

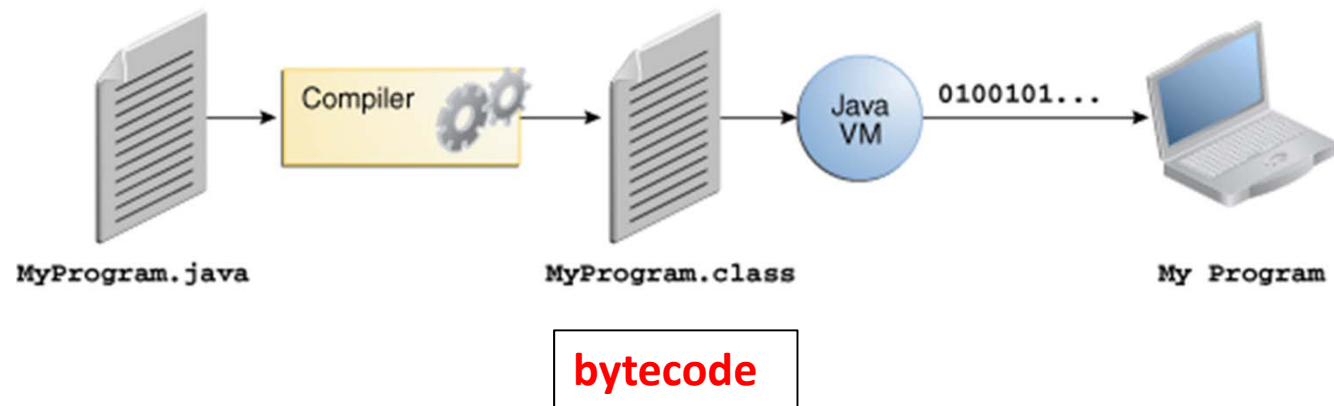
Java程序设计

Java Programming Design 毛斐巧

感谢：教材《Java大学实用教程》的作者和其他老师提供PowerPoint讲义等资料！
说明：本课程所使用的所有讲义，都是在以上资料上修改的。

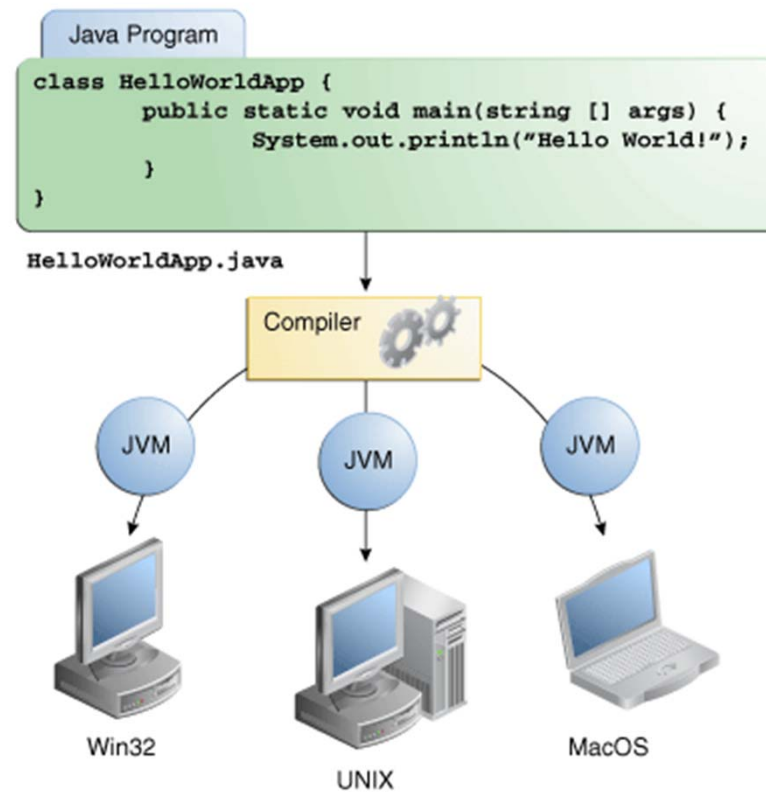
回顾

- An overview of the software development process



回顾

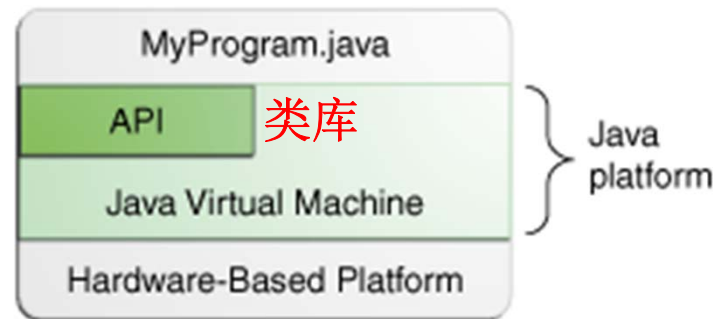
- Through the Java VM, the same application is capable of **running on multiple platforms**



<http://docs.oracle.com/javase/tutorial/getStarted/intro/definition.html>

回顾

- The **API** and **Java Virtual Machine** insulate (隔离) the program from the underlying hardware



- As a platform-independent environment, the Java platform can be **a bit slower than native code**. However, advances in compiler and virtual machine technologies are bringing performance close to that of native code without threatening portability.

标识符与基本数据类型及数组

主要内容

- 2.1 标识符和关键字
- 2.2 基本数据类型
- 2.3 基本数据类型的转换
- 2.4 数据的输入和输出
- 2.5 数组

2.1 标识符和关键字

- **1.标识符（identifiers）**
- 标识符就是一个名字，用来标识类名、变量名、方法名、数组名、文件名等的有效字符序列。
- Java语言规定标识符由字母、下划线、美元符号和数字组成，并且第一个字符不能是数字。长度不受限制。
- 标识符中的字母是区分大小写的，例如Beijing 和beijing是不同的标识符（即：case sensitive）。
- 标识符不能是true, false, null（尽管true, false, null不是关键字）。

2.1 标识符和关键字

- 2.关键字
- Java语言中已经被赋予特定意义的一些单词，它们在程序上有着不同的用途，不可以把关键字作为名字来用。
- abstract, continue, for, new, switch, assert***, default, goto*, package, synchronized, boolean, do, if, private, this, break, double, implements, protected, throw, byte, else, import, public, throws, case, enum****, instanceof, return, transient, catch, extends, int, short, try, char, final, interface, static, void, class, finally, long, strictfp**, volatile, const*, float, native, super, while
- * not used
- ** added in 1.2
- *** added in 1.4
- **** added in 5.0

Outline

- 2.1 标识符和关键字
- 2.2 基本数据类型
- 2.3 基本数据类型的转换
- 2.4 数据的输入和输出
- 2.5 数组

2.2 基本数据类型

- 基本数据类型（**primitive data types or fundamental types**）也称作**简单数据类型**。
- **基本数据经类封装后**，Java完全可以通过对象来处理基本数据类型，这就是Java声称它的所有数据都是**对象**的原因。

2.2 基本数据类型

- Java语言有8种基本数据类型
 - boolean, char, byte, short, int, long, float, double
- 8种基本数据类型可分为4大类型
 - 逻辑类型: boolean
 - 字符类型: char
 - 整数类型: byte, short, int, long
 - 浮点类型: float, double

2.2 基本数据类型

- **1.逻辑类型**
- 常量: true, false
- 变量的定义:
 - 关键字: boolean

2.2 基本数据类型

- 2.整数类型
- 常量：123(十进制)；077(八进制)；0x3ABC(十六进制)
- 变量的定义：
- (1) byte型
 - 关键字：byte；内存：1个字节，8位；取值范围：-2⁷~2⁷-1
- (2) short型
 - 关键字：short；内存：2个字节，16位；取值范围：-2¹⁵~ 2¹⁵-1
- (3) int型
 - 关键字：int；内存：4个字节，32位；取值范围：-2³¹~2³¹-1
- (4) long型
 - 关键字：long；内存：8个字节，64位；取值范围：-2⁶³~2⁶³-1

2.2 基本数据类型

```
8      short a = 1;  
9      short b = 2;  
10     short c = 3;  
11     b = a;  
12     System.out.println(b);  
13     a = c;  
14     System.out.println(a);  
15  
16     int d = 4;  
17     short e = d;
```

Java整数常量值默认为int类型

为什么第8行不报错，第17行报错？

数值1在-128在127值域范围内，所以第8行编译通过，不报错。Int型数据类型d赋给short型数据类型e，将大数据类型装进小数据类型，会发生溢出，所以编译不通过，报错。

2.2 基本数据类型

- 问题：银行卡号是一长串数字，很难辨认，怎么办？

```
public class LearningJava
{
    public static void main(String[] args)
    {
        long creditCardNumber = 2324_4545_4519_3415L;
        System.out.println(creditCardNumber);
    }
}
```

数字之间加入下划线分割数字，增强可读性。Java7之后引入的新特性。

2.2 基本数据类型

- In C and C++, **int** denotes the integer type that depends on the target machine.（平台相关）
 - On a 16-bit processor, like the 8086, integers are **2 bytes**.
 - On a 32-bit processor like the Sun SPARC, they are **4-byte** quantities.
 - On an Intel Pentium, the integer type of C and C++ depends on the operating system: For DOS and Windows 3.1, integers are **2 bytes**.
When 32-bit mode is used for Windows programs, integers are **4 bytes**.
- In Java, the sizes of all numeric types are **platform independent**.（平台无关）

2.2 基本数据类型

```
public class LearningJava
{
    public static void main(String[] args)
    {
        System.out.println( Byte.SIZE );
        System.out.println( Short.SIZE );
        System.out.println( Integer.SIZE );
        System.out.println( Long.SIZE );
    }
}
```

8
16
32
64

2.2 基本数据类型

- **3. 字符类型**
- 常量: **java**内部使用Unicode, Unicode表中的字符就是一个字符常量, 例如'A', '?', '9', '好', 'き', 等。Java还使用转意字符常量, 如:
 - '\n': 换行
 - '\t': 水平制表
 - '\': 单引号
 - '\"': 双引号
- 变量的定义:
- 关键字: **char**; 内存: **2个字节, 16位**; **最高位不是符号位**, 没有负数
取值范围: $0 \sim 2^{16}-1$, 即 $0 \sim 65535$
- Java支持中文变量名
- **Hints:** Always use '**single quotes**' for **char** literals and "**double quotes**" for **String** literals.

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

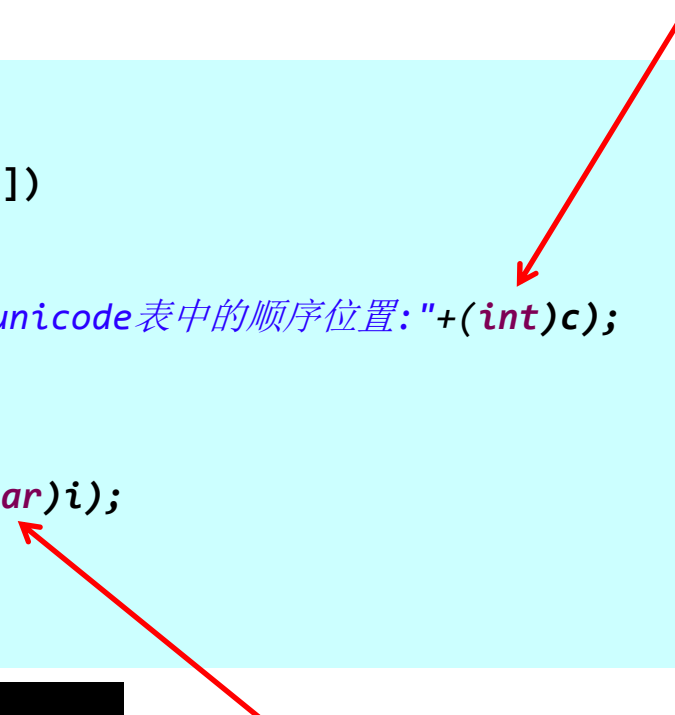
2.2 基本数据类型

- 要观察一个字符在Unicode表中的顺序位置，必须使用**int**类型显式转换，不可以使用short类型转换，**因为char的最高位不是符号位，有可能超出short的取值范围。**
- 同样，要得到一个0~65535之间的数所代表的Unicode表中相应位置上的字符也必须使用**char**类型显式转换。
- `char` $\leftrightarrow$ `int`

2.2 基本数据类型

- 【例子】用**显式转换**来显示一些字符在Unicode表中的位置，以及某些位置上的字符。

```
public class Example2_1
{
    public static void main (String args[ ])
    {
        char c='α';
        System.out.println("字母"+c+"在unicode表中的顺序位置:"+ (int)c);
        System.out.println("字母表: ");
        for(int i=(int)c; i<c+25; i++)
        {
            System.out.print(" "+(char)i);
        }
    }
}
```



字母α在unicode表中的顺序位置:945

字母表:

α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ τ υ φ χ ψ ω

2.2 基本数据类型

- 4. 浮点类型
- (1) float 型
- 常量: 453.5439f, 21379.987F, 2e40f (2乘10的40次方, 科学计数法)
- 变量的定义:
 - 关键字: **float**; 内存: 4个字节, 32位
 - 取值范围: $10^{-38} \sim 10^{38}$ 和 $-10^{38} \sim -10^{-38}$
 - A float value has **7 to 8 number of significant digits** (有效数字)

2.2 基本数据类型

- **(2) double型**
- 常量: 21389.5439d (d可以省略), **3.402**, 6e-140 (6乘10的-140次方)
- 变量的定义:
 - 关键字: **double**; 内存: 8个字节, 64位
 - 取值范围: $10^{-308} \sim 10^{308}$ 和 $-10^{308} \sim -10^{-308}$
 - A double value has **15 to 17 number of significant digits (有效数字)**
- Floating-point numbers **without an F suffix** (such as 3.402) are always considered to be of type **double**. You can optionally supply the D suffix (for example, 3.402D).
- **Hints: Normally, you should use the double type**, because it is more accurate than the **float** type.

2.2 基本数据类型

- Floating-point numbers are not suitable for **financial** calculation in which **roundoff (舍入) errors** cannot be tolerated.

```
class LearningJava
{
    public static void main(String[] args)
    {
        System.out.println(2.0-1.1);
    }
}
```

0.8999999999999999

- Such **roundoff errors** are caused by the fact that floating-point numbers are **represented in the binary number system**.
- Hints:** If you need precise numerical computations without roundoff errors, use the **BigDecimal** class.

2.2 基本数据类型

- 问题：为何称为浮点类型？
- The **float** and **double** types are used to represent numbers with a decimal point. Why are they called *floating-point* numbers?
- These numbers are stored in **scientific notation** (科学计数) internally. When a number such as 50.534 is converted into scientific notation, such as 5.0534E+1, its decimal point (小数点) is **moved (i.e., floated)** to a new position.

2.2 基本数据类型

- **Default Values**

Data Type	Default Value (for fields)
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
String (or any object)	null
boolean	false

- **Hints:** Relying on such default values, however, is generally considered **bad programming style**.

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

2.2 基本数据类型

- You may have noticed that the **new** keyword isn't used when initializing a variable of a primitive type. (基本数据类型变量初始化时不需用new)
- **Primitive data types (基本数据类型)** are special data types built into the language; they are **not objects** created from a class.

<http://docs.oracle.com/javase/tutorial/java/nutsandbolts/datatypes.html>

Outline

- 2.1 标识符和关键字
- 2.2 基本数据类型
- 2.3 基本数据类型的转换
- 2.4 数据的输入和输出
- 2.5 数组

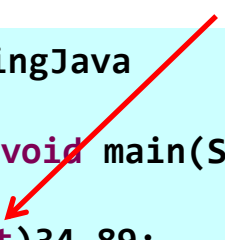
2.3 基本数据类型的转换（casting）

- 基本数据类型的转换就是把一种基本数据类型**变量**转变成另一种基本数据类型**变量**。下列基本类型会涉及数据转换，不包括逻辑类型和字符类型。我们将这些类型按**精度**从“低”到“高”排列（**smaller range to larger range**）
 - byte, short, int, long, float, double
- 当把**级别低的变量**的值赋给**级别高的变量**时，系统**自动**完成数据类型的转换，如int型转换成long型。
- 当把**级别高的变量**的值赋给**级别低的变量**时，必须使用**显式类型转换**运算。显示转换的格式：**(类型名)要转换的值**。
- 注：此处所说的级别是指“**精度**”上的级别。

2.3 基本数据类型的转换

- 【例子】

```
public class LearningJava
{
    public static void main(String[] args)
    {
        int x=(int)34.89;
        System.out.println("x=" + x);
    }
}
```



x=34

2.3 基本数据类型的转换

- 【例子】

```
public class Example2_2
{
    public static void main (String args[])
    {
        byte a=120;
        short b=130;
        int c=2200;
        long d=8000;
        float f;
        double g=0.1234567812345678;
        a=(byte)b;    //导致精度的损失
        c=(int)d;      //未导致精度的损失
        f=(float)g;   //导致精度的损失
        System.out.println("a="+a);
        System.out.println("c="+c);
        System.out.println("f="+f);
        System.out.println("g="+g);
    }
}
```

```
a=-126
c=8000
f=0.12345678
g=0.1234567812345678
```

2.3 基本数据类型的转换

分析: `a=(byte)b;` short数据类型是 16 位、有符号的以二进制补码表示的整数

- 130
- -> 0000 0000 1000 0010
- -> 1000 0010 最高位是符号位, 1表示负数
- -> 取反加1: 0111 1110
- -> 126
- -> -126

Outline

- 2.1 标识符和关键字
- 2.2 基本数据类型
- 2.3 基本数据类型的转换
- 2.4 数据的输入和输出
- 2.5 数组

2.4 数据的输入和输出

- 由于C语言出现的比较早，那个时候还没有图形用户界面（Graphics User Interface）的概念，因此，C语言提供了许多用来输入、输出数据的函数，例如printf, scanf等。
- Java不象C，提供在命令行进行数据输入、输出的功能不多。现在只需知道它的作用是在命令行窗口输入、输出数据即可。

2.4 数据的输入和输出

- 1.数据输出
- System.out.printf
- System.out.printf的功能完全类似C语言中的printf函数。printf的一般格式： printf(格式控制部分,表达式1,表达式2,...表达式n);
 - %d: 输出整型类型数据
 - %c: 输出字符类型数据
 - %f: 输出浮点类型数据，小数部分最多保留6位
 - %s: 输出字符串数据
 - %md: 输出的整型类型数据占m列
 - %m.nf: 输出的float数据占m列，小数点保留n位

2.4 数据的输入和输出

- 【例子】

```
public class Example2_3
{
    public static void main (String args[])
    {
        char c='A';
        float f=123.456789f;
        double d=123456.12345678;
        long x=5678;
        System.out.printf("%c\n%10.3f\n%f,%12d\n%d", c, f, d, x, x=x+2);
    }
}
```

```
A
    123.457
123456.123457,      5678
5680
```

2.4 数据的输入和输出

- 2.数据的输入
- Scanner是SDK1.5新增的一个类，可以使用该类创建一个对象：
 - `Scanner reader=new Scanner(System.in);`
- 然后reader对象调用下列方法，读取用户在命令行输入的各种数据类型，
 - `nextByte()`, `nextShort()`, `nextInt()`, `nextLong()`, `nextFloat()`, `nextDouble()`, `nextLine()`
- 上述方法执行时都会堵塞，等待你在命令行输入数据回车确认。

2.4 数据的输入和输出

这是wildcard import，也可以改为specific import（`import java.util.Scanner;`）

- 【例子】

```
import java.util.*;
public class Example2_4
{
    public static void main (String args[])
    {
        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);
    }
}
```

```
1
2
3
4
5
end
5个数的和为15.000000
5个数的平均值是3.000000
```

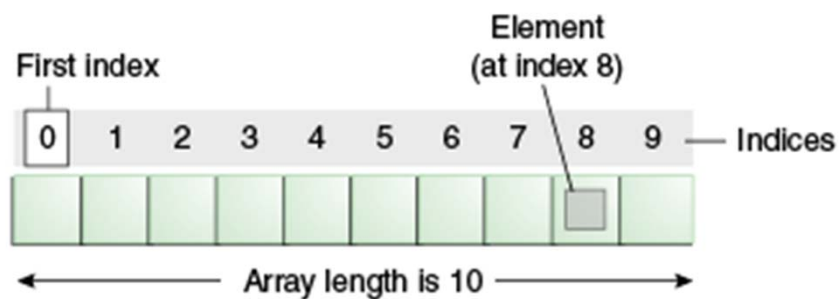
用户在键盘依次输入若干个数字，每输入一个数字都需要按回车键确认，最后在键盘输入一个非数字字符结束整个的输入操作过程。程序将计算出这些数的和及平均值。

Outline

- 2.1 标识符和关键字
- 2.2 基本数据类型
- 2.3 基本数据类型的转换
- 2.4 数据的输入和输出
- 2.5 数组

2.5 数组

- 数组（array）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（index）来使用数组中的数据。下标从0开始。



2.5 数组

- 1.声明数组
- 声明数组包括数组的**名字**、数组包含的元素的**数据类型**。
- 声明一维数组有下列两种格式：
 - 数组元素类型 **数组名字**[];
 - 数组元素类型 [] **数组名字**;
- 声明二维数组有下列两种格式：
 - 数组元素类型 **数组名字**[][];
 - 数组元素类型 [][] **数组名字**;

推荐的方式



2.5 数组

- **2.创建数组**
- 声明数组仅仅是给出了数组名和元素（`element`）的数据类型，要想使用数组还必须为它**分配内存空间**，即创建数组。
- 在为数组分配内存空间时**必须**指明数组的长度。

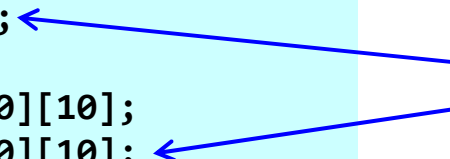
2.5 数组

- 【例子】

```
public class LearningJava
{
    public static void main(String[] args)
    {
        int arrayInt1 [] = new int[10];
        int [] arrayInt2 = new int[10];

        int arrayInt3 [][] = new int[10][10];
        int [][] arrayInt4 = new int[10][10];
    }
}
```

推荐的方式



2.5 数组

- 3.数组元素的使用
- 一维数组通过下标访问自己的元素。需要注意的是下标从0开始，因此，数组若是10个元素，下标到9为止，如果你使用的下标超过9将会发生异常（exception），即ArrayIndexOutOfBoundsException。
- 二维数组也通过下标访问自己的元素。下标也是从0开始。

2.5 数组

- 4.数组的初始化
- 创建数组后，系统会给每个数组元素一个默认的值，如，float型是0.0。在声明数组的时候给数组中的元素一个初始值是好的编程习惯。
- 【例子】

```
public class LearningJava
{
    public static void main(String[] args)
    {
        float [] arrayFloat = {21.3f, 23.89f, 2.0f, 23f, 778.98f};
        System.out.println(arrayFloat[0]);
        System.out.println(arrayFloat[1]);
        System.out.println(arrayFloat[2]);
        System.out.println(arrayFloat[3]);
        System.out.println(arrayFloat[4]);
    }
}
```

```
21.3
23.89
2.0
23.0
778.98
```

2.5 数组

- 数组属于引用型变量，因此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。

2.5 数组

- 【例子】

```
public class Example2_5
{
    public static void main(String args[])
    {
        int [] a={1,2,3};
        int [] b={10,11};
        System.out.println("数组a的引用是:"+a);
        System.out.println("数组b的引用是:"+b);
        System.out.printf("b[0]=%-3db[1]=%-3d\n",b[0],b[1]);
        b=a;
        System.out.println("数组a的引用是:"+a);
        System.out.println("数组b的引用是:"+b);
        b[1]=888;
        b[2]=999;
        System.out.printf("a[0]=%-5da[1]=%-5da[2]=%-5d\n",a[0],a[1],a[2]);
        System.out.printf("b[0]=%-5db[1]=%-5db[2]=%-5d\n",b[0],b[1],b[2]);
    }
}
```

```
数组a的引用是:[I@21ef48fb
数组b的引用是:[I@64a06824
b[0]=10 b[1]=11
数组a的引用是:[I@21ef48fb
数组b的引用是:[I@21ef48fb
a[0]=1      a[1]=888  a[2]=999
b[0]=1      b[1]=888  b[2]=999
```

2.5 数组

- 补充: arraycopy方法

```
class ArrayCopyDemo
{
    public static void main(String[] args)
    {
        char[] copyFrom = { 'd', 'e', 'c', 'a', 'f', 'f',
                             'e', 'i', 'n', 'a', 't', 'e', 'd' };
        char[] copyTo = new char[7];

        System.arraycopy(copyFrom, 2, copyTo, 0, 7);
        System.out.println(new String(copyTo));
    }
}
```

caffeine

```
public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)
```

2.5 数组

- 补充: `java.util.Arrays` class类
- <http://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html>

小节

- 2.1 标识符和关键字
 - 2.2 基本数据类型
 - 2.3 基本数据类型的转换
 - 2.4 数据的输入和输出
 - 2.5 数组
-
- 补充: <http://docs.oracle.com/javase/tutorial/>

补充：Java程序解剖

- Class name
- Main method
- Statements
- Statement terminator
- Reserved words
- Comments
- Blocks

Class Name

Every Java program must have at least one class. Each class has a name. By convention, class names start with an uppercase letter. In this example, the class name is Welcome.

```
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Welcome to Java!");
    }
}
```

Main Method

Line 2 defines the main method. In order to run a class, the class must contain a method named main. The program is executed from the main method.

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Statement

A statement represents an action or a sequence of actions. The statement `System.out.println("Welcome to Java!")` in the program in Listing 1.1 is a statement to display the greeting "Welcome to Java!".

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Statement Terminator

Every statement in Java ends with a semicolon (;).

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Reserved words (保留字)

Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word `class`, it understands that the word after `class` is the name for the class.

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Blocks

A pair of braces in a program forms a block that groups components of a program.

```
public class Test {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

Class block

Method block

Special Symbols

Character	Name	Description
{ }	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

{ ... }

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

(...)

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

;

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

// ...

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

'' ''
...

```
// This program prints Welcome to Java!  
public class Welcome {  
    public static void main(String[] args) {  
        System.out.println("Welcome to Java!");  
    }  
}
```

编程风格和文档

- Appropriate Comments
- Naming Conventions
- Proper Indentation and Spacing Lines
- Block Styles

Appropriate Comments

Include a summary at the beginning of the program to explain what the program does, its key features, its supporting data structures, and any unique techniques it uses.

Include your name, class section, instructor, date, and a brief description at the beginning of the program.

Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
 - Capitalize the first letter of each word in the name. For example, the class name `ComputeExpression`.

Proper Indentation and Spacing

- Indentation（缩进）
 - Indent two spaces.（缩进两个空格）
- Spacing（间隔）
 - Use blank line to separate segments of the code.

Block Styles

Use end-of-line style for braces.

*Next-line
style*

```
public class Test
{
    public static void main(String[] args)
    {
        System.out.println("Block Styles");
    }
}
```

*End-of-line
style*

```
public class Test {
    public static void main(String[] args) {
        System.out.println("Block Styles");
    }
}
```

编程错误

- Syntax Errors
 - Detected by the compiler
- Runtime Errors
 - Causes the program to abort
- Logic Errors
 - Produces incorrect result

Syntax Errors

```
public class ShowSyntaxErrors {  
    public static main(String[] args) {  
        System.out.println("Welcome to Java");  
    }  
}
```

ShowSyntaxErrors

Runtime Errors

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        System.out.println(1 / 0);  
    }  
}
```

ShowRuntimeErrors

Logic Errors

```
public class ShowLogicErrors {  
    public static void main(String[] args) {  
        System.out.println("Celsius 35 is Fahrenheit degree ");  
        System.out.println((9 / 5) * 35 + 32);  
    }  
}
```

ShowLogicErrors