

Chapter 2 Elementary Programming



先看一个例子

Listing 2.1 已知一个圆的半径，求圆面积。



分解动作

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```

给radius分配内存

radius

随机值



分解动作

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```

内存

radius

随机值

area

随机值

给area分配内存



分解动作

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```

radius

area

给radius赋值20

20

随机值



分解动作

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```



计算面积，把结果赋值
给变量area



分解动作

```
public class ComputeArea {  
    /** Main method */  
    public static void main(String[] args) {  
        double radius;  
        double area;  
  
        // Assign a radius  
        radius = 20;  
  
        // Compute area  
        area = radius * radius * 3.14159;  
  
        // Display results  
        System.out.println("The area for the circle of radius " +  
            radius + " is " + area);  
    }  
}
```

内存

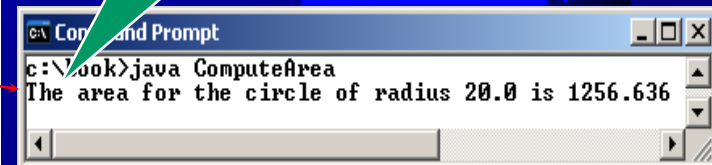
radius

20

area

1256.636

在控制台输出信息



```
CA Command Prompt  
c:\book>java ComputeArea  
The area for the circle of radius 20.0 is 1256.636
```

从控制台获取输入

1. `import java.util.*;` //使用Scanner前要先导入这个
2. 创建一个Scanner对象，随意取一个名字叫input。

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

3. 使用next(), nextByte(), nextShort(), nextInt(), nextLong(), nextFloat(), nextDouble(), nextBoolean()方法来读取各种不同类型的值。例如从控制台读取一个double，一个int，可以这样做：

```
System.out.print("请输入一个double和一个int: ");  
Scanner input = new Scanner(System.in);  
double d = input.nextDouble();  
int i = input.nextInt();
```



Scanner类的获取控制台输入的几个方法

☞ 一个程序只需要创建一个Scanner对象即可反复使用，例如上一个例子。

<i>Method</i>	<i>Description</i>
<code>nextByte()</code>	reads an integer of the <code>byte</code> type.
<code>nextShort()</code>	reads an integer of the <code>short</code> type.
<code>nextInt()</code>	reads an integer of the <code>int</code> type.
<code>nextLong()</code>	reads an integer of the <code>long</code> type.
<code>nextFloat()</code>	reads a number of the <code>float</code> type.
<code>nextDouble()</code>	reads a number of the <code>double</code> type.
<code>next()</code>	reads a string that ends before a whitespace character.
<code>nextLine()</code>	reads a line of text (i.e., a string ending with the <i>Enter</i> key pressed).

标识符

- ➡ 标识符可以由字母、数字、下划线 (_) 和美元符号 (\$) 组成。
- ➡ 标识符不能以数字开头，也不能使用关键字，例如 `class`, `true`, `false`, `null`。
- ➡ 标识符的长度是没有限制的。



变量的值是可变的

// 计算第一个面积

```
radius = 1.0;
```

```
area = radius * radius * 3.14159;
```

```
System.out.println("The area is " +  
    area + " for radius " + radius);
```

// 计算第二个面积

```
radius = 2.0;
```

```
area = radius * radius * 3.14159;
```

```
System.out.println("The area is " +  
    area + " for radius " + radius);
```



变量声明

```
int x;           // 声明x为整型变量
                // integer variable

double radius;  // 声明radius为浮点型变量
                // double variable

char a;         // 声明a为字符型变量
                // character variable
```

变量声明只是起到分配存储空间的作用，这些变量目前是没有值的，或者说，只有随机值。



赋值语句

```
x = 1;           // Assign 1 to x;  
radius = 1.0;    // Assign 1.0 to radius;  
a = 'A';         // Assign 'A' to a;
```



声明变量的时候可以同时初始化，这样变量就有了初值。

☞ `int x = 1;`

☞ `double d = 1.4;`



常量

语法:

```
final datatype CONSTANTNAME = VALUE;
```

举例:

```
final double PI = 3.14159;
```

```
final int SIZE = 3;
```

说明:

常量是有类型的。常量的值一旦确定后不可改变。



数值类型

Name	Range	Storage Size
byte	-2^7 (-128) to 2^7-1 (127)	8-bit signed
short	-2^{15} (-32768) to $2^{15}-1$ (32767)	16-bit signed
int	-2^{31} (-2147483648) to $2^{31}-1$ (2147483647)	32-bit signed
long	-2^{63} to $2^{63}-1$ (i.e., -9223372036854775808 to 9223372036854775807)	64-bit signed
float	Negative range: -3.4028235E+38 to -1.4E-45 Positive range: 1.4E-45 to 3.4028235E+38	32-bit IEEE 754
double	Negative range: -1.7976931348623157E+308 to -4.9E-324 Positive range: 4.9E-324 to 1.7976931348623157E+308	64-bit IEEE 754

算术运算符

Name	Meaning	Example	Result
+	Addition	$34 + 1$	35
-	Subtraction	$34.0 - 0.1$	33.9
*	Multiplication	$300 * 30$	9000
/	Division	$1.0 / 2.0$	0.5
%	Remainder	$20 \% 3$	2

两个整数相除，结果依然为整数

$5 / 2$ 得到整数2。

$5.0 / 2$ 得到浮点数 2.5。

$5 \% 2$ 得到1 (余数)



余数运算符（取模运算符） %

取模运算在编程中经常用到。

例如，一个偶数 % 2 结果一定是0，所以可以用这个运算判定一个数的奇偶性。

另一个例子是：假设今天是周六，10天之后是星期几呢？下面这个运算会告诉你是周二。

Saturday is the 6th day in a week

(6 + 10) % 7 is 2

A week has 7 days

The 2nd day in a week is Tuesday

After 10 days

编程练习：

将用户输入的秒数转为分钟数+秒数显示

LISTING 2.5 DisplayTime.java

```
1  import java.util.Scanner;                                import Scanner
2
3  public class DisplayTime {
4      public static void main(String[] args) {
5          Scanner input = new Scanner(System.in);          create a Scanner
6          // Prompt the user for input
7          System.out.print("Enter an integer for seconds: ");
8          int seconds = input.nextInt();                    read an integer
9
10         int minutes = seconds / 60; // Find minutes in seconds    divide
11         int remainingSeconds = seconds % 60; // Seconds remaining  remainder
12         System.out.println(seconds + " seconds is " + minutes +
13             " minutes and " + remainingSeconds + " seconds");
14     }
15 }
```

字面常量

字面常量指的是程序中直接出现的数字。例如下面例子中的 34, 1000000, 5.0:

```
int i = 34;
```

```
long x = 1000000;
```

```
double d = 5.0;
```



整型常量

- ✎ 整型常量的默认类型是int型，所以下面这个赋值编译器会直接报错：

```
byte b = 1000;
```

原因在于1000已经超出了byte的表示范围。

- ✎ int 的表示范围是 -2^{31} (-2147483648) 到 $2^{31}-1$ (2147483647)，超出这个范围的整数只能用long类型表示，写法是在数字后面多一个l或者L，表示long。大写的L是推荐用法，因为小写的l太像数字1了。例如这个写法就不推荐：

```
long value = 1l; //等号右边可不是11，要注意看。
```

浮点常量

- ➡ 浮点常量默认是double类型的。当然你也可以在数字后面直接补上d或者D，例如100.2d或100.2D，表示这个数真是double。
- ➡ 如果一定要让一个浮点常量是float型，必须在数字后面补上f或者F，例如100.2f或100.2F。



科学记数法

- ➡ 浮点数可以写成科学记数法，例如 $1.23456e+2$ ，或者写作 $1.23456e2$ ，等于 123.456，
- ➡ E（或者e）都可以用作科学记数法，二者没有区别。需要注意的是，E前面和后面都必须有数。E前面可以是整数或者浮点数，e后面只能用整数。



复合运算符

<i>Operator</i>	<i>Example</i>	<i>Equivalent</i>
<code>+=</code>	<code>i += 8</code>	<code>i = i + 8</code>
<code>-=</code>	<code>f -= 8.0</code>	<code>f = f - 8.0</code>
<code>*=</code>	<code>i *= 8</code>	<code>i = i * 8</code>
<code>/=</code>	<code>i /= 8</code>	<code>i = i / 8</code>
<code>%=</code>	<code>i %= 8</code>	<code>i = i % 8</code>



自增与自减运算符

Operator Name	Description
---------------	-------------

<u>++var</u>	先增 把var先自增，然后var新值作为整个表达式的值。
--------------	------------------------------

<u>var++</u>	后增 整个表达式的值取var原值，然后var再自增。
--------------	----------------------------

<u>--var</u>	先减 把var先自减，然后var新值作为整个表达式的值。
--------------	------------------------------

<u>var--</u>	后减 整个表达式的值取var原值，然后var再自减。
--------------	----------------------------

自增与自减运算符

```
int i = 10;  
int newNum = 10 * i++;  
System.out.print("i is " + i  
    + ", newNum is " + newNum);
```

Same effect as

```
int newNum = 10 * i;  
i = i + 1;
```

```
int i = 10;  
int newNum = 10 * (++i);  
System.out.print("i is " + i  
    + ", newNum is " + newNum);
```

Same effect as

```
i = i + 1;  
int newNum = 10 * i;
```



自增与自减运算符

为了避免自增和自减运算的二义性，当一个表达式中出现自增自减运算符时，**不能**让同一个变量出现多次，例如：

int k = ++i + i;

或者更变态的：

int k = ++i + ++i;



数值类型转换

下面的表达式都使用了隐式类型转换：

```
byte i = 100;
```

```
long k = i * 3 + 4;
```

```
double d = i * 3.1 + k / 2;
```



类型自动转换规则

如果参与运算的两个数类型不同，Java参照以下规则进行自动类型转换：

1. 如果一个数是double，另一个会被转成 double；
2. 否则，如果一个数是float，另一个也转成float；
3. 否则，如果一个数是long，另一个也转成long；
4. 否则，两个数都被转成int。



类型转换

隐式转换

`double d = 3;` (类型扩大, 不会丢失精度)

显式转换

`int i = (int)3.0;` (类型变小)

`int i = (int)3.9;` (小数点被截断)

这个式子错在哪里?

`int x = 5 / 2.0;`

range increases

byte, short, int, long, float, double

例题：如何保留两位小数？

假设tax为double，将tax保留两位小数可以利用强制类型转换来完成：

(int)(tax * 100) / 100.0

如果是四舍五入保留两位小数，可以这样：

(int)(tax * 100 + 0.5) / 100.0



命名规范

☞ 选择有意义的名字

☞ 变量和方法的命名：

- 使用小写。如果由多个单词 构成，则第一个单词小写，后面每个单词的首字母大写，例如，变量名radius, area, 方法名computeArea.



命名规范

☞ 类名:

- 每一个单词的首字母大写。例如类名
`ComputeArea`

☞ 常量:

- 所有字母全大写，多个单词之间用下划线分开，例如：`PI`和`MAX_VALUE`



程序错误

☞ 语法错

- 编译器可以检查此类错误

☞ 运行错

- 程序异常退出

☞ 逻辑错

- 结果错误



语法错

```
public class ShowSyntaxErrors {  
    public static void main(String[] args) {  
        i = 30;  
        System.out.println(i + 4);  
    }  
}
```



运行错

```
public class ShowRuntimeErrors {  
    public static void main(String[] args) {  
        int i = 1 / 0;  
    }  
}
```



逻辑错

```
public class ShowLogicErrors {  
    // Determine if a number is between 1 and 100 inclusively  
    public static void main(String[] args) {  
        // Prompt the user to enter a number  
        String input = JOptionPane.showInputDialog(null,  
            "Please enter an integer:",  
            "ShowLogicErrors", JOptionPane.QUESTION_MESSAGE);  
        int number = Integer.parseInt(input);  
  
        // Display the result  
        System.out.println("The number is between 1 and 100, " +  
            "inclusively? " + ((1 < number) && (number < 100)));  
  
        System.exit(0);  
    }  
}
```

程序调试

逻辑错称为*bugs*。找到这些错误并改正，称为debug。
IDE都提供了调试功能，有关调试的技巧，需要在编程的实践中通过大量的练习才能熟练掌握。



JOptionPane方式输入

Java至少支持以下两种方式输入数据：

1. 使用Scanner类 (控制台输入);
2. 使用JOptionPane, 弹出对话框供输入。



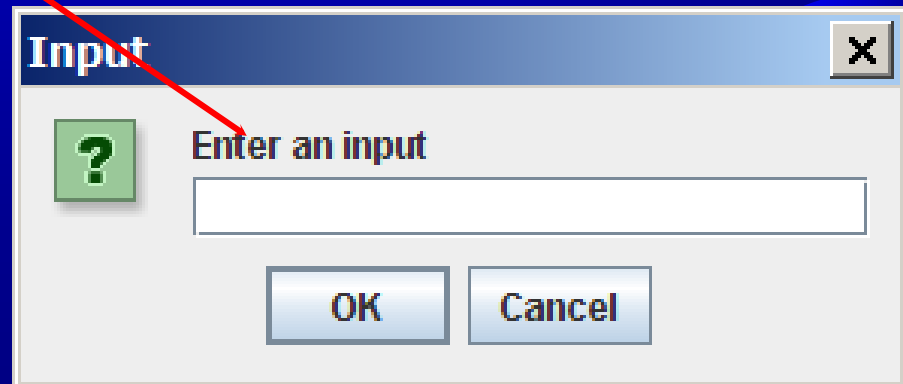
从对话框获取输入

首先要`import javax.swing.*;`

然后就可以这么用了：

```
String input = JOptionPane.showInputDialog(  
    "Enter an input");
```

注意得到的输入结果总是字符串，当然也可能是空串`null`，当用户点Cancel按钮或者点右上角那个叉的时候。



将字符串转成整型

如果用户在对话框中输入123，对话框会返回一个字符串“123”。如果你希望把这个串当作整数看待，只能通过下面这个方式做一个转换：

```
int intValue = Integer.parseInt(intString);
```

这里 intString 指的是数值形式的字符串，例如“123”，如果是其它不符合规范的串，例如“abc”或者是空串null，上述语句会导致程序挂掉。

一开始不要纠结程序挂掉的问题，老老实实输入符合规范的串就可以了。

将字符串转换成浮点数

如果希望将字符串转换成浮点数，可以这样：

```
double doubleValue = Double.parseDouble(doubleString);
```

这里doubleString 是形如浮点数的字符串，如“123.45”。同样的，不符合规范的串，也会导致程序挂掉。



