



Java 程序设计

第3章 Java 基本编程结构

田英鑫 tyx@hit.edu.cn

哈尔滨工业大学软件学院



第3章 Java 基本编程结构

& 本章导读

- n 3.1 Java 基本语法
- n 3.2 注释
- n 3.3 标识符
- n 3.4 关键字
- n 3.5 基本数据类型
- n 3.6 变量
- n 3.7 赋值和初始化
- n 3.8 常量
- n 3.9 操作符
- n 3.10 字符串
- n 3.11 控制流程
- n 3.12 大数字
- n 3.13 数组
- n Java 命名习惯
- n Java 代码风格





第3章 Java 基本编程结构

3/56

& 本章重点

n 3.5 基本数据类型

n 3.10 字符串

n 3.13 数组

& 本章难点

n 3.10 字符串

n 3.13 数组



3.1 Java 基本语法

n Java 代码的位置

- n Java的所有代码都是在类（或接口）中
- n 与C++不同，C++可以在类外定义变量和函数

n Java 是严格区分大小写的

- n Java是大小写敏感的语言

n Java 是一种自由格式的语言

- n 程序代码分为结构定义语句和功能执行语句
- n 功能执行语句的最后必须用分号（;）结束



3.2 注释

n Java 程序有三种注释方式

n 单行注释

//

n 多行注释

/* */

n 用于生成Java文档的注释

/** */



3.2 注释

n 适当的注释

- n 应该在程序开头写一个摘要，来说明程序的目的和主要特点是什么，解释所用的重要数据结构和独特技术
- n 在程序中要加上注释，介绍每一个主要步骤并解释难懂之处
- n 注释要简明，不能挤满程序反而降低了程序的可读性



3.3 标识符

n 什么是标识符？

- n 标识符是用来表示Java语言中包、类、方法、参数和变量等的名字的符号

n 标识符使用规则

- n 标识符由字母、数字、下划线（_）或美元号（\$）组成
- n 标识符以字母、下划线（_）或美元号（\$）开头
- n 标识符不能是关键字





3.4 关键字

abstract	do	implement	private	this
boolean	double	import	protected	throw
break	else	instanceof	public	throws
byte	extend	int	return	transient
case	false	interface	short	true
catch	final	long	static	try
char	finally	native	strictfp	void
class	float	new	super	volatile
continue	for	null	switch	while
default	if	package	synchronized	

注：java没有goto、const关键字，但不能用goto、const作为标识符



3.5 基本数据类型

n 数值型

类型	存储大小	取值范围
byte	8 bits	$-2^7 \sim 2^7-1$
short	16 bits	$-2^{15} \sim 2^{15}-1$
int	32 bits	$-2^{31} \sim 2^{31}-1$
long	64 bits	$-2^{63} \sim 2^{63}-1$
float	32 bits	$1.4e-45 \sim 3.4e+38$ $-1.4e-45 \sim -3.4e+38$
double	64 bits	$4.9e-324 \sim 1.7e+308$ $-4.9e-324 \sim -1.7e+308$



3.5 基本数据类型

n 数值字面量 (literal)

n 字面量是在程序中直接出现的基本类型的数据值，又称为右值

n 整数字面量默认类型为int

n 浮点数字面量默认类型为double

```
int i = 34;  
long l = 1000000;  
float f = 100.2F; or  
float f = 100.2f;  
double d = 100.2D; or  
double d = 100.2d;
```



3.5 基本数据类型

n 字符数据类型 char

n 字符数据类型用于表示单个字符

n 字符常量的表示

n `char letter = 'A';` (ASCII)

n `char letter = 65;` (ASCII)

n `char letter = '\u0041';` (Unicode)



3.5 基本数据类型

n 转意字符

转义字符	英文名称	Unicode
\b	Backspace	\u0008
\t	Tab	\u0009
\n	Linefeed	\u000A
\r	Carriage return	\u000D
\\	Backslash	\u005C
\'	Single Quote	\u0027
\"	Double Quote	\u0022



3.5 基本数据类型

n 布尔数据类型 boolean

n 布尔类型来自布尔代数，其值域包括两个值

n true

n false

n 与C/C++不同，在Java中数值不能代表布尔值

n 布尔类型的定义

n `boolean lightsOn = true;`

n `boolean lightsOn = false;`



3.6 变量

n 什么是变量？

- n 变量用来存储各种类型的数据

- n 如输入、输出和中间数据

- n 变量是系统为程序分配的一块内存单元

- n 根据所存储的数据类型的不同，有不同类型的变量

n 变量的声明

- n 先给出变量的类型，再给出变量名

```
double salary;  
int vacationDays;  
long earthPopulation;  
char yesChar;  
boolean done;
```



3.7 赋值和初始化

n 赋值语句

n 声明一个变量后，必须通过赋值语句对它进行初始化

n 永远不要使用一个未初始化的变量的值

n 要对一个已声明的变量赋值，应把此变量名写在左边，随后是等号（=），然后后边是有恰当值的Java表达式，声明和初始化可以一步完成

```
int vacationDays; //这条语句是声明  
vacationDays = 12; //这条语句是赋值  
int vacationDays = 12; //声明的同时初始化变量  
double salary = 65000.0;  
System.out.println(salary);  
int vacationDays = 12; //可以在代码的任何地方进行变量声明，与c不同
```



3.8 常量

n 什么是常量？

- n 不变的变量
- n 常量必须在定义时初始化，之后不能赋值

n 常量的声明

- n 使用关键字final来表示常量，例如：

- n `final double PI = 3.14159;`

- n `final int SIZE = 3;`

- n 常量的命名规则

- n 常量名一般用大写字母表示



3.8 常量

n 类常量

- n 在Java中，常常希望一个常量在某个类的多个方法中都是可用的，这种常量通常称为类常量
- n 使用static final可设定类常量

```
public class Constants
{
    public static final double CM_PER_INCH = 2.54;
    public static void main(String[] args)
    {
    }
    ... ..
}
```



3.9 操作符

n 算数操作符

n + , - , * , / , and %

//加法运算

5+2 等于 7

//减法运算

5-2 等于 3

//乘法运算

5*2 等于 10

//除法运算

5/2 等于 2

//模 (mod) 运算

5%2 等于 1



3.9 操作符

n 递增和递减操作符

```
x = 1;  
y = 1 + x++;  
y = 1 + ++x;  
y = 1 + x--;  
y = 1 + --x;
```

n 简捷赋值操作符

```
+=      i += 8      i = i+8  
-=      f -= 8.0    f = f-8.0  
*=      i *= 8      i = i*8  
/=      i /= 8      i = i/8  
%=      i %= 8      i = i%8
```





3.9 操作符

n 关系操作符

n 关系操作符又称比较操作符，用于结果是布尔值的表达式中

<	小于
<=	小于等于
>	大于
>=	大于等于
==	等于
!=	不等于
instanceof	检查对象的类型



3.9 操作符

n 布尔操作符

n 布尔操作符又称为逻辑操作符，用于对布尔值进行运算，其运算结果也是布尔值

&& 条件与
|| 条件或
! 非
?: 问号表达式

n 位操作符

& 与 | 或 ~ 非 ^ 异或
<< 左移 >> 右移 >>>



3.9 操作符

n & 与 | 用于布尔计算

n & 和 | 用于布尔运算时，将会产生布尔值

n 功能同 && 和 || 相似，但& 和 | 不按“短路”方式进行工作

```
public class Test {  
    public static void main(String[] args) {  
        int a = 2;  
        int b = 3;  
        if((a < 3) || (b-- < 2))  
            ;  
        System.out.println("a = " + a);  
        System.out.println("b = " + b);  
    }  
}
```



3.9 操作符

n 数学函数和常量

n Math类中的方法和常量

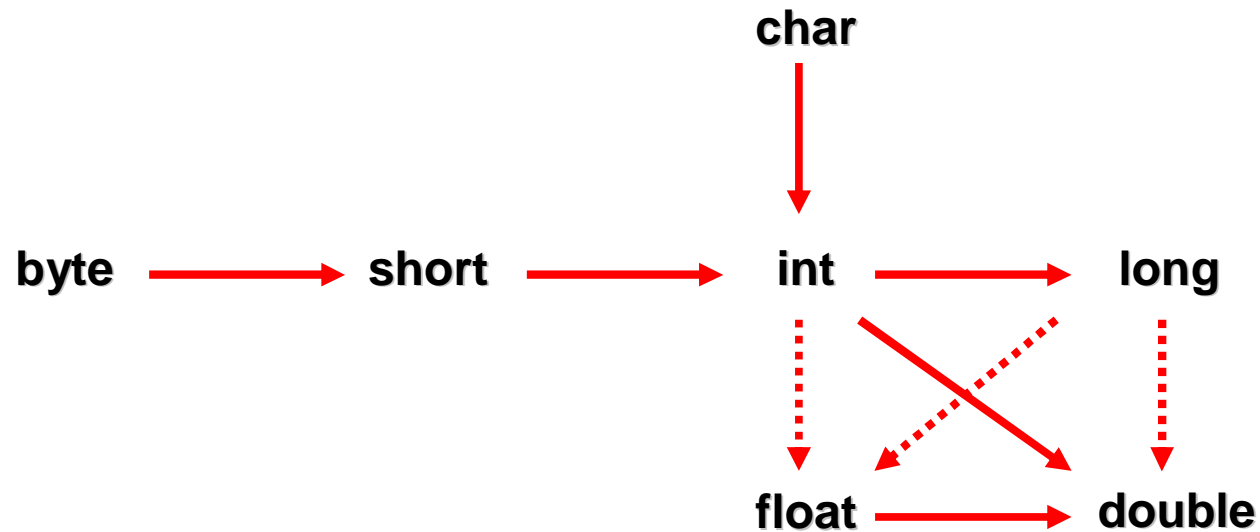
方法或常量名	功能	示例
Math.sqrt	求平方根	<code>y = Math.sqrt(x);</code>
Math.pow	求幂	<code>y = Math.pow(x,a);</code>
Math.sin	三角函数	
Math.exp	指数函数	
Math.log	自然对数	
Math.PI		
Math.E	e	



3.9 操作符

n 数字类型之间的转换

n 隐式转换



注：实线表示转换无信息损失，虚线表示转换时可能丢失精度



3.9 操作符

n 数字类型之间的转换

n 二元操作符表达式类型转换规则

- n 如果两个操作数中有一个是double类型的，则另一个将会转换成double类型
- n 否则，如果两个操作数中有一个是float类型的，另一个将会转换成float类型
- n 否则，如果两个操作数中有一个是long类型的，另一个将会转换成long类型
- n 否则，两个操作数都将转换成int类型



3.9 操作符

n 强制类型转换

n 格式

- n 在圆括号中给出要转换的目标类型，随后是待转换的变量名，例如：

```
double x = 9.997;
```

```
int nx = (int)x; //nx的值为9
```

```
nx = (int)Math.round(x); //nx的值为10
```

```
byte bx = (byte)300; //bx值为44
```

n 注意事项

- n 不能在布尔值和任何数字类型间强制类型转换，从而避免了常见的错误



3.9 操作符

n 圆括号

- n 最好使用圆括号指明运算的执行顺序

n 操作符级别

- n 见教材45页表3-4



3.10 字符串

n 什么是字符串？

n 字符串指的是字符序列

n Java中并没有内置的字符串类型，而是在标准Java库中包含一个String类

n 字符串是一种简单而且常用的数据结构

n 字符串的声明

n 象使用其它基本数据类型一样，使用String定义字符串变量

`String e = "";` //一个空字符串

`String greeting = "Hello";`



3.10 字符串

n 字符串连接

- n 和大多数程序语言一样，Java允许使用符号"+"把两个字符串连接（Concatenate）在一起

```
String s1 = "Welcome to Java";
```

```
String s2 = s1 + " Programming";
```

```
//s2为"Welcome to Java Programming"
```

- n 当连接一个字符串和一个非字符串时，非字符串将转换成字符串

```
int age = 13;
```

```
String rating = "PG" + age; //rating为字符串"PG13"
```

```
int answer = 100;
```

```
System.out.println("The answer is " + answer);
```



3.10 字符串

n 子串

- n String类中的substring方法可以从字符串中截取一个子串，例如：

```
String greeting = "Hello";
```

```
String s = greeting.substring(0,4); //s的值为"Hell"
```

- n `substring(int beginIndex, int endIndex);`
 - n `beginIndex`为欲截取字符串的起始位置索引
 - n `endIndex`为不想截取的起始位置
 - n 字符串中第一个字符的位置为0



3.10 字符串

n 字符串编辑

- n length方法用来获取字符串长度

```
String greeting = "Hello";
```

```
int n = greeting.length(); //n等于5
```

- n charAt方法用来获取字符串中的单个字符

```
char last = greeting.charAt(4); //第4个字符是o
```



3.10 字符串

n 测试字符串是否相等

n 使用equals方法比较两个字符串是否相等

```
String s1 = "Hello";
String s2 = "Hello";
if (s1.equals(s2))
{
    // s1和s2具有相同的内容
}
if (s1 == s2)
{
    // s1和s2具有相同的引用
}
```




3.10 字符串

- n 其它常用方法

- n 参见教材49页和在线API文档



3.10 字符串

n 读取输入

n 在JDK1.4版本之前，Java中没有象C中的scanf或C++中的cin这样方便的从标准输入设备输入数据的方法

n 可以利用JOptionPane类的showInputDialog方法，该方法返回值为String类型，例如：

```
JOptionPane.showInputDialog("How old are you?");
```





3.10 字符串

n 输入练习

```
import javax.swing.*;
```

例3-2 : InputTest.java

```
public class InputTest {  
    public static void main(String[] args) {  
        String name = JOptionPane.showInputDialog  
            ("What is your name?");  
        String input = JOptionPane.showInputDialog  
            ("How old are you?");  
        int age = Integer.parseInt(input);  
        System.out.println("Hello," + name +  
            ". Next year,you'll be " + (age + 1));  
        System.exit(0);  
    }  
}
```



3.10 字符串

n 从字符串到基本类型的转换

n 使用目标数据类型包装类提供的类型转换方法

```
n String input =  
    JOptionPane.showInputDialog("How old are  
    you?");
```

```
n int age = Integer.parseInt(input);
```

n 先将字符串转换成基本数据类型的相应包装类数据，再通过包装类的成员方法转换成所需的基本类型

```
n boolean b =  
    Boolean.valueOf("true").booleanValue();
```

```
n int i = Integer.valueOf("100").intValue();
```



3.10 字符串

n 格式化输出

n NumberFormat类

```
import java.text.*;
class NumberFormatTest {
    public static void main(String[] args) {
        double x = 10000.0 / 3.0;
        NumberFormat formatter =
            NumberFormat.getNumberInstance();
        String s = formatter.format(x);
        System.out.println(s); //输出：3,333.333
    }
}
```



3.10 字符串

- n JDK 1.5 增强的数据输出功能

- n **System.out.printf**方法

- n 功能完全类似于C语言中的printf函数

- n JDK 1.5 增强的数据输入功能

- n **Scanner**类

- n Scanner是JDK 1.5新增的一个类，可以使用该类创建一个对象，然后调用该类的下列方法

- nextByte() nextDouble()

- nextFloat() nextInt()

- nextLine() nextLong() nextShort()

- n Scanner类包含在java.util包中



3.10 字符串

n JDK 1.5 的输入输出练习

```
import java.util.*;
public class Example {
    public static void main(String[] args) {
        System.out.println("请输入若干个数，每输入一个数回车确认");
        System.out.println("最后输入一个非数字结束输入操作");
        Scanner reader = new Scanner(System.in);
        double sum = 0;
        int m = 0;
        while(reader.hasNextDouble()) {
            double x = reader.nextDouble();
            m = m + 1;
            sum = sum + x;
        }
        System.out.printf("%d个数的和为%f\n", m, sum);
        System.out.printf("%d个数的平均值是%f\n", m, sum/m);
    }
}
```



3.11 控制流程

n 略



3.12 大数字

- n 如果基本的整型和符点型数据无法达到要求的精度，可以使用这两个类
 - n BigInteger和BigDecimal 类
 - n 所在包
 - n Java.math包
- n 使用大数字

```
BigInteger a = BigInteger.valueOf(100);  
//c = a + b  
BigInteger c = a.add(b);  
//d = c * (b + 2)  
BigInteger d = c.multiply(b.add(BigInteger.valueOf(2)));
```

参见教材例3-6 : BigIntegerTest.java



3.13 数组

n 数组介绍

- n 数组是用来存储一组相同类型数据的数据结构

n 声明数组

```
datatype[] arrayname; //例如 : int[] a; (推荐方式)  
datatype arrayname[]; //例如 : int a[];
```

n 创建数组

```
arrayName = new datatype[arraySize];  
//例如 : a = new int[100];
```

n 声明和创建一步完成

```
datatype[] arrayname = new datatype[arraySize];  
//例如 : int[] a = new int[100];
```



3.13 数组

n 数组初始化

n 默认初始化

- n 数组创建后，它的元素会按默认值进行初始化
- n 整数初始化为0，浮点数初始化为0.0，字符初始化为空字符，布尔类型初始化为false

n 使用循环

```
for(int i = 0; i < a.length; i++)  
    a[i] = i;
```

n 声明, 创建, 初始化一步完成

```
int[] smallPrimes = {2,3,5,7,11,13};
```



3.13 数组

n 匿名数组

- n 在Java中可以初始化一个匿名数组

```
new int[] {17,19,23,29,31,37};
```

- n 使用匿名数组对现有数组变量赋值

```
smallPrimes = new int[] {17,19,23,29,31,37};
```

- n 上述语句相当于下面两条语句的简写形式

```
int[] anonymous = {17,19,23,29,31,37};
```

```
smallPrimes = anonymous;
```



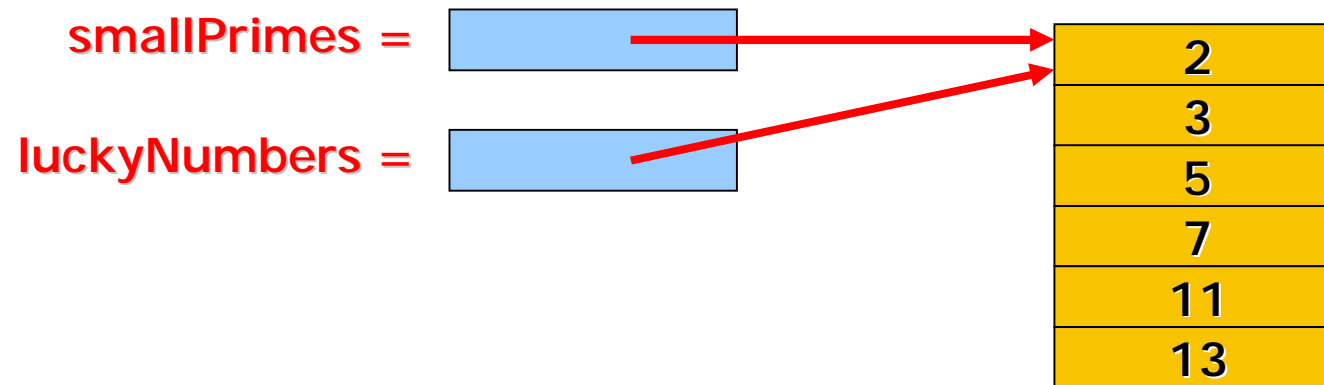
3.13 数组

n 拷贝数组

- n 可以把一个数组变量拷贝给另一个，这时，两个变量都指向相同的数组

```
int[] luckyNumbers = smallPrimes;
```

```
luckyNumbers[5] = 12; //smallPrimes[5]的值也是12
```





3.13 数组

n 拷贝数组

- n 可以使用循环语句将一个数组中的值拷贝给另一个数组变量

```
int[] sourceArray = {2, 3, 1, 5, 10};  
int[] targetArray = new int[sourceArray.length];  
  
for (int i = 0; i < sourceArray.length; i++)  
    targetArray[i] = sourceArray[i];
```



3.13 数组

n 拷贝数组

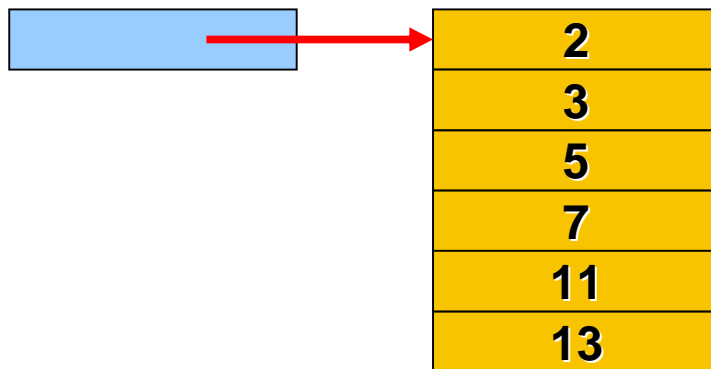
- n 如果希望将一个数组中的值拷贝给另一个数组变量，可以使用System类中的arraycopy方法

```
int[] smallPrimes = {2,3,5,7,11,13};
```

```
int[] luckyNumbers = {1001,1002,1003,1004,1005,1006,1007};
```

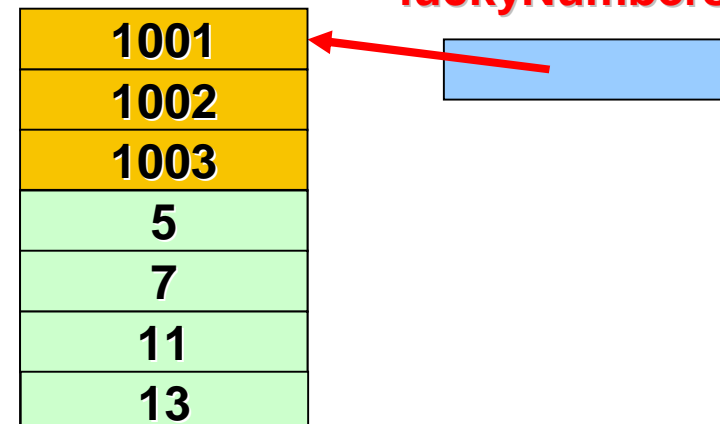
```
System.arraycopy(smallPrimes,2,luckyNumbers,3,4);
```

smallPrimes =



2
3
5
7
11
13

luckyNumbers =



1001
1002
1003
5
7
11
13



3.13 数组

n 拷贝数组

n 可以使用clone方法复制整个数组元素

```
int[] targetArray = (int[])sourceArray.clone();
```




3.13 数组

n 命令行参数

```
public class Message {  
    public static void main(String[] args) {  
        if(args[0].equals("-h"))  
            System.out.println("Hello,");  
        else if(args[0].equals("-g"))  
            System.out.println("Goodbye,");  
        //打印其它命令行参数  
        for(int i = 1;i < args.length;i++)  
            System.out.print(" " + args[i]);  
        System.out.println("!");  
    }  
}  
  
//运行时输入：java Message -g cruel world  
//程序输出：Goodbye,cruel world!
```



3.13 数组

n 对数组排序

- n 对数组中的元素排序，可以使用Arrays类中的sort方法，这个方法采用的是经过优化的快速排序算法，Arrays类包含在java.util包中

```
int[] a = new int[10000];
```

```
.....
```

```
Arrays.sort(a);
```

n 抽彩游戏

参见教材例3-7 : LotteryDrawing.java



3.13 数组

n 划分成绩等级

n 在这个例子中，程序能够通过从键盘读入的学生成绩（int），挑选最好的成绩，并且根据下表确定等级

- n 如果分数 \geq 最高分数 - 10，等级为 A；
- n 如果分数 \geq 最高分数 - 20，等级为 B；
- n 如果分数 \geq 最高分数 - 30，等级为 C；
- n 如果分数 \geq 最高分数 - 40，等级为 D；
- n 其他情况为E。

参见：AssignGrade.java



3.13 数组

n 多维数组

n Java中并没有真正的多维数组，所谓多维数组就是“数组的数组”

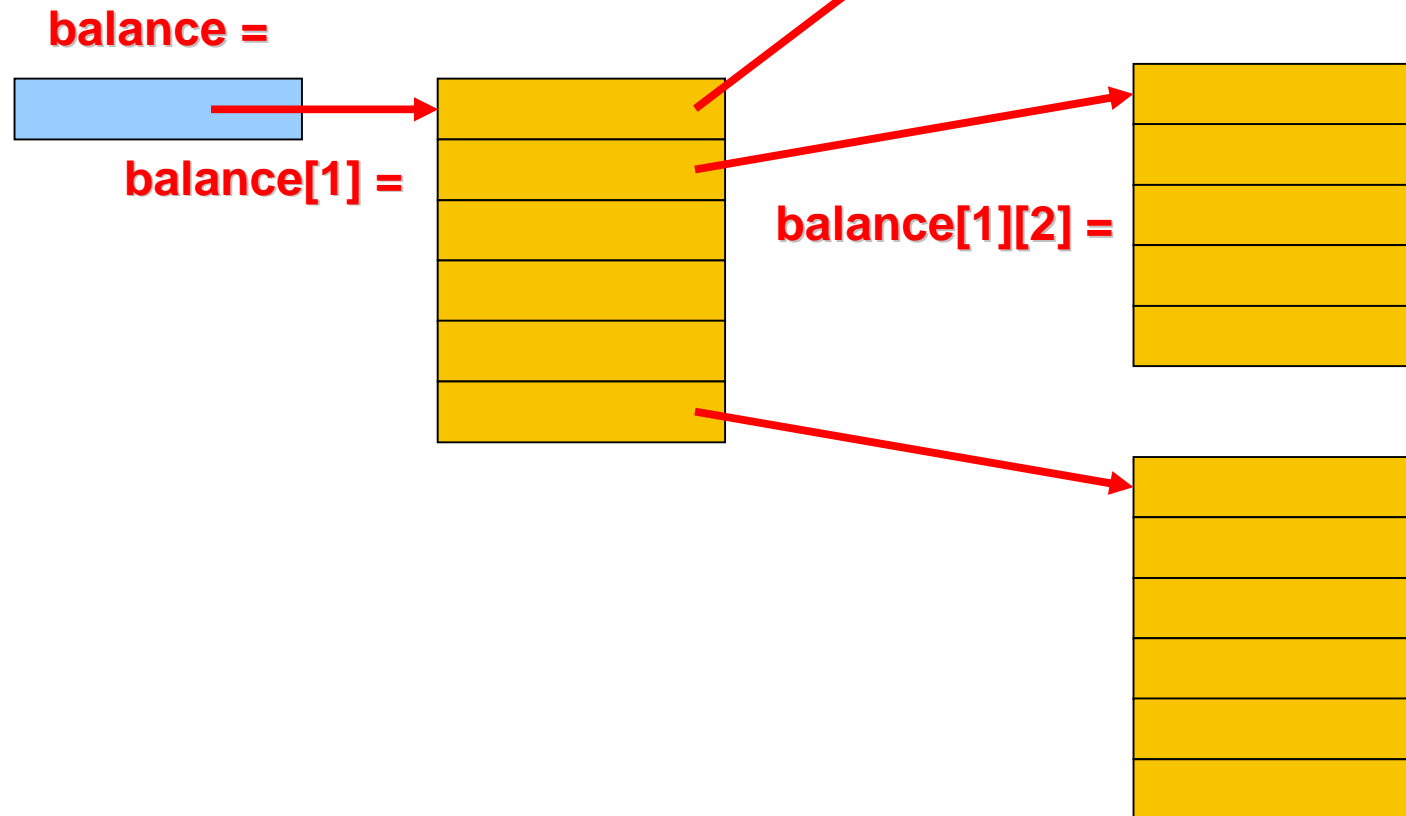
n 多维数组的定义和初试化

```
int[][] magicSquare =  
{  
    {16, 3, 2, 13},  
    {5, 10, 11, 8},  
    {9, 6, 7, 12},  
    {4, 15, 14, 1}  
};
```



3.13 数组

n 多维数组





Java 命名习惯

- n 选择有意义和描述性的名称
- n 变量和方法的命名
 - n 常用小写，如果名字包括几个词，则第一个词的字母小写而后面的每个词的首字母大写
 - n 例如, 变量 `radius` 和 `area`, 及方法 `computeArea`
- n 类命名
 - n 每个单词的首字母大写
 - n 例如, 类名 `ComputeArea`
- n 常量命名
 - n 常量中的所有字母都大写，多个词用下滑线连接
 - n 例如, 常量 `PI` 和 `MAX_VALUE`



Java 代码风格

n 风格一

```
class Test
{
    public void fun()
    {
        System.out.println(...);
    }
}
```

n 风格二

```
class Test {
    public void fun() {
        System.out.println("");
    }
}
```

n 风格三

```
class Test
{
    public void fun()
    {
        System.out.println(...);
    }
}
```



第3章 Java 基本编程结构

56/56



Any Question?