

第二章笔记

程序：计算机执行某些操作或解决某个问题而编写的一系列**有序指令的集合**。

Java 的特点：(1) Java 语言是面向对象的 (oop); (2) Java 的强类型机制、异常处理、垃圾的自动收集等是 Java 程序健壮性的重要保证；(3) Java 语言是跨平台的；(4) Java 语言是解释性的（区别于编译性语言：c/c++，区别在于：解释性语言，编译后的代码不能直接被机器执行，需要解释器来执行；编译性语言，编译后的代码可以直接被机器执行）

Java 编程先得到.java 文件，后面用到.class 文件，.class 的运行是在 JVM 上运行的。有了 JVM 就实现了 Java 程序的跨平台性。

Java 的核心机制-Java 虚拟机【JVM: Java virtual machine】(JVM 是包含在 JDK 中的)

- 1) JVM 是一个虚拟的计算机，具有指令集并使用不同的存储区域。负责执行指令，管理数据、内存、寄存器，包含在 JDK 中；
- 2) 对于不同的平台，有不同的虚拟机；
- 3) Java 虚拟机机制屏蔽了底层运行平台的差别，实现了“一次编译，到处执行”。

Test.java 到 Test.class 的过程叫做编译

JDK (Java development Kit:Java 开发工具包)

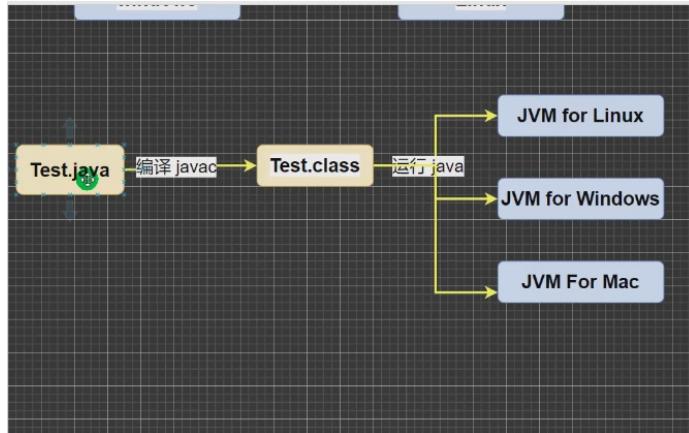
- 1.JDK = JRE + Java 的开发工具[java, javac, javadoc, javap 等]
- 2.JDK 是提供给 Java 开发人员使用的，其中包含了 Java 的开发工具，也包含了 JRE。

JRE (Java Runtime Environment: Java 运行环境)

- 1.JRE = JVM + Java 的核心类库[类]
- 2.包括 Java 虚拟机 (JVM) 和 Java 程序所需的核心类库等，若想运行一个开发完成的 Java 程序，计算机中只需要安装 JRE 即可。

JDK、JRE 和 JVM 的关系：

- 1.JDK = JRE + 开发工具集（如：Javac, java 编译工具等）
- 2.JRE = JVM + Java SE 标准类库
- 3.JDK = JVM + Java SE 标准类库 + 开发工具集
- 4.若只想运行开发后的.class 文件，只需要 JRE

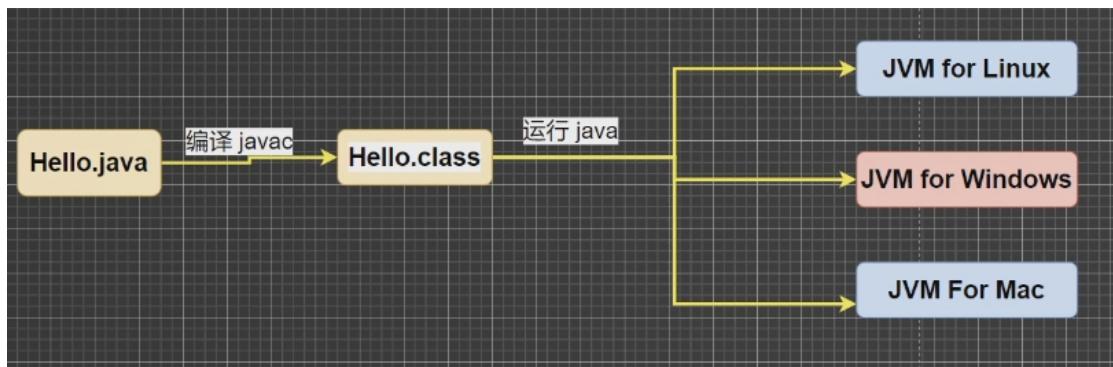


运行原理图

注意：在 cmd 内进行将.java 文件编译为.class 文件的时候注意要把文件的编码类型搞清楚，到底是 GBK 还是 UTF-8 或其他的编码类型（记得切换完后保存）

`javac Hello.java` 就可以将其转为.class 文件

转为.class 后，在 cmd 内执行时，**直接使用 `java Hello` 就可以执行**，不用写成 `java Hello.class`。原因是程序执行的是 Hello 这个类而不是这个.class。



运行原理图

小练习：输出 “wnegan is studying java!”

注：文件中有中文时，把文件的存储编码设置为 GBK 保存。文件->设置文件编码->GBK->重新保存。

```

public class Hello{
    //编写一个main方法
    public static void main(String[] args) {
        System.out.println("wenagn is studying java!");
    }
}

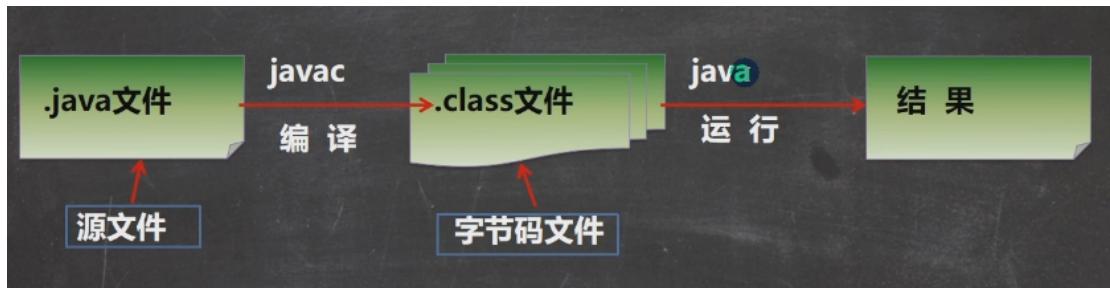
```

```

D:\javacode>javac Hello.java
D:\javacode>java Hello
wenagn is studying java!

```

Java 执行流程



Java 编译:

`javac Hello.java` (编译的时候必须把后缀带上)

1. 有了 java 源文件，通过编译器将其编译成 JVM 可以识别的字节码文件。
2. 在该源文件目录下，通过 `javac` 编译工具对 `Hello.java` 文件进行编译。
3. 若程序没有错误，没有任何提示，但在当前目录下会出现一个 `Hello.class` 文件，该文件为字节码文件，也可以执行 `java` 程序。

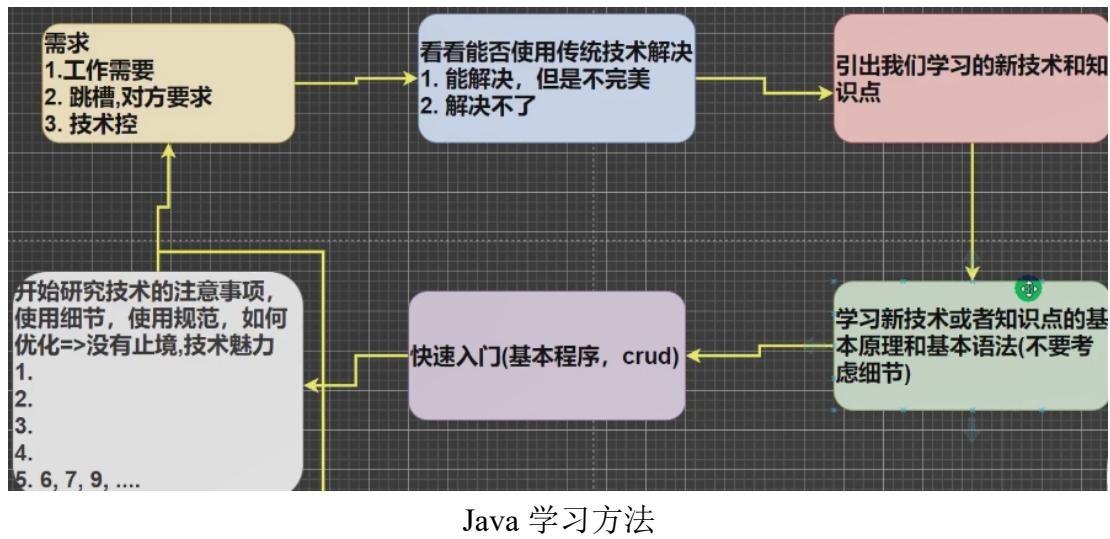
Java 运行:

1. 有了可运行的 `java` 程序 (`Hello.class` 字节码文件)
2. 通过运行工具 `java.exe` 对字节码文件进行执行，本质就是 `.class` 装到 `JVM` 机执行

(注意：对修改后的 `Hello.java` 源文件需要重新编译即再来一次 `javac Hello.java`，生成新的 `class` 文件后，再进行执行 `java Hello`，才能够生效)

Java 开发注意事项：(编译后，每一个类都对应一个.class 文件)

1. Java 源文件以 `.java` 为扩展名。源文件的基本组成部分是类 (class)，如本类中的 `Hello` 类；
2. Java 应用程序的执行入口是 `main()` 方法，有固定的书写格式：
`public static void main(String[] args){...}`
3. Java 语言**严格区分大小写**；
4. Java 方法由一条条语句构成，**每个语句以“;”结束**。
5. 大括号都是成对出现的，缺一不可。(养成习惯，先写 {} 再写代码)
6. 一个源文件中**最多只能有一个 public** 类，其他类的个数不限。
7. 如果源文件中包含一个 `public` 类，则**文件名必须按该类名命名！**
8. 一个源文件中最多只能有一个 `public` 类。其他类的个数不限，也可以将 `main` 反复写在非 `public` 类中，然后指定运行非 `public` 类，这样入口的方法就是非 `public` 的 `main` 方法。



Java 的常用转义字符:

(在控制台 cmd, 输入 tab 键, 可以实现命令的补全)

1. \t: 一个制表位, 实现对齐的功能
2. \n: 换行符
3. \\: 一个\ (要输出两个\\就写成\\\\)
4. \"": 一个"
5. \'": 一个'
6. \r: 一个回车。(回车区别于换行, 回车的意思是把当前行的光标放到当前行的开头 (原本应该在最后), 然后把\r后面的内容替换掉该行最前面的内容。例如: System.out.println("天天向上\r学习"); 则输出的结果为: 学习向上, 想要换行输出就用: System.out.println("天天向上\r\n学习");便可以输出成换行格式)

小练习

```
public class ChangeChar{
    //转义字符的使用
    public static void main(String[] args) {
        System.out.println("书名\t作者\t价格\t销量\n三国\t罗贯中\t120\t1000");
    }
}
```

```
D:\javacode>javac ChangeChar.java
D:\javacode>java ChangeChar
书名      作者      价格      销量
三国      罗贯中    120      1000
```

易犯错误汇总:

1. 找不到文件 (解决方案: 源文件名不存在或者写错, 或者是当前路径错误)

```
D:\javacode>javac ChangeCha.java
javac: 找不到文件: ChangeCha.java
用法: javac <options> <source files>
-hhelp 用于列出可能的选项
```

2. 主类名和文件名不一致（解决方案：声明为 public 的主类应与文件名一致，否则编译失败）

```
D:\javacode>javac ChangeCharExer01.java
ChangeCharExer01.java:2: 错误: 类ChangeCharExer0是公共的,
件中声明
public class ChangeCharExer0 {
```

3. 缺少分号（解决办法：编译失败，注意错误出现的行数，再到源代码中指定位置改错）

```
D:\javacode>javac ChangeCharExer01.java
ChangeCharExer01.java:7: 错误: 需要';'
    System.out.println("书名\t"
;
```

4. 最常犯语法错误：注意看错误信息，注意英文字符别写成中文字符，注意 **void**（正确）别写成 **viod**（错误）。

注释（comment）：用于注解说明程序的文字，注释提高了代码的可读性。

- 1) 单行注释：

格式：//注释文字

- 2) 多行注释：

格式：/*注释文字*/（中间可以写很多行注释内容）

（被注释的问题不会被 JVM（java 虚拟机）解释执行）

（多行注释里不允许有多行注释嵌套）

- 3) 文档注释：

注释内容可以被 JDK 提供的工具 **javadoc** 所解析，生成一套以网页文件形式体现的该程序的文档说明，一般写在类。

格式：

```
/**
 * @author wengan (javadoc 标签)
 * @version 1.0
 */
```

```

1  /**
2  * @author wengan
3  * @version 1.0
4  */
5
6
7 public class Comment02{
8
9     //编写一个main方法
10    public static void main(String[] args) {
11        System.out.println("wenagn is studying java!");
12    }
13 }
14

```

在 cmd 内输入： javadoc -d d:\\temp -author -version Comment02.java （其中 temp 是 D 盘中的一个文件夹）

可以使用的 javadoc 标签：

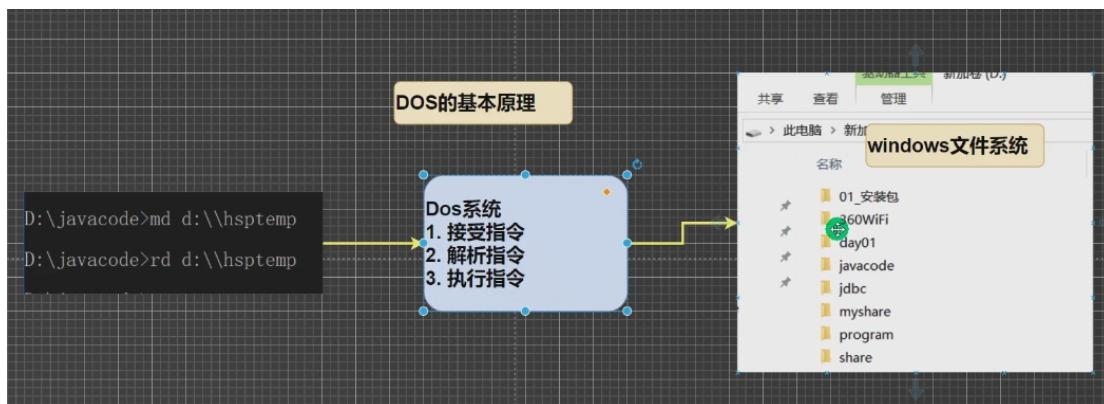
标签	描述	示例
@author	标识一个类的作者	@author description
@deprecated	指名一个过期的类或成员	@deprecated description
{@docRoot}	指明当前文档根目录的路径	Directory Path
@exception	标志一个类抛出的异常	@exception exception-name explanation
{@inheritDoc}	从直接父类继承的注释	Inherits a comment from the immediate superclass.
{@link}	插入一个到另一个主题的链接	{@link name text}
{@linkplain}	插入一个到另一个主题的链接，但是该链接显示纯文本字体	Inserts an in-line link to another topic.
@param	说明一个方法的参数	@param parameter-name explanation
@return	说明返回值类型	@return explanation
@see	指定一个到另一个主题的链接	@see anchor
@serial	说明一个序列化属性	@serial description
@serialData	说明通过 writeObject() 和 writeExternal() 方法写的数据	@serialData description

{@linkplain}	插入一个到另一个主题的链接，但是该链接显示纯文本字体	Inserts an in-line link to another topic.
@param	说明一个方法的参数	@param parameter-name explanation
@return	说明返回值类型	@return explanation
@see	指定一个到另一个主题的链接	@see anchor
@serial	说明一个序列化属性	@serial description
@serialData	说明通过 writeObject() 和 writeExternal() 方法写的数据	@serialData description
@serialField	说明一个 ObjectStreamField 组件	@serialField name type description
@since	标记当引入一个特定的变化时	@since release
@throws	和 @exception 标签一样。	The @throws tag has the same meaning as the @exception tag.
{@value}	显示常量的值，该常量必须是 static 属性。	Displays the value of a constant, which must be a static field.
@version	指定类的版本	@version info

Java 代码规范：

1. 类、方法的注释，要以 javadoc 的方式来写；
2. 非 Java Doc 的注释往往是着重告诉读者为什么代码这样写，如何修改以及需要注意什么问题等；
3. 使用 tab 操作，实现缩进，默认整体往右移动，shift+tab 整体往左移动；
4. 运算符和=两边习惯性各加一个空格，例： $2 + 4 * 4 = 18$ ；
5. 源文件使用 utf-8 编码；
6. 行宽度不超过 80 个字符；
7. 代码编写有次行风格和行尾风格。

DOS 命令（磁盘操作系统）



常用 DOS 命令

DOS命令(了解)

- DOS介绍

Dos: Disk Operating System 磁盘操作系统, 简单说一下windows的目录结构。 [原理图]

- 相关的知识补充: 相对路径, 绝对路径

- 常用的dos命令

1. 查看当前目录是有什么内容 dir

dir dir d:\abc2\test200

2. 切换到其他盘下: 盘符号 cd : change directory

案例演示: 切换到 c盘 cd /D c:

3. 切换到当前盘的其他目录下 (使用相对路径和绝对路径演示), ..\表示上一级目录

案例演示: cd d:\abc2\test200 cd ..\..\abc2\test200

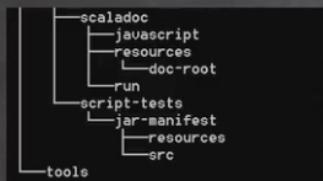
4. 切换到上一级:

案例演示: cd ..

5. 切换到根目录: cd \

案例演示: cd \

6. 查看指定的目录下所有的子级目录 tree



7. 清屏 cls [苍老师]

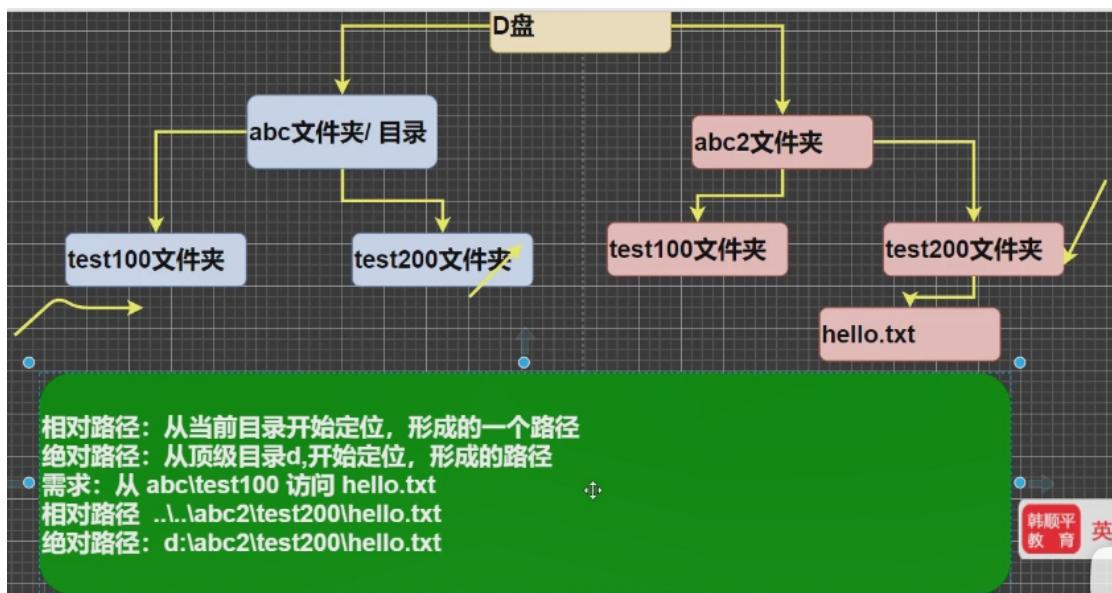
8. 退出DOS exit

9. 说明: 因为小伙伴后面使用DOS 非常少, 所以对下面的几个指令, 老韩给大家演示下, 大家了解即可 (md[创建目录],rd[删除目录],copy[拷贝文件],del[删除文件],echo[输入内容到文件],type,move[剪切]) => Linux

相对路径: 从当前目录开始定位, 形成的一个路径

绝对路径: 从顶级目录 d, 开始定位, 形成的路径

..\就是转到上一级目录



变量：是程序的基本组成单位

三个基本要素：类型+名称+值

例如：

```
class Test
{
    public static void main(String[] args)
    {
        int a = 1;//定义了一个变量，类型为 int(整型)，名称为 a，值为 1
        int b = 2;
        b = 89;
        System.out.println(a);
        System.out.println(b);
    }
}
```

变量使用的基本步骤：

- 1) 声明变量：int a;
- 2) 赋值：a = 80;
- 3) 使用 System.out.println(a);

变量的类型：

int（整型） double（浮点型） char（字符型：后面写的东西要用单引号） String
（字符串：后面写的东西要用双引号）

变量的注意事项：

1. 变量表示内存中的一个存储区域（不同的变量，其类型不同，占用空间的大小也不同）
2. 该区域有自己的名称[变量名]和类型[数据类型]
3. 变量在程序最开始必须先声明，后使用
4. 该区域的数据可以在同一类型范围内不断变化
5. 变量在同一个作用域内不能重名
6. 变量 = 变量名 + 数据类型 + 值

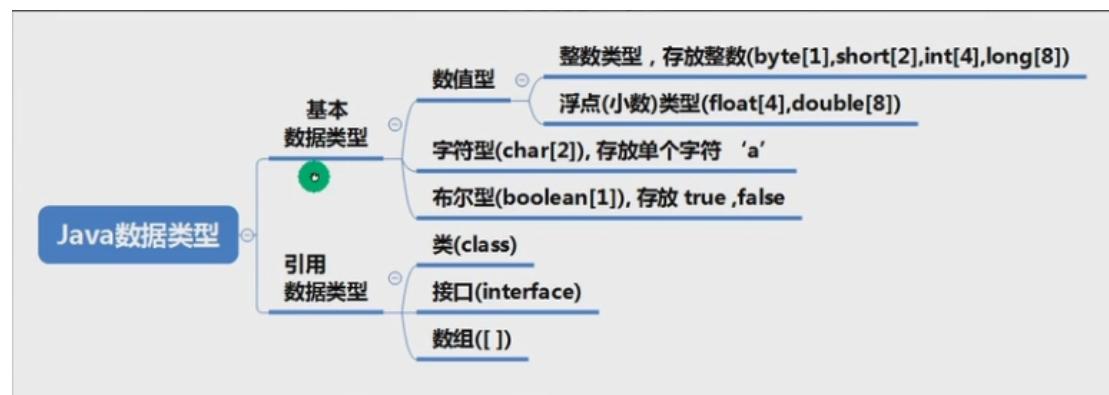
程序中“+”的使用：

1. 当两边都是数值型时，则做加法运算；
2. 当左右两边有一边（不管左边右边）为字符串时，则做拼接运算；
3. 运算顺序都是从左往右的。

```
//+号的使用
//
public class Var03{
    public static void main(String[] args){
        System.out.println(100 + 98);
        System.out.println("100" + 98);
        System.out.println(100 + 3 + "hello");
        System.out.println("hello" + 100 + 3);
    }
}
```

```
d:\javacode\chapter03>java Var03
198
10098
103hello
hello1003
```

Java 数据类型:



数值型: byte[字节:1个字节], short[短整型:2个字节], int[整型:4个字节], long[长整型:8个字节]

● 整型的类型

类 型	占 用 存 储 空 间	范 围
byte [字节]	1字节	-128 ~ 127 为啥存放的范围是这个=>二进制(二进制我们详解)
short [短整型]	2字节	-(2 ¹⁵) ~ 2 ¹⁵ -1 -32768 ~ 32767
int [整型]	4字节	-2 ³¹ ~ 2 ³¹ -1 -2147483648 - 2147483647
long [长整型]	8字节	-2 ⁶³ ~ 2 ⁶³ -1

浮点型: float[4], double[8]

字符型: char[2]

布尔型: boolean[1]存放 true, false

整数类型使用细节:

1. Java 各个整数类型有固定的 范围和字段长度, 不受具体 OS[操作系统]的影响,

以及保证 java 程序的可移植性；

2. Java 的整数常量（具体值）默认为 int 型，声明 long 型常量需在后面加 ‘L’；
3. Java 程序中变量常声明为 int 型，除非不足以表示大数，才使用 long
4. bit：计算机中的最小存储单位。byte：计算机中基本存储单元，1byte = 8bit。
注：long 类型有 $8 \times 8 = 64$ 个字节。

浮点类型：

● 浮点型的分类

类 型	占 用 存 储 空 间	范 围
单精度float	4字节	-3.403E38 ~ 3.403E38
双精度double	8字节	-1.798E308 ~ 1.798E308

1. 浮点数在机器中存放形式：浮点数 = 符号位 + 指数位 + 尾数位
2. 尾数部分可能丢失，造成精度损失（小数都是近似值）
3. 浮点类型默认为 double 型（8 个字节）
4. 与整数类型类似，java 浮点类型也有固定的范围的字段长度，不受具体 OS 的影响。[float 4 个字节 double 8 个字节]
5. Java 的浮点型常量（具体值）默认为 double 型，声明 float 型常量，需加 ‘f’ 或 ‘F’
6. 浮点型常量有两种形式：十进制数形式：如：5.12 512.0f .512（必须有小数点）；科学计数法：5.12e2[5.12*10 的 2 次方] 5.12E-2[5.12/10 的 2 次方]
7. 通常情况下，应该使用 double 型，因为它比 float 型更加精确

注：浮点数陷阱：2.7 和 8.1/3 的比较

```
//浮点数陷阱 2.7和8.1 / 3的比较
double n6 = 2.7;
double n7 = 8.1 / 3;// 8.1 / 3 = 2.7
System.out.println(n6);
System.out.println(n7);
```

```
d:\javacode\chapter03>javac FloatDetail.java
d:\javacode\chapter03>java FloatDetail
0.0512
2.7
2.6999999999999997
```

当我们对运算的结果是小数的进行相等判断时，要小心应该是以两个数的差值的绝对值，在某个精度范围内来判断。

浮点型为什么存在精度问题：

浮点型在计算机中，先将小数表示为2进制数，再将二进制用科学计数法来表示。

浮点型的内存结构分为3个部分，正负号，指数，尾数。

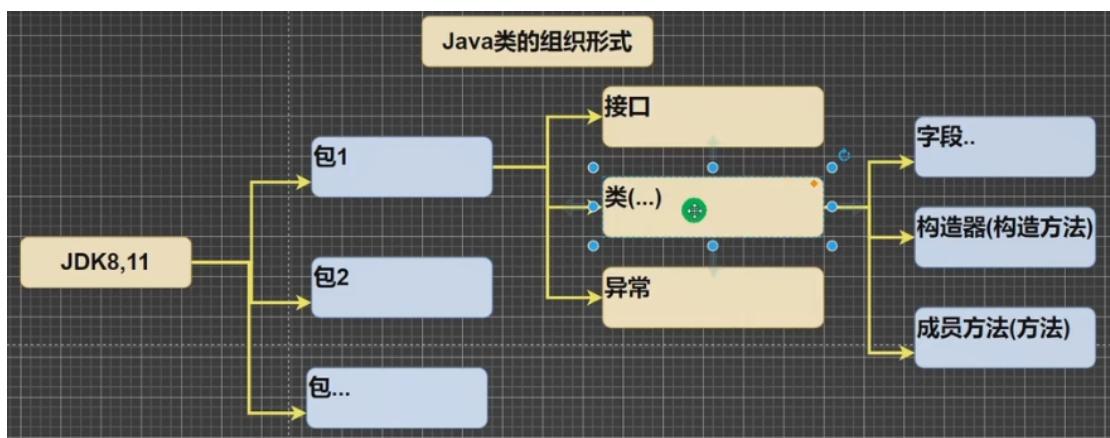
指数决定浮点型的取值范围，尾数决定浮点型的精度。

计算机无法存储一个无线循环的数，只能将这个数零舍一入（二进制版的四舍五入），这是浮点数存在误差的根本原因。

Java API 文档（Application Programming Interface，应用程序编程接口）是 java 提供的基本编程接口（java 提供的类还有相关的方法），中文在线文档：

<https://www.matools.com>

Java 类的组织形式：



字符类型（char）：

字符类型可以表示单个字符，字符类型为 char，char 是两个字节（可以存放汉字），
多个字符用字符串 String。

字符类型可以直接存放一个数字，当输出其对应的数字时，会输出数字对应的字符

快捷键：Ctrl+shift+D 可以直接把光标所在行复制到下一行

快捷键：Ctrl+shift+k 可以直接删除整行

```
1 //字符串char类型的使用
2 //
3
4 public class Char01{
5     public static void main(String[] args){
6         char c1 = 'a';
7         char c2 = '\t';
8         char c3 = '文';
9         char c4 = 97;//字符类型可以存放一个数字
10        System.out.println(c1);
11        System.out.println(c2);
12        System.out.println(c3);
13        System.out.println(c4);//当输出c4的时候，就会输出97表示的数字=>编码的概念
14    }
15 }
```

```
D:\javacode\chapter03>java Char01
'a'
文
a
D:\javacode\chapter03>
```

字符类型的使用细节：

1. 字符常量是用单引号('')括起来的单个字符，比如：char c1 = 'a'; char c2 = '中'; char c3 = '9';
2. Java 中还允许使用转义字符'\'来将其后的字符转变为特殊字符型常量。例如：char c3 = '\n'; // '\n' 表示换行符；
3. 在 java 中，char 的本质是一个整数，在输出时，是 unicode 码对应的字符 <http://tool.chinaz.com/Tool/Unicode.aspx>；
4. 可以直接给 char 赋一个整数，然后输出时，会按照对应的 unicode 字符输出 [97->a]；
5. char 类型是可以进行运算的，相当于一个整数，因为它都对应有 Unicode 码。
(注：要输出对应的数字，可以 (int) 字符)。

```
//字符串char类型的使用
//

public class Char01{
    public static void main(String[] args){
        char c1 = 'a' + 3;//a对应97,所以结果为100
        char c2 = '\t';
        char c3 = '文';
        char c4 = 97;//字符类型可以存放一个数字，输出就是输出该数字对应的字母或者其他
        System.out.println(c1);
        System.out.println(c2);
        System.out.println(c3);
        System.out.println(c4);//当输出c4的时候，就会输出97表示的数字=>编码的概念
        System.out.println('a' + 3);//当输出c4的时候，就会输出97表示的数字=>编码的概念
        //
        char c5 = 'b' + 1;//b对应的ASCII码为98
        System.out.println((int)c5);//输出99
        System.out.println(c5);//输出99对应的字符c
    }
}
```

字符类型(char)

● 字符类型本质探讨

1. 字符型 存储到 计算机中，需要将字符对应的码值（整数）找出来，比如'a'

存储：'a' ==> 码值 97 ==> 二进制(110 0001) ==> 存储

读取：二进制(110 0001) ==> 97 ==> 'a' ==> 显示

2. 字符和码值的对应关系是通过字符编码表决定的(是规定好)

● 介绍一下字符编码表 [sublime测试]

ASCII (ASCII 编码表 一个字节表示，一个128个字符，实际上一个字节可以表示256个字符,只用128个)
Unicode (Unicode 编码表 固定大小的编码 使用两个字节来表示字符，字母和汉字统一都是占用两个字节，这样浪费空间)

utf-8 (编码表，大小可变的编码 字母使用1个字节，汉字使用3个字节)

gbk (可以表示汉字，而且范围广，字母使用1个字节，汉字2个字节)

gb2312 (可以表示汉字，gb2312 < gbk)

big5 码(繁体中文, 台湾, 香港)



ASCII 编码十分有限，只用 256 种，所以有用 Unicode 码。

utf-8 编码：

1. utf-8 是互联网上使用最广的一种 Unicode 的实现方式；
2. utf-8 是一种变长的编码方式。它可以用 1-6 个字节表示一个符号，根据不同的符号而变化字节长度；
3. 使用大小可变的编码字母占 1 个字节，汉字占 3 个字节。

布尔类型：boolean

1. 布尔类型数据只允许取值 true 和 false，没有 null；
2. boolean 类型占 1 个字节；
3. boolean 类型适用于逻辑运算，一般用于程序流程控制：
if 条件控制语句； while 循环控制语句； do-while 循环控制语句； for 循环语句。
4. 使用细节：不可以用 0 或者非 0 的整数类替代 true 和 false（和 c 不同）。

```
1 //boolean类型
2 //
3 public class Boolean01{
4     public static void main(String[] args){
5         //定义一个boolean变量
6         boolean pass = false;
7         if(pass == true){
8             System.out.println("考试通过");
9         }else if (pass == false) {
10            System.out.println("考试失败");
11        }
12    }
13
14 }
```

```

D:\javacode\chapter03>java Boolean01
考试通过

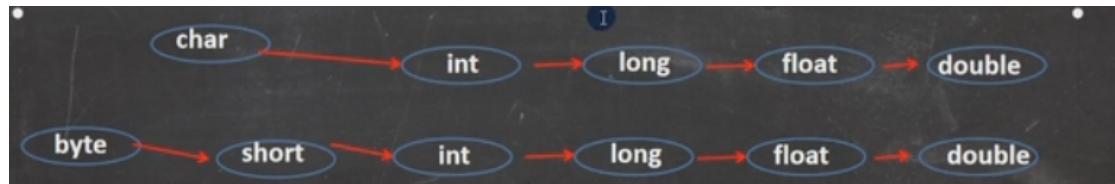
D:\javacode\chapter03>javac Boolean01.java
未找到 D:\javacode\chapter03>java Char01
d
文
a
100
99
c

D:\javacode\chapter03>java Boolean01
考试失败

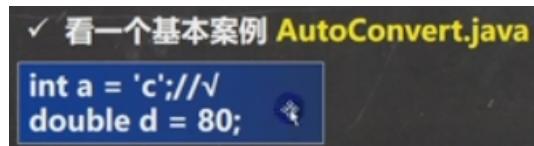
```

基本数据类型转化：

在 java 中，精度小的类型自动转换为精度大的数据类型，这个就是自动类型转换。数据类型按精度排序为：



低精度类型的可以直接赋值给高精度类型。(高的不能给小的，小的可以给大的)



注：(1) 如果用数值来赋值，计算机会先判断数值是否在该类型的范围内，如果是，那就正确；如果不是，那就会出现编译错误；

(2) 如果用变量来赋值，则会先判断变量是什么类型，如果该变量类型的精度比所赋的变量的精度大，错误；反之，如果精度比所赋的变量的精度小，那就可能。

基本数据类型转换细节：

1. 有多种类型的数据混合运算时，系统首先自动将所有数据转换成容量最大的那种数据类型，然后再进行运算；
2. 当我们把**精度（容量）大的数据类型赋值给精度（容量）小的数据类型时，就会报错**，反之就会进行自动类型转换；
3. **(byte, short) 和 char 之间不会相互自动转换；**
4. **byte, short, char 三者之间可以计算，在计算时首先转换为 int 类型；**
5. boolean 不参与转换；
6. 自动提升原则：表达式结果的类型自动提升为操作数中的最大类型。

强制类型转换：

为自动类型转换的逆过程，将容量大的数据类型转换为容量小的数据类型，在使

用的时候要加上强制转换符 ()，但要注意可能造成精度降低或溢出。

例如：

```
int i = (int)1.9;//这样可能会造成 1.9 的小数位即 0.9 的丢失（精度损失）
System.out.println(i);
int j = 100;
byte b1 = (byte)j;
System.out.println(j);//造成数据溢出
```

细节说明：

1. 当进行数的大小从大->小时，就需要使用到强制转换；
2. 强转符号只针对最近的操作数有效，往往使用小括号来提升优先级
int x = (int)10*3.5+6*1.5;//错误
int y = (int)(10*3.5+6*1.5); //正确
3. char 类型可以保存 int 的常量值，但不能保存 int 的变量值，就需要强转
char c1 = 100;//ok
int m = 100;//ok
char c2 = m;//错误
char c3 = (char)m;//ok=>输出 100 对应的字符（因为其是 char 类型）
4. byte 和 short 类在运行时，当做 int 处理。进行计算后其类型就会变为 int 类型。
例如：

```
short s = 12;//ok
s = s - 9;//错误， s-9 之后就变成 int，再赋值给 short 类型的 s 会溢出
```

```
byte b = 10;//ok
b = b + 11;//错误， int->byte
b = (byte)(b+11); //ok

char c = 'a';//ok
int i = 16;//ok
float d = .314F;//ok
double result = c + i + d;//ok-> c + i + d 变为最大精度为 float<double
```

```
byte b = 16;//ok
short s = 14;//ok
short t = s + b;//错误， s + b 变为 int 型，不能赋值给 short，溢出了
```

基本数据类型和 String 类型的转换：

1. 基本数据类型转换为 String 类型：将基本类型的值+””即可

```
//  
public class StringtoBasic{  
    public static void main(String[] args){  
        int n1 = 100;  
        float f1 = 1.1F;  
        double d1 = 4.4;  
        boolean b1 = true;  
        String s1 = n1 + " ";  
        String s2 = f1 + " ";  
        String s3 = d1 + " ";  
        String s4 = b1 + " ";  
        System.out.println(s1 + s2 + s3 + s4); //+ " "  
    }  
}
```

2. String 类型转换为基本数据类型: 通过基本类型的包装类调用 parseXX 方法即可 (字符串转为字符的意思是把字符串的第一个字符得到)

```
//String类型转基本数据类型  
String s5 = "123";  
int num1 = Integer.parseInt(s5); //将s5转换为int类型  
double num2 = Double.parseDouble(s5); //将s5转换成double类型  
float num3 = Float.parseFloat(s5); //将s5转换成float类型  
long num4 = Long.parseLong(s5); //将s5转换成long类型  
byte num5 = Byte.parseByte(s5);  
boolean b = Boolean.parseBoolean("true");  
short num6 = Short.parseShort(s5);  
  
System.out.println(s5 + 1);
```

注意事项:

- 在将 String 类型转换成基本数据类型时, 要确保 String 类型能够转成有效的数据, 比如可以把“123”转成一个整数, 但是不能把“hello”转成一个整数;
- 若格式不正确, 就会抛出异常, 进而程序终止

第四章 运算符笔记

运算符用以表示数据的运算、赋值和比较等。包括: 算术运算符、赋值运算符、关系运算符(比较运算符)、逻辑运算符、位运算符(需要二进制基础)、三元运算符。

算术运算符:

● 算术运算符一览

运算符	运算	范例	结果
	正号	+7	7
-	负号	b=11; -b	-11
+	加	9+9	18
-	减	10-8	2
*	乘	7*8	56
/	除	9/9	1
%	取模(取余)	11%9	2
++	自增 (前) : 先运算后取值	a=2;b=++a;	a=3;b=3
++	自增 (后) : 先取值后运算	a=2;b=a++;	a=3;b=2
--	自减 (前) : 先运算后取值	a=2;b=- - a	a=1;b=1
--	自减 (后) : 先取值后运算	a=2;b=a- -	a=1;b=2
+	字符串相加	"hsp"+"edu"	"hsp edu"

算术运算符的使用：

1. +, -, *, ,, %, ++, --
2. 自增：++：作为独立语句使用时，前++和后++都完全等价于 $i=i+1$ ；作为表达式使用时，前++：++i 先自增后赋值，后++：i++先赋值后自增；
3. --, +, -, *是一个道理，可以进行类比推理。

注意：

//取余=取模

```
//java 中取模的本质：一个公式：a % b = a - a / b * b
// -10 % 3 = -10 - (-10) / 3 * 3 = -1
// 10 % (-3) = 10 - 10 / (-3) * (-3) = 1
// -10 % (-3) = -10 - (-10) / (-3) * (-3) = -1
```

```

//取余=取模
//java中取模的本质：一个公式：a % b = a - a / b * b
//-10 % 3 = -10 - (-10) / 3 * 3 = -1
//10 % (-3) = 10 - 10 / (-3) * (-3) = 1
// -10 % (-3) = -10 - (-10) / (-3) * (-3) = -1
System.out.println(10 % 3); //得到1
System.out.println(-10 % 3); //得到-1
System.out.println(10 % -3); //得到1
System.out.println(-10 % -3); //得到-1
//++的使用
//
int i = 10;
i++; //自增 => i = i + 1 => 11
++i; //自增 => i = i + 1 => 12
System.out.println("i = " + i);
//前++：++i先自增后赋值；
//后++：i++先赋值后自增；
int j = 8;
//int k = ++j; //等价于j=j+1;k=j;
int k = j++; //等价于k=j;j=j+1;
System.out.println("j = " + j + "k = " + k);

```

注意：要考虑数学公式和 java 语言的特性

```

int i = 1;
i = i++; //这里会使用临时变量：(1) temp=i; (2) i = i+1;(3) i = temp;前++先赋值后自增
System.out.println(i); //结果为 1

```

```

int i = 1;
i = ++i; //这里会使用临时变量：(1) i = i+1; (2) temp=i;(3) i = temp;后++先自增后赋值
System.out.println(i); //结果为 2

```

```

int i1 = 10;
int i2 = 20;
int i = i1++; // (1) temp=i1; (2) i1 = i1+1;(3) i = temp;=>10
System.out.println("i = " + i); //10
System.out.println("i1 = " + i1); //11
i = --i2; // (1) i2 = i2-1; (2) temp=i2;(3) i = temp;=>21
System.out.println("i = " + i); //19

```

```

System.out.println("i2 = " + i2);//19

//把 59 天变成周和剩余的天数
//
public class ArithmeticOperatorExercise{
    public static void main(String[] args){
        //一周为 7 天
        int i1 = 59;
        int i2 = 7;
        int w = i1 / i2;
        int d = i1 % i2;
        System.out.println("距离放假还有 " + w + " 周 " + d + " 天");
        //华式温度转摄氏温度，公式为 5/9*(华氏温度-100)，求出 234.5 华式温度对应的摄氏温度
        double t1 = 234.5;
        double t2 = 5.0 / 9 * (t1 - 100);
        System.out.println("234.5 华式温度对应的摄氏温度为" + t2);
    }
}

```

关系运算符（比较运算符）

1. 关系运算符的结果都是 boolean 型，也就是 true 或 false；
2. 关系表达式经常在 if 结构或循环结构的条件中；

● 关系运算符一览

运算符	运算	范例	结果
<code>==</code>	相等于	<code>8==7</code>	false
<code>!=</code>	不等于	<code>8!=7</code>	true ✓
<code><</code>	小于	<code>8<7</code>	false
<code>></code>	大于	<code>8>7</code>	true
<code><=</code>	小于等于	<code>8<=7</code>	false
<code>>=</code>	大于等于	<code>8>=7</code>	true
<code>instanceof</code>	检查是否是类的对象	<code>"hsp" instanceof String</code>	true

3. 关系运算符的结果都是 boolean 型，也就是 true 或 false；
4. 关系运算符组成的表达式，称为关系表达式；
5. 比较运算符 “==” 别误写成 “=”。

逻辑运算符（最终结果也是 boolean 值）

- (1) 短路与`&&`，短路或`||`，取反`!`；

(2) 逻辑与&，逻辑或|，逻辑异或

a	b	a&b	a&&b	a b	a b	!a	a^b
true	true	true	true	true	true	false	false
true	false	false	false	true	true	false	true
false	true	false	false	true	true	true	true
false	false	false	false	false	false	true	false

● 逻辑运算符一览

✓ 说明逻辑运算规则：

1. a&b : & 叫逻辑与：规则：当a 和 b 同时为true ,则结果为true, 否则为false
2. a&&b : && 叫短路与：规则：当a 和 b 同时为true ,则结果为true,否则为false
3. a|b : | 叫逻辑或，规则：当a 和 b ，有一个为true ,则结果为true,否则为false
4. a||b : || 叫短路或，规则：当a 和 b ，有一个为true ,则结果为true,否则为false
5. !a : 叫取反，或者非运算。当a 为true, 则结果为false, 当 a 为false是，结果为true
6. a^b: 叫逻辑异或，当 a 和 b 不同时，则结果为true, 否则为false

&&和&&使用的区别：

1. 对于&&短路与而言，如果第一个条件为 false, 后面的条件将不会再判断执行；
2. 对于&逻辑与而言，如果第一个条件为 false，后面的条件依然会判断执行；

```
//区别
int a = 4;
int b = 9;
//对于&&短路与而言，如果第一个条件为false ,后面的条件不再判断
//对于&逻辑与而言，如果第一个条件为false ,后面的条件仍然会判断
if(a < 1 & ++b < 50) {
    System.out.println("ok300");
}
System.out.println("a=" + a + " b=" + b); // 4 10
```

3. 在开发中，通常使用&&短路与，效率高。

|和|使用的区别：

- 1.对于||短路或而言，如果第一个条件为 true，后面的条件将不会再判断执行,最终结果为 true;
- 2.对于|逻辑或而言，如果第一个条件为 true，后面的条件依然会判断执行。
- 3.开发中常用短路或||，效率高。

取反操作和逻辑异或操作：

- ! 为取反操作: T 变 F, F 变 T;
- $a \wedge b$ 称为逻辑与或, 当 a 和 b 不同时, 则结果为 true, 否则为 false。

● 练习题1请写出每题的输出结果

<pre>int x = 5; int y=5; if(x++==6 & ++y==6){ //逻辑与 x = 11; } System.out.println("x="+x+",y="+y);</pre>	<pre>int x = 5,y = 5; if(x++==6 && ++y==6){ x = 11; } System.out.println("x="+x+",y="+y);</pre>		
<pre>int x = 5,y = 5; if(x++==5 ++y==5){ x=11; } System.out.println("x="+x+",y="+y);</pre>	<pre>int x = 5,y = 5; if(x++==5 ++y==5){ x=11; } System.out.println("x="+x+",y="+y);</pre>		
1. x=6,y=6	2. x=6,y=5	3. x=11,y=6	4. x=11,y=5

● 练习题2请写输出结果

```
boolean x=true;
boolean y=false;
short z=46;
if( (z++==46)&& (y=true) ) z++;
if((x=false) || (++z==49)) z++;
System.out.println("z="+z);
```

z=50

赋值运算符: 就是将某个运算后的值, 赋给指定的变量。

有基本赋值运算符 (=) 和复合赋值运算符 (+=, -=, *=, /=, %=)

$a+b \Rightarrow a=a+b$ $a-b \Rightarrow a=a-b$

赋值运算符的特点:

- 运算从右往左, 即先把等号右边的值获取到之后再赋给等号左边的变量;
- 赋值运算符的左边只能是变量, 右边可以是变量、表达式或常量值;
- 复合赋值运算符会进行类型转换: byte b=2; $b+=3$; $b++$ 。(因为进行运算后会变成 int 类型 (类型强转), 但在复合运算符中会自动转换成 byte 再赋值给 b)。

三元运算符:

基本语法格式: 条件表达式?表达式 1: 表达式 2;

运算规则:

1. 如果条件表达式为 true，运算后的结果是表达式 1；
2. 如果条件表达式为 false，运算后的结果是表达式 2。

```
1 //三元运算符使用
2
3 public class TernaryOperator {
4
5     //编写一个main方法
6     public static void main(String[] args) {
7
8         int a = 10;
9         int b = 99;
10        // 解读
11        // 1. a > b 为 false
12        // 2. 返回 b--，先返回 b的值,然后在 b-1
13        // 3. 返回的结果是99
14        int result = a > b ? a++ : b--;
15        System.out.println("result=" + result);
16
17    }
18 }
```

3. 表达式 1 和表达式 2 要为可以赋给接受变量的类型（或可以自动转换）

```
//编写一个main方法
public static void main(String[] args) {
    //表达式1和表达式2要为可以赋给接收变量的类型(或可以自动转换)
    int a = 3;
    int b = 8;
    int c = a > b ? (int)1.1 : (int)3.4;//可以的
    double d = a > b ? a : b + 3;//可以的, 满足 int -> double
}
```

4. 三元运算符可以转成 if--else 语句： int res = a > b ? a++ : --b; => if(a>b){res = a++}else{res = b--}

运算符的优先级：（从上往下优先级依次降低）

	. 0 0 ; ,
R->L	++ -- ~ !(data type)
L->R	* / %
L->R	+ -
L->R	<< >> >>> 位移
L->R	< > <= >= instanceof
L->R	== !=
L->R	&
L->R	^
L->R	韩顺平 教育 英
L->R	&&
L->R	
L->R	? :
R->L	= *= /= %=
	+= -= <<= >>=
	>>>= &= ^= =

只有单目运算符、赋值运算符是从右往左运算的。

标识符命名规则：

- 由 26 个英文字母大小写，0-9，或\$组成；
- 数字不可以开头，int 3ab = 1;//错误；
- 不可以使用关键字和保留字（比如 public， static 等），但能包含关键字和保留字；
- Java 中严格区分大小写，长度无限制。int totalNum = 10; int n = 90;
- 标识符不能包含空格。

● 判断下面变量名是否正确

```

hsp //ok
hsp12 //ok
1hsp //错误, 数字不能开头
h-s // 错误, 不能有 -
x h // 错误, 有空格
•h$4 // ok
class //错误, class 关键字
int // 错误,int 是关键字
double //错误,double是关键字
public //错误,public 是关键字
static //错误,static是关键字
goto //错误, goto是保留字
stu_name //ok

```

标识符命名规范：

● 标识符命名规范[更加专业]

1. 包名：多单词组成时所有字母都小写：aaa.bbb.ccc //比如 com.hsp.crm
2. 类名、接口名：多单词组成时，所有单词的首字母大写：XxxYyyZzz [大驼峰]
比如：TankShotGame
3. 变量名、方法名：多单词组成时，第一个单词首字母小写，第二个单词开始每个单词首字母大写：xxxYyyZzz [小驼峰，简称 驼峰法]
比如：tankShotGame
4. 常量名：所有字母都大写。多单词时每个单词用下划线连接：XXX_YYY_ZZZ
比如：定义一个所得税率 TAX_RATE
5. 后面我们学习到类，包，接口，等时，我们的命名规范要这样遵守，更加详细的看文档。

关键字：被 java 语言赋予了特殊含义，用做专门用途的字符串（单词）。其特点为：关键字中所有的字母都为小写。

用于定义数据类型的关键字				
class	interface	enum	byte	short
int	long	float	double	char
boolean	void			
用于定义数据类型值的关键字				
true	false	null		
用于定义流程控制的关键字				
if	else	switch	case	default
while	do	for	break	continue
return				

保留字：现有的 java 版本尚未使用，但以后的版本可能会作为关键字使用（在命名是要尽量避免使用这些保留字）：byValue, cast, future, generic, inner, operator, outer, rest, var, goto, const。

键盘输入语句：

步骤：

- 1) 导入该类的所在包，java.util.*
- 2) 创建该类对象（声明变量）
- 3) 调用里面的功能