情境4：数组

1.数组的使用

2.一维数组

- （1）定义数组
- （2）数组的空间分配
- （3）访问数组
- （4）应用举例

3.二维数组

- （1）定义数组
- （2）数组的空间分配
- （3）访问数组
- （4）应用举例

变量在内存中如何分配

- 在讲数组之前，首先简单介绍变量如何被分配内存空间。
-
- 为存放变量值，**Java**采用两种方式来存放：
  - ✓ 堆内存
  - ✓ 栈内存

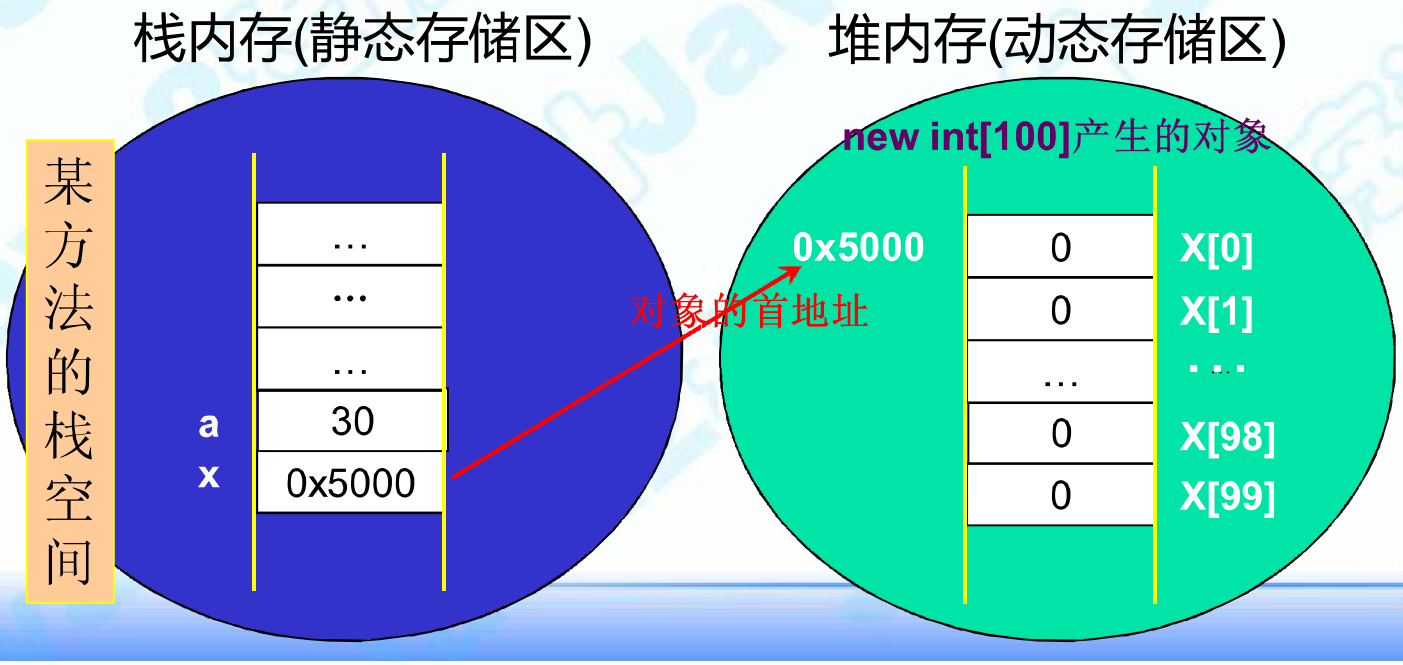
变量在内存中如何分配

- 栈内存：基本数据类型定义的变量，都是在栈内存中分配存储单元的。当每个方法被调用时，该方法会在栈中申请一个空间，在此方法中定义的变量都会分配到这个申请的栈空间。当方法运行结束时，分配给方法的栈空间被收回，在方法中定义的变量被自动释放。

- 堆内存：堆内存就是用**new**关键字创建数组和对象时，它们都被分配在堆内存中。堆内存中分配的数组和对象由**java**虚拟机的自动垃圾回收器来管理。在堆中产生的数组或对象后，必须在栈中定义一个变量，这个变量的取值是在堆栈产生数组或对象的首地址。这个在栈中定义的变量就是在堆栈产生数

变量在内存中如何分配

- 例：　int a=30;
- 　　int [] x;
- 　　x = new int[100];
- 　　x=null;



数组

- **Java**语言中，数组是一种最简单的复合数据类型。数组是有序数据的集合，要求数组中的每个元素具有相同的数据类型，可以用一个统一的数组名和下标来唯一地确定数组中的元素，下标用**[]**封装，数组的元素数目称为数组长度。数组有一维数组和多维数组。

换一说法：

- 数组是具有同一名字和相同类型的一组边续内存单元构成，数组中的每个数据称为元素，为了访问数组的某一无素，就应指定数组名及数组元素的位置序号，该序号称为数组下标。

如图所示为一个整型数组。

S[0]	1 1
S[1]	3 4
S[2]	5 6
S[3]	2 3
S[4]	7 2
S[5]	3 1

数组的声明

声明数组的语法格式如下：

数组元素类型　数组名［］

或者

数组元素类型［］　数组名

例如：　int c[];　或者　int[] c;

可以同时声明多个同类型的数组变量。

如：　**double[] c1,c2;**

注意：**java**与**C**或**C++**不同，不允许在数组名后面的方括号内指定数组元素的个数。

如：**int c[12]**　//语法错误

数组的内存分配

一个数组类型的变量存放一个对象引用。数组要占据内存空间，声明数组变量后需为数组分配内存空间。

语法格式如下：

数组名=new 数组元素类型[数组元素个数]

例：　int c[];　//声明数组变量

c=new int[12]　//分配内存

数组的声明和内存分配也可以合并到一起。

如： **int c[]=new int [12];**

一个声明语句可以同时声明多个数组，并为它们保留内存空间。

如： String a[]=new String[100],

b[]=new String[50];

一维数组

- **1.一维数组的声明**
  - 数据类型[ ] 数组名；
  - 数据类型 数组名[ ]；

  数据类型可以为**Java**中任意的数据类型，包括简单类型和复合类。

  例如：

```
int[] intArray;
date[] dateArray;
```

一维数组

2.一维数组的创建

- 当一个数组被声明以后，就可以通过下面的语法用**new**操作符创建它：

  数组名**=new**数据类型**[**数组大小**];**

- 另外，声明和创建数组可以被合并在一个语句里。

  数据类型**[]** 数组名**=new** 数据类型**[**数组大小**];**

  数据类型 数组名**[]=new** 数据类型**[**数组大小**];**

例如：**int[] myArray = new int[10];**

这条语句能够创建一个由**10**个**int**型元素构成的数组，为了指定数组中能够储存多少元素，给数组分配内存空间时，数组的大小必须事先给定。当一个数组创建完毕，不能再改变它的大小。

一维数组

3.一维数组的初始化

- 静态初始化

```
int[]  intArray = {1,2,3,4};
String  stringArray[] = {"abc", "How", "you"};
```

- 动态初始化
  - 简单类型的数组

```
int intArray[];
intArray = new int[5];
```

或

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

  - 复合类型的数组

```
String stringArray[ ];
String stringArray= new String[3];
```

一维数组

4. 一维数组元素的引用

数组元素的引用方式为：

数组名[下标]

数组下标，可以为整型常数或表达式，下标从**0**开始。每个数组都有一个属性**length**指明它的长度，数组下标从**0**到**length-1**。

例如：

intArray.length指明数组**intArray**的长度。数组元素分别是

intArray[0]、

intArray[1]、

……

intArray[intArray.length-1]。

一维数组

例】源程序**ArrayDemo.java**，创建一个整型数组。

```
1   //本程序创建一个整型数组
2   class ArrayDemo {
3       public static void main(String[] args) {
4           int[] anArray;                  //声明一个整型数组
5           anArray = new int[3];           //分配存储空间
6           anArray[0] = 100;               //初始化第一个元素
7           anArray[1] = 200;               //初始化第二个元素
8           anArray[2] = 300;               //初始化第三个元素
9           System.out.println("Element at index 0: " + anArray[0]);
10          System.out.println("Element at index 1: " + anArray[1]);
11          System.out.println("Element at index 2: " + anArray[2]);
12      }
13  }
```

一维数组

- 【运行结果】

- 【程序分析】

在第**6**～**13**句先创建了一个整型数组，然后输出其中元素的值。

二维数组

Java语言中，多维数组被看作数组的数组。

- 二维数组的声明
  - 数组类型 数组名[ ][ ];
  - 数组类型 [ ][ ]数组名;
- 二维数组的初始化
  - 静态初始化

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

注意：

Java中，由于把二维数组看作是多个一维数组组成，这些一维数组在分配的内存空间不是连续的，所以不要求二维数组每一维的大小相同。

二维数组

- 动态初始化
  - 直接为每一维分配空间，格式如下：

    数组名=new 数据类型[行数][列数];

    例：int a[ ][ ] = new int[2][3];

  - 从最高维开始，分别为每一维分配空间：

    数组名=数据类型[行数][ ];

    数组名[0] = new 数据类型[列数];

    数组名[1] = new 数据类型[列数];

二维数组

- 二维数组的引用
  - 对二维数组中的元素引用格式如下：
    数组名[行下标][列下标]
- 例：
- **int[ ][ ] xx=new int[3][ ];**
- 数组**xx**有三个都是**int [ ]**类型一维数组的元素

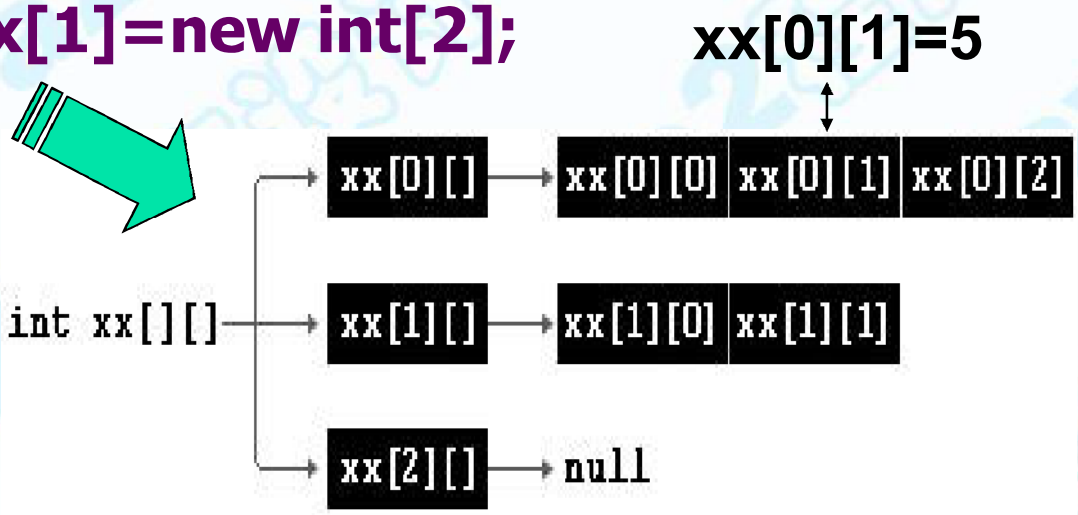
int xx[0][ ]

int xx[1][ ]

int xx[2][ ]

二维数组

- xx[0]=new int[3];
- xx[1]=new int[2];



思考：

xx.length //?

xx[0].length //?

int[][] xxx={{1,2,3},{4,5},{6}};//内存示意图?

二维数组

由上可以看出，对二维复合数据类型的数组，必须首先为最高维分配引用空间，然后再顺次为低维分配空间。而且，必须为每个数组元素单独分配空间。

在如：

```
String s[ ][ ] = new String[2][ ];
    s[0]= new String[2];              //为最高维分配引用空间
    s[1]= new String[2];              //为最高维分配引用空间
    s[0][0]= new String(“Good”);      // 为每个数组元素单独分配空间
    s[0][1]= new String(“Luck”);      // 为每个数组元素单独分配空间
    s[1][0]= new String(“to”);        // 为每个数组元素单独分配空间
    s[1][1]= new String("You");       // 为每个数组元素单独分配空间
```

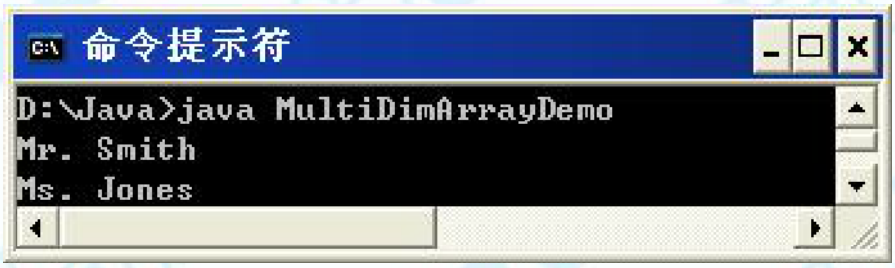
二维数组

例：二维数组的例子，程序名是**MultiDimArrayDemo**。

```
//本程序演示了二维数组的用法
class MultiDimArrayDemo {
    public static void main(String[] args) {
        String[][] names = {{"Mr. ", "Mrs. ", "Ms. "}, {"Smith", "Jones"}};
        System.out.println(names[0][0] + names[1][0]);
        System.out.println(names[0][2] + names[1][1]);
    }
}
```

二维数组

- 【运行结果】



- 【程序分析】
  - 本程序通过一个简单的小例子描述了二维数组的定义、初始化及其应用方法。
  - 需要注意的是数组中的下标是从**0**开始的。

小结

- **Java**中，对数组定义时并不为数组元素分配内存，只有初始化后，才为数组中的每一个元素分配空间。
- 已定义的数组必须经过初始化后，才可以引用。数组的初始化分为静态初始化和动态初始化两种。

Thank you!