计算机科学技术学院
School of Computer Science and Technology

Java 程序设计

第 2 章　基本数据类型与数组

导读

主要内容

- 标识符与关键字
- 基本数据类型
- 类型转换运算
- 输入、输出数据
- 数组

重点和难点

- **重点**：标示符、 Java 语言的数据类型、数组
- **难点**：基本数据类型的精度和数组

2.1 标识符与关键字

- 标识符
  - 用来标识类名、变量名、方法名、类型名、数组名、文件名的有效字符序列称为标识符，简单地说，标识符就是一个名字
  - class Person {}
  - int iCount = 10;

2.1 标识符与关键字

- 标 Java 语言规定
  - 标识符由字母、下划线、美元符号 ($) 和数字组成
  - 标识符的第一个字符不能是数字字符
  - 标识符不能是关键字
  - 标识符不能是 true 、 false 和 null （尽管 true 、 false 和 null 不是关键字）
  - 区分大小写

2.1 标识符与关键字

关键字

- 关键字就是具有特定用途或被赋予特定意义的一些单词，不可以把关键字作为标识符来用。
- 例如：

  boolean 、 byte 、 short 、 int 、 float 、 double 、 char 、 if 、 else 等
- 关键字都是小写的。遇到大写肯定不是关键字

2.1 标识符与关键字

- ✓ myVar 、 _strName 、 obj1 、 $int
- ✓ _Identifier 、 \u005fIdentifier
- ✗ 99var 、 It’sOK 、 enum 、 int

张三 ???

标识符可以包含 Unicode 字符，可以用 \uXXXX 指定 ！！

！

2.1 标识符与关键字 - 补充

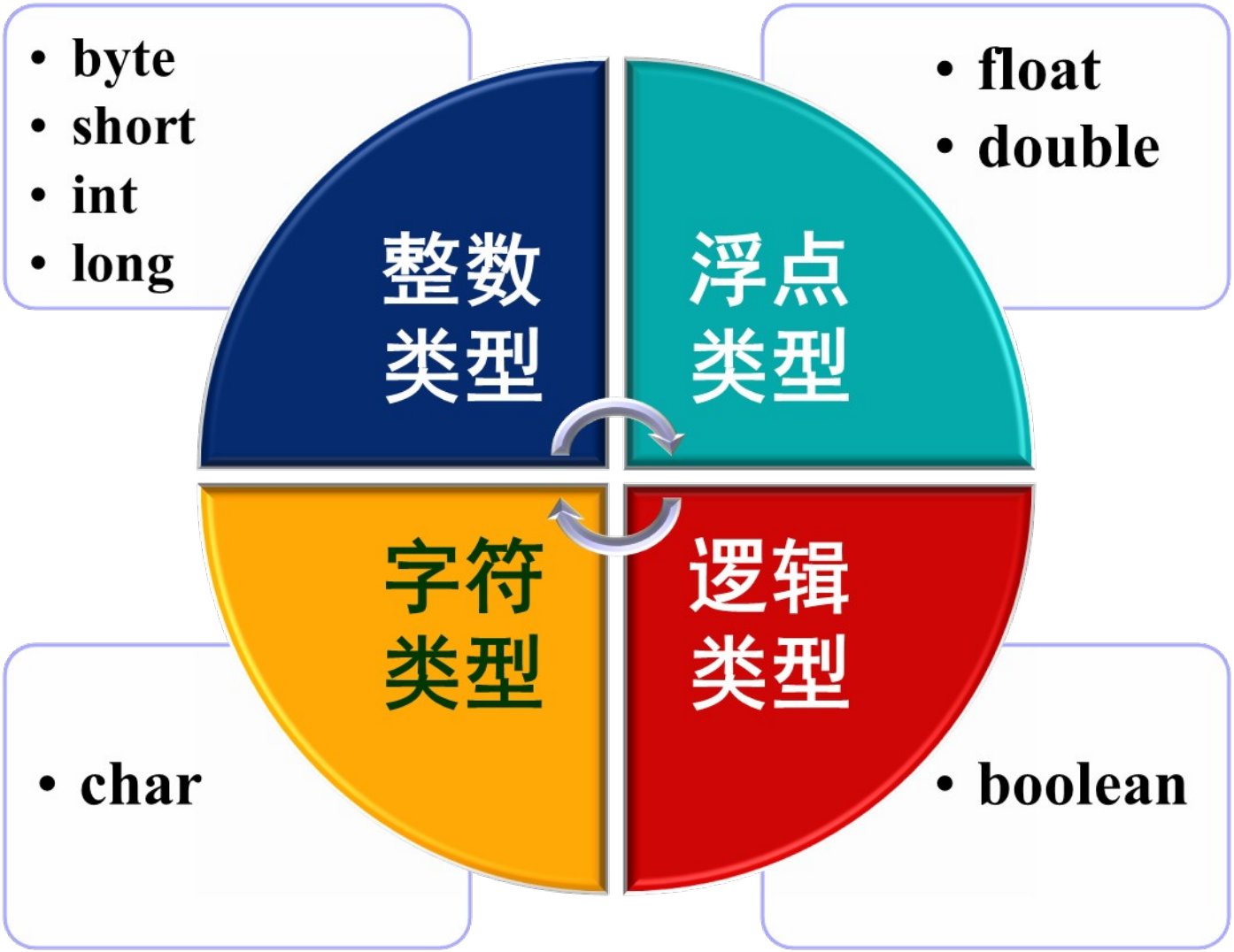
- 编码习惯
  - 类名首字母大写，变量、方法及对象首字母小写
  - 匈牙利—小写前缀

    iCount; sString; btnConfirm
  - camel— 大小写混合

    printExperimentReport; print_Experiment_Report
  - pascal— 大写混合

    PrintExperimentReport

2.2 基本数据类型


• byte
• short
• int
• long
整数类型
浮点类型
• float
• double
字符类型
逻辑类型
• char
• boolean

2.2.1 逻辑类型

- 变量
  - 使用关键字 boolean 来声明逻辑变量，声明时也可以赋给初值。
- 常量
  - true 、 false

例如：

boolean x, ok = true, 关闭 = false;

2.2.2 整数类型

- byte 型
  - 变量：使用关键字 byte 来声明 byte 型变量

    例如： byte x = -12, tom = 28, 漂亮 = 98;
  - 常量：一定范围内的 int 型常量赋值给 byte 型变量 .
  - 对于 byte 型内存分配给 1 个字节，占 8 位 .

2.2.2 整数类型

- short 型
  - 变量：使用关键字 short 来声明 short 型变量。

    例如： short x=12, y=1234;
  - 常量：和 byte 型类似， Java 中不存在 short 型常量的表示法，但可以把一定范围内的 int 型常量赋值给 short 型变量
  - 对于 short 型变量，内存分配给 2 个字节，占 16 位 .

2.2.2 整数类型

- int 型
  - 变量：使用关键字 int 来声明，声明时可赋给初值

    例如： int age = 20, 平均 = 9898, jiafei;
  - 常量： int 型常量共有三种表示方法：
    - 十进制： 123 ， 6000 （十进制）
    - 八进制： 077 （八进制，是零开头）
    - 十六进制： 0x3ABC （十六进制）
  - 对于 int 型变量，内存分配 4 个字节 (byte) ， 占 32 位

2.2.2 整数类型

- long 型
  - 变量：使用关键字 long 来声明 long 型变量
  - 常量： long 型常量用后缀 L 来表示，例如 108L (十进制)、 07123L(八进制)、 0x3ABCL(十六进制)

    例如： long width=12L, height=2005L, length;
  - 对于 long 型变量，内存分配给 8 个字节，占 64 位

§2.2.3 字符类型

- char 类型
  - 常量：‘A’，‘b’，‘?’，‘!’，‘9’，‘好’，‘\t’，‘き’，‘モ’等，即用单引号扩起的 Unicode 表中的一个字符
  - 变量： 使用关键字 char 来声明 char 型变量，对于 char 型变量，内存分配给 2 个字节，占 16 位

例如： char ch=‘A’, home=‘家’, handsome=‘酷’;

§2.2.3 字符类型

- 转意字符常量
  - 有些字符（如回车符）不能通过键盘输入到字符串或程序中，就需要使用转意字符常量

    例如： \n(换行), \b(退格), \t(水平制表), \‘(单引号),

    \“(双引号), \\(反斜线)等

Java 语言使用 Unicode 标准字符集，最多可以识别 65536 个字符 .

例子1

2.2.4 浮点类型

- float 型
  - 常量： 453.54F(小数表示法)， 2e40f(2 乘 10 的 40 次方，指数表示法).
  - 变量：使用关键字 float 来声明 float 型变量

    例如： float x=22.76f, tom=1234.987f, weight=1e-12F;
  - 精度 :float 变量在存储时保留 8 位有效数字。
  - 对于 float 型变量，内存分配给 4 个字节，占 32 位

需要特别注意的是 float 常量后面必须要有后缀 f 或

2.2.4 浮点类型

- double 型
  - 常量：同 float. 后缀有“ d” 或“ D”，允许省略
  - 变量：使用关键字 double 来声明 double 型变量

    例如： double height=23.345, width=34.56D, length=1e12;
  - 对于 double 型变量，内存分配给 8 个字节，占 64 位
  - 精度：保留 16 位有效数字，实际精度取决具体数值

注意具有小数部分的数据的缺省类型是 double 而不是 float

2.3 类型转换运算

- Java 中数据的基本类型 (不包括逻辑类型) 按精度从“低”到“高”排列：

2.3 类型转换运算

低->高
- 自动完成
- float x=100

高->低
- 显式转换
- int x = (int)12.5;

Int->byte, short
- byte b = 120;
- byte b = 129;

例题

- 下面的例子 2 使用了类型转换运算，运行效果如图

```
byte b = 22;
int n = 129;
float f = 123456.6789f;
double d = 123456789.123456789;
System.out.println(b);
System.out.println(n);
System.out.println(f);
System.out.println(d);
b = (byte)n;
f = (float)d;
System.out.println("b = " + b);
System.out.println("f = " + f);
```

```
C:\chapter2>java Example2_2
b=  22
n=  129
f=  123456.68
d=  1.2345678912345679E8
b=  -127
f=  1.2345679E8
```

图 2.2 类型转换运算

2.4 输入、输出数据 _1

2.4.1 输入基本型数据

- 可以使用 Scanner 类创建一个对象：

```
Scanner reader=new Scanner(System.in);
```

其中： reader 对象调用下列方法，读取用户在命令行输入的各种基本类型数据： nextXXX()　　nextInt()

- 上述方法执行时都会堵塞，程序等待用户在命令行输入数据回车确认

例题

例子 3 中，用户在键盘依次输入若干个数字，每输入一个数字都需要按回车键确认，在键盘输入数 0 结束整个的输入操作过程，程序将计算出这些数的和，运行效果如图 2.2

```
Scanner sc = new Scanner(System.in);
double sum = 0;
double x = sc.nextDouble();
while (x != 0) {
sum += x;
x = sc.nextDouble(); }
System.out.println("sum  = " + sum);
```

2.4 输入、输出数据 _2

2.4.2 输出基本型数据

- java 使用 System 类中的方法实现数据输出

  System.out.println() / System.out.print()

  System.out.println(m + " 个数的和为 " + sum);

  System.out.println(":" + 123 + " 大于 " + 122);

§2.4 输入、输出数据 _2 续

- JDK1.5 新增了和 C 语言中 printf 函数类似的输出数据的方法，格式如下：
- System.out.printf(" 格式控制 " ， 表达式 1 ， 表达式 2 ， …表达式 n)
  - %d 输出 int 类型数据值； %c 输出 char 型数据； %f 输出浮点型数据，小数部分最多保留 6 位； %s 输出字符串数据。
  - 输出数据时也可以控制数据在命令行的位置，
  - 例如： %md=int 型数据占 m 列； %m.nf= 浮点型数据占 m 列，小数点保留 n 位。

例如： System.out.printf("%d, %f", 12, 23.78);

2.5 数组

- 数组是相同类型的数据按顺序组成的一种复合数据类型。通过数组名加数组下标，来使用数组中的数据。下标从 0 开始排序

2.5.1 声明数组

声明一维数组有下列两种格式：

数组的元素类型 数组名 [];

数组的元素类型 [] 数组名 ;

例如：

```
float boy[];
char [] cat;
```

声明二维数组有下列两种格式：

数组的元素类型 数组名 [][];

数组的元素类型 [][] 数组名 ;

例如：

```
float a[][];
char [][] b;
```

2.5.2 为数组分配元素空间

为数组分配元素的格式如下：

数组名 = new 数组元素的类型 [数组元素的个数];

例如： boy = new float[4];

说明：数组属于引用型变量，数组变量中存放着数组的首元素的地址，通过数组变量的名字加索引使用数组的元素（内存示意如图 2.4 所示）．比如：

```
boy[0] = 12;
boy[1] = 23.908F;
boy[2] = 100;
boy[3] = 10.23f;
```

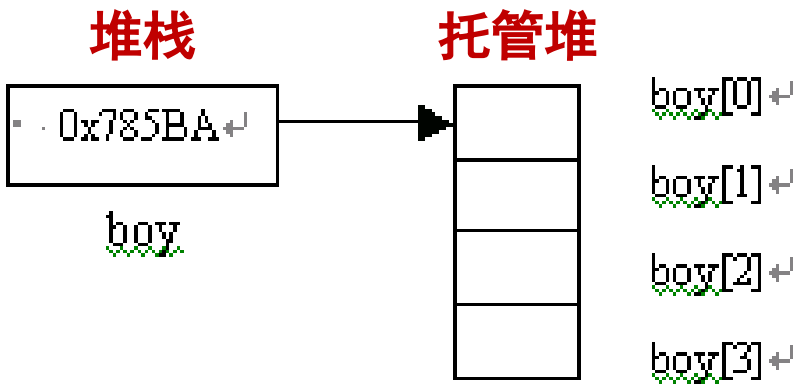


图 2.4 数组的内存模型

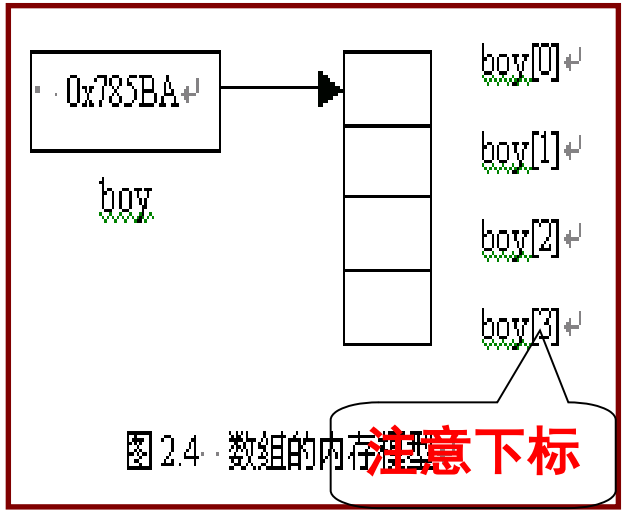
注意：数组的声明和分配空间可以在声明时同时完成：

float boy [] = new float[4];

2.5.3 数组元素的使用

- 数组元素的使用
  - 数组索引从 0 开始
  - 一维数组通过索引（下标运算）符访问自己的元素。如：boy[0]， boy[1] 等
  - 注意索引越界异常 ArrayIndexOutOfBoundsException

boy[4] = 384.98f;
①程序可以编译通过 ???
②运行是否会出错 ???



2.5.4 length 的使用

- length 字段的使用
  - 一维数组，“数组名 .length” 的值就是数组中元素的个数；
  - 二维数组“数组名 .length” 的值是它含有的一维数组的个数

例如：

```
float boy[] =new float[4];
boy.length 的值为 4
```

```
int [][] a = new int[3][8];
a.length 的值是 3.
a[0].length 的值是 8
a[1].length 的值是 8
a[2].length 的值是 8
```

2.5.5 数组的初始化

- 数组初始化
  - 创建数组后，数组中每个元素赋默认值，如 float 型是 0.0
  - 如果需要赋值，就要为每个元素赋值：例如：

```
float   boy  []=new float[4];
boy[0] = 12;
boy[1] = 23.908F;
boy[2] = 100;
boy[3] = 10.23f;
```

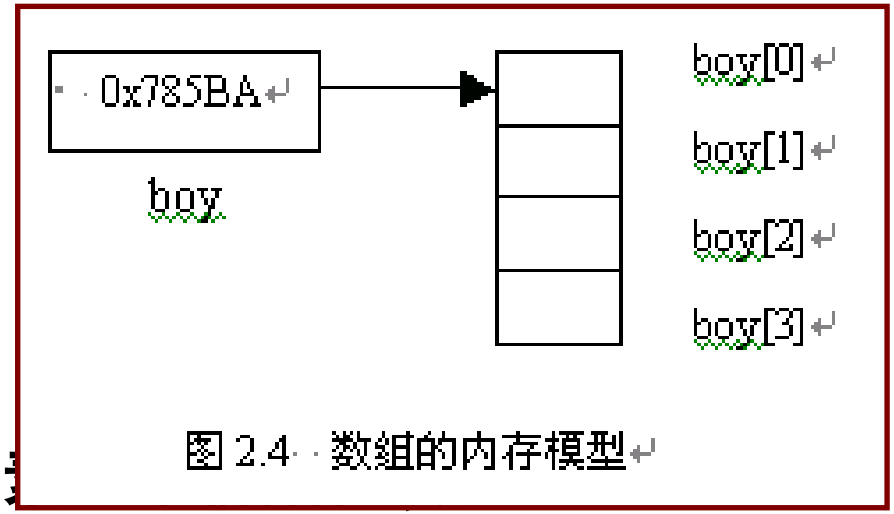


图 2.4 数组的内存模型

- 声明数组的同时也可以给数组元素

```
float boy[] = { 21.3f, 23.89f, 2.0f, 23f, 778.98f };
```

2.5.6 数组的引用

- 数组的引用
  - 数组属于引用型变量，两个相同类型的数组如果具有相同的引用，它们就有完全相同的元素

下面的例子 4 使用了数组，请读者注意程序的输出结果，运行效果如图 2.7

```
int[] a = {1, 2, 3, 4};
int b[] = {100, 200, 300};
```

```
数组a的元素个数=4
数组b的元素个数=3
数组a的引用=[I@de6ced
数组b的引用=[I@c17164
数组a的元素个数=3
数组b的元素个数=3
a[0]=100, a[1]=200, a[2]=300
b[0]=100, b[1]=200, b[2]=300
```

图 2.7 使用数组

重要结论

数组属于**引用型**变量，两个相同类型的数组如果具有**相同的引用**，它们就有完全**相同的元素**。

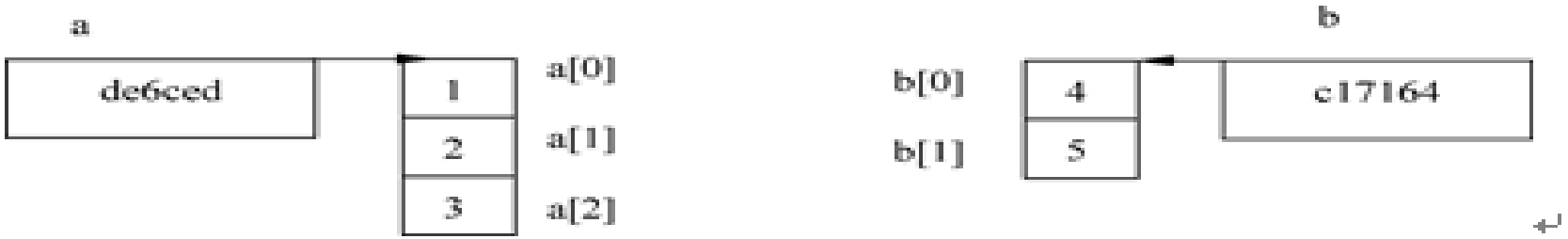


图 2.5　数组 a、b 的内存模型

```
a = b;
```

图 2.6　a=b 后的数组 a、b 的内存模型

2.6 应用举例

- 二维数组－转置

```
int[][] A =  { {1,2},{3,4},{5,6},{7,8}};
int[][] B = new int[2][4];
for (int i=0; i<A.length; i++)  {
    for (int j=0; j < A[i].length; j++)  {
        System.out.print(A[i][j]);
        System.out.print('\t');
    }
    System.out.print('\n');
}
```

2.6 应用举例

- **折半法：**对于从小到大排序的数组，我们只要判断数据是否和数组中间的值相等，然后在当前查找范围的一半中进行查找

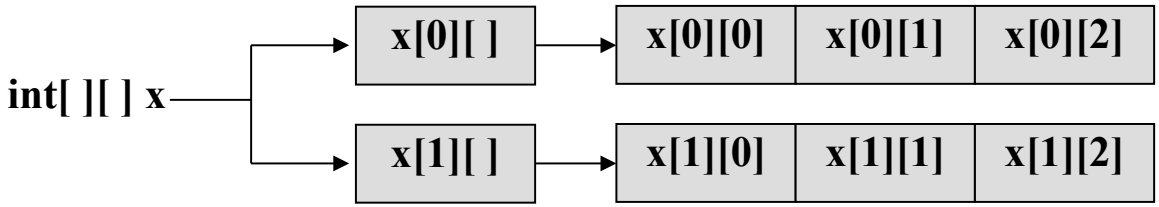
- 例子 5 能判断用户输入的一个整数是否在已知的数组中。程序效果如图 2.8 (P29)

多维数组

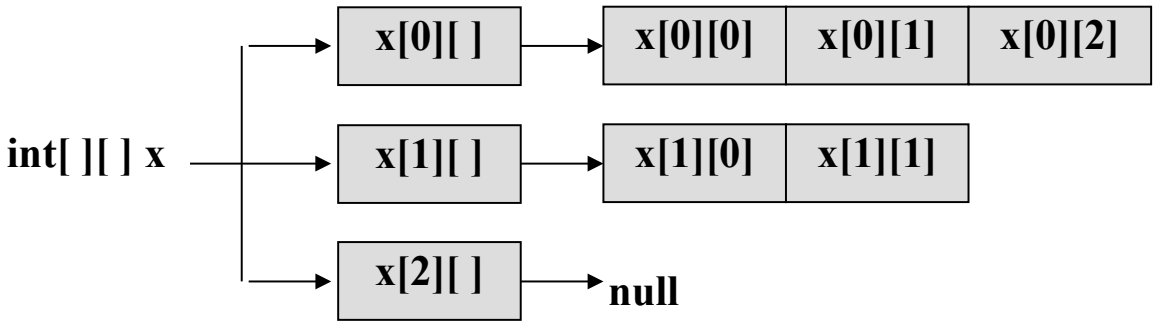
- 二维数组申请内存方式（必须指定高层维数）：
  - int[][] MyArray = new int [10][]; ✓
  - int[][] MyArray = new int [10][3]; ✓
  - int[][] MyArray = new int [][3]; ✗
  - int[][] MyArray = new int [][]; ✗

多维数组

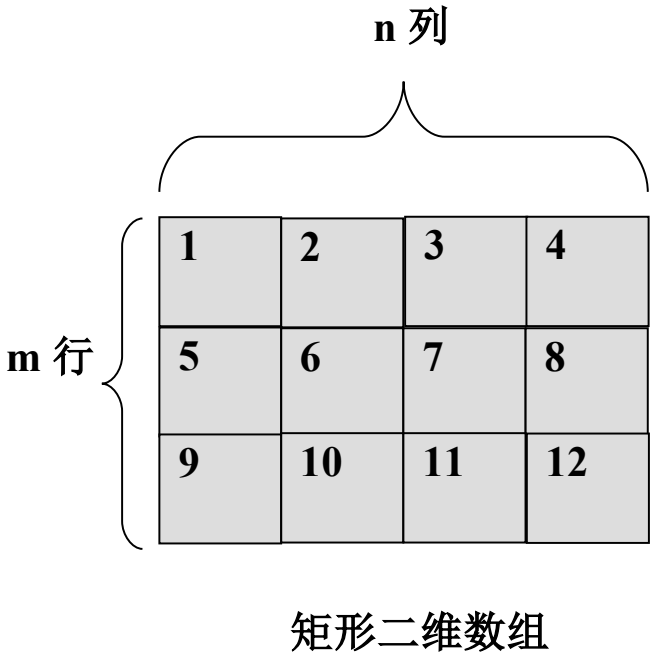
- 二维数组为规则 m ×n 矩阵
  - 创建高维数组对象的同时，创建所有的低维数组对象
  - 例如： int[][] x = new int [2][3];



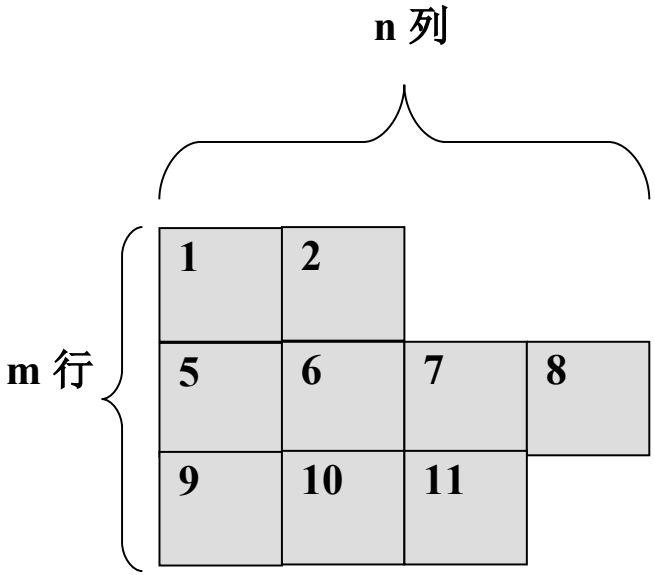
规则的二维数组内存分配



多维数组



矩形二维数组



总结

1　标识符由字母、下划线、美元符号和数字组成，
　　并且第一个字符不能是数字字符 (关键字不能做标识符).

2　Java 语言有 8 种基本数据类型
（ boolean byte short char int long float double ）

3　数组是相同类型的数据元素按顺序组成的一种
　　复合数据类型，数组属于引用型变量 .

4 两个相同类型的数组如果具有相同的引用，它们就有完全相同的元素。