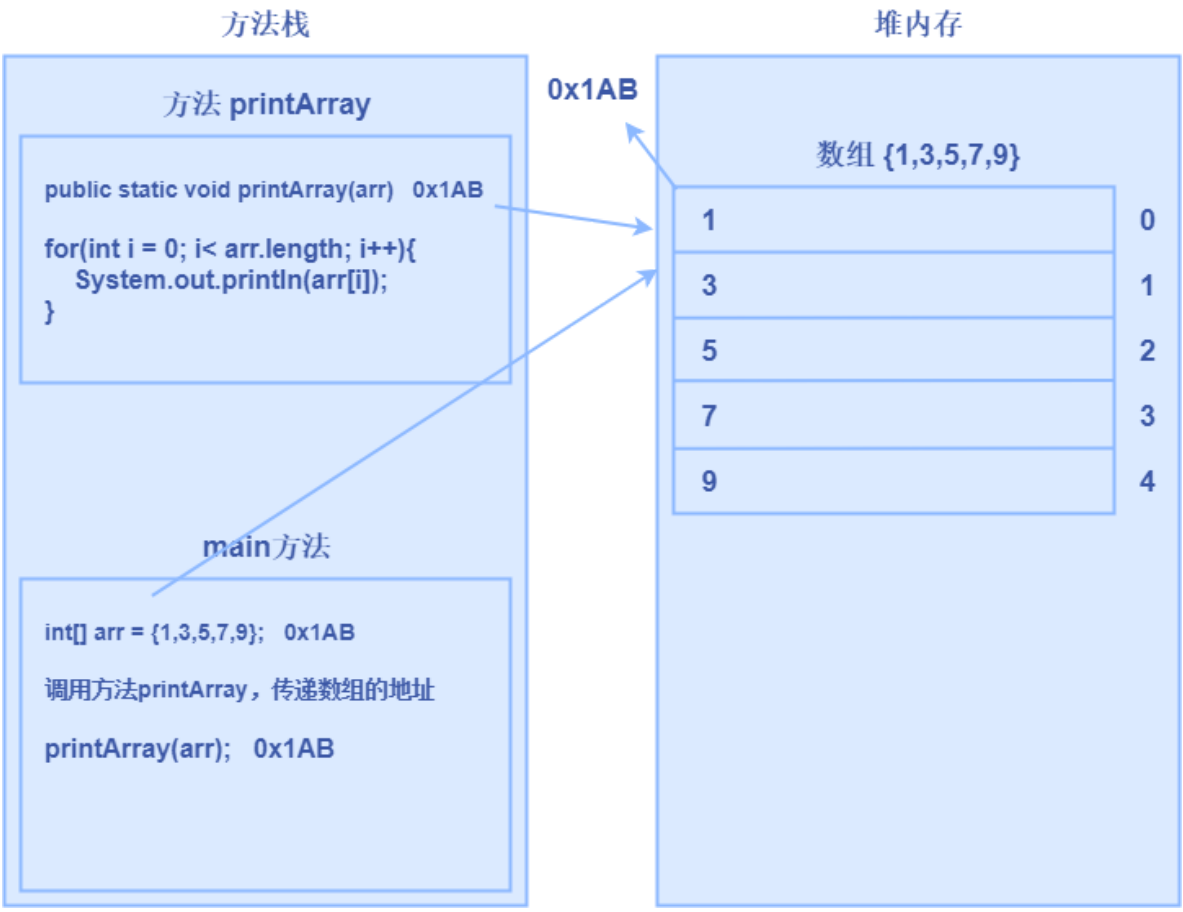
之前学习的方法，其参数和返回值使用的都是基本数据类型，作为引用数据类型的数组一样可以作为方法的参数和返回值进行传递

- 方法的参数为基本类型时，传递的是数据值；方法的参数为引用类型时，传递的是地址值

4.1 数组作为方法参数

- **数组作为方法参数传递，传递的参数是数组内存的地址**

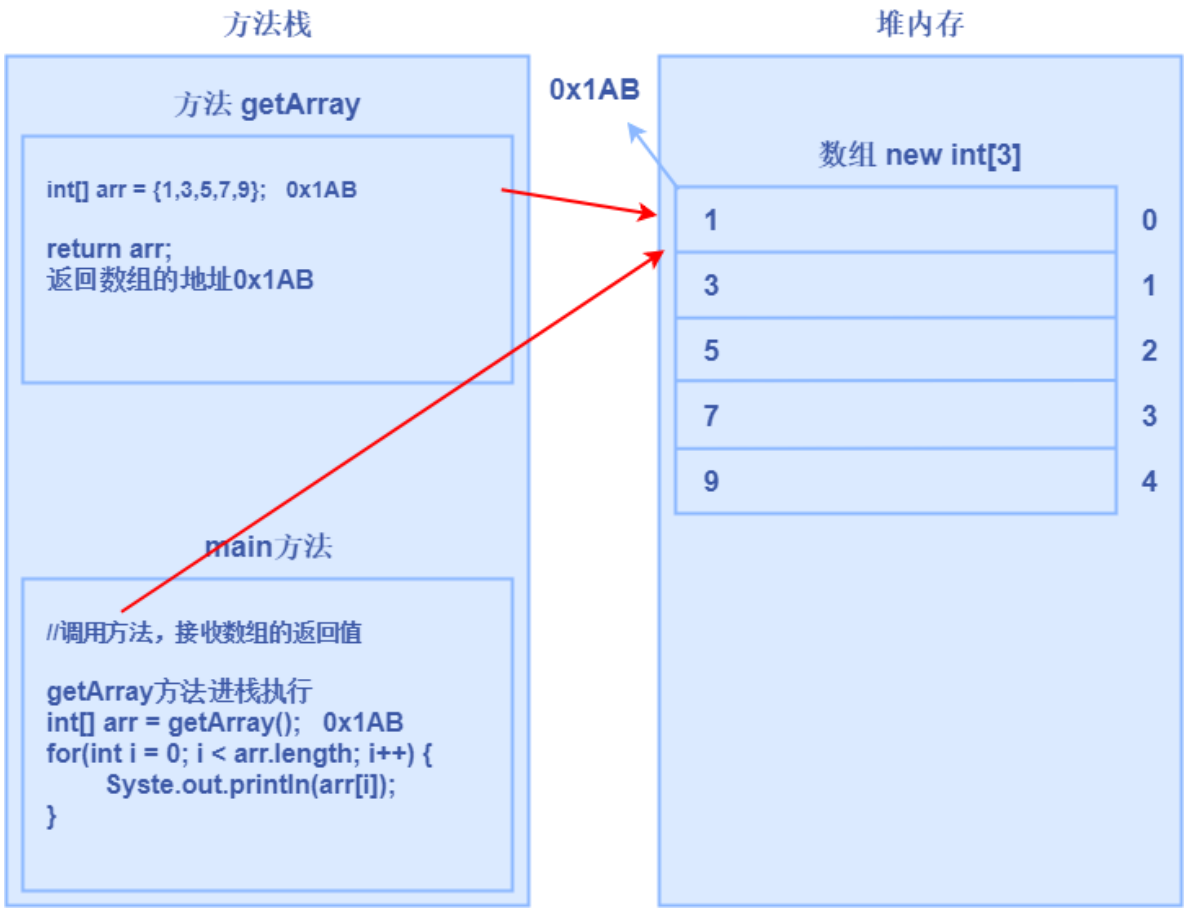
```
public static void main(String[] args) {
    int[] arr = {1,3,5,7,9};
    printArray(arr);
}

public static void printArray(int[] arr) {
    for(int i = 0; i < arr.lenght; i++) {
        System.out.println(arr[i]);
    }
}
```



4.2 数组作为方法返回值

- **数组作为方法返回值，返回的是数组内存的地址**

```
public static void main(String[] args) {
    int[] arr = getArray();
    for(int i = 0; i < arr.length; i++) {
        Syste.out.println(arr[i]);
    }
}

public static int[] getArray() {
    int[] arr = {1,3,5,7,9};
    return arr;
}
```


方法栈
堆内存
方法 getArray
int[] arr = {1,3,5,7,9}; 0x1AB
return arr;
返回数组的地址0x1AB
0x1AB
数组 new int[3]
1
3
5
7
9
0
1
2
3
4
main方法
//调用方法，接收数组的返回值
getArray方法进栈执行
int[] arr = getArray(); 0x1AB
for(int i = 0; i < arr.length; i++) {
Syste.out.println(arr[i]);
}