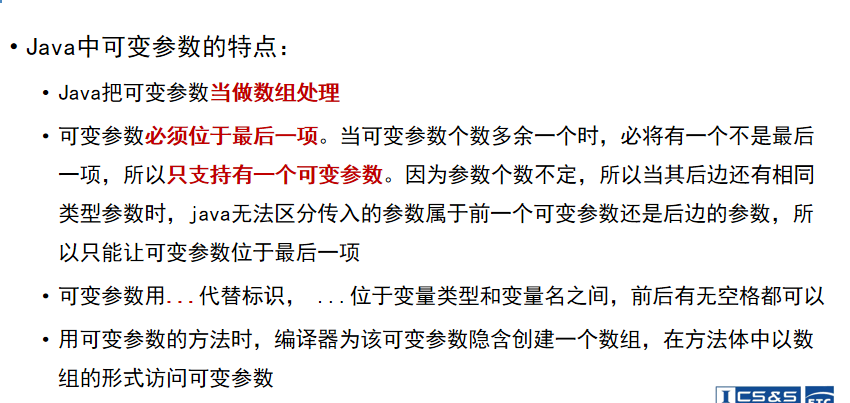
# 回顾

1. String的特性:不变性:值在创建之后不能再修改
2. StringBuffer和StringBuilder:可变的字符串,效率高,以后在拼接字符串的时候,尽量 使用二者(append);单线程用StringBuilder,线程不安全;多线程的情况下,使用StringBuffer
3. 方法参数传递:基本数据类型/String作为参数:传递的是值的一个副本给形参,在方法内对形参做操作,不影响实参的值->值传递

数组,类引用类型作为参数:传递的是引用名的值其实是堆的地址给形参,则实参和形参2个引用名指向堆中的同一个对象,在方法体中堆改地址中的值改变就会影响实参.

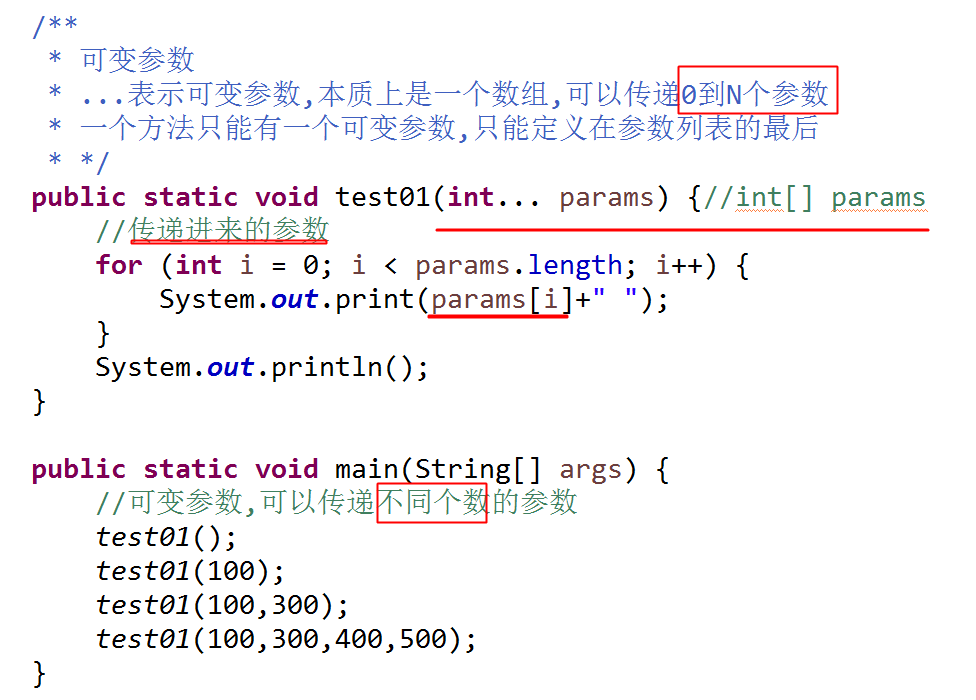
# 可变参数

Java1.5增加了可变参数特性，适用于参数个数不确定，类型确定的情况

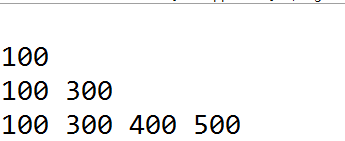


int... pa

pa[1]



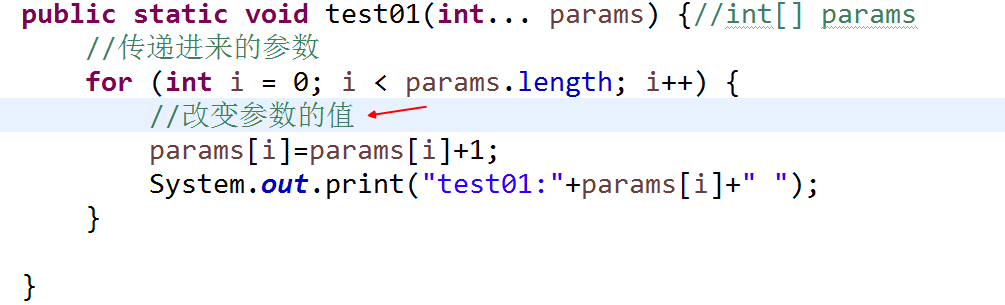
测试类结果:



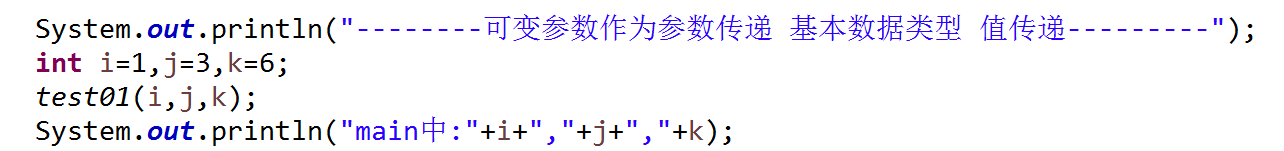
# 方法的参数传递

## 可变参数作为参数

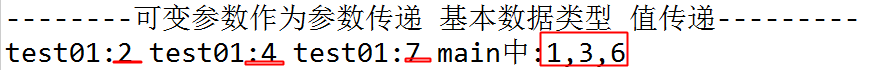
### 1基本数据类型作为参数【值传递】：



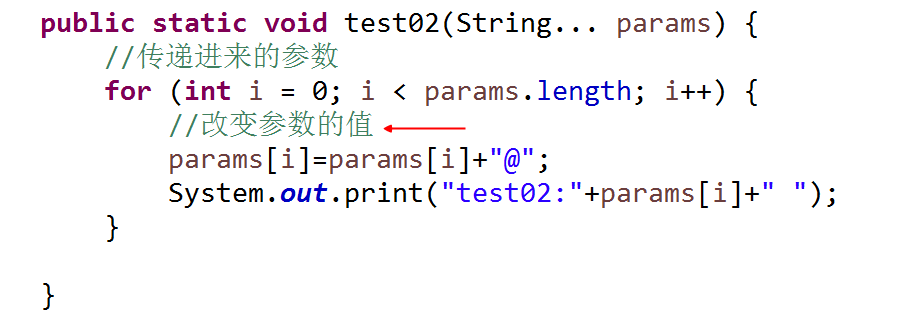
测试类:



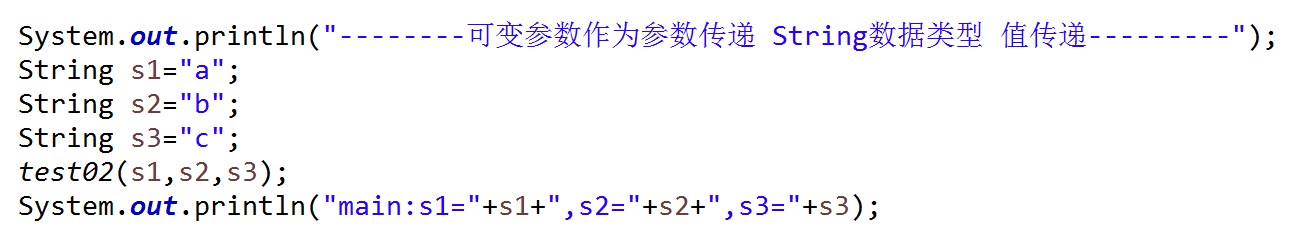
测试结果：



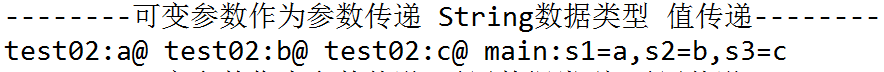
### 2 String可变参数作为传递：【值传递】



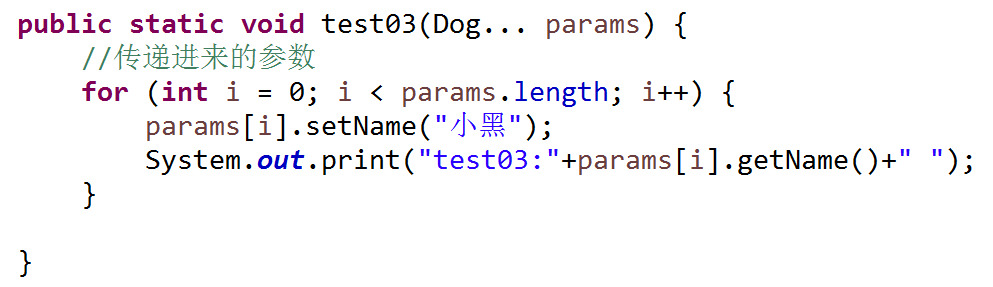
测试：



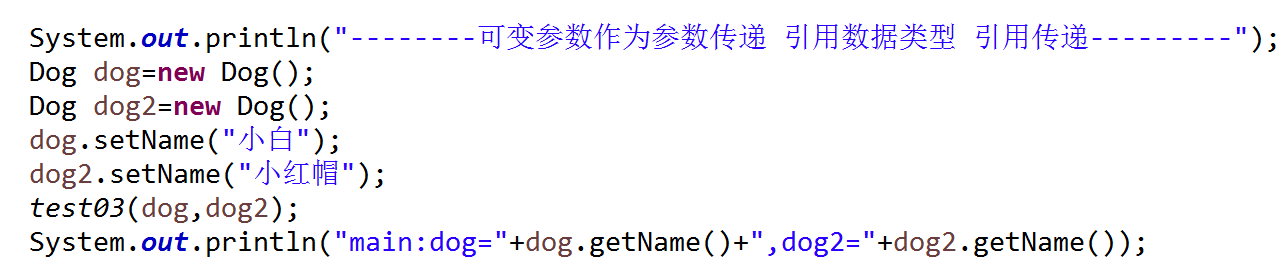
结果：



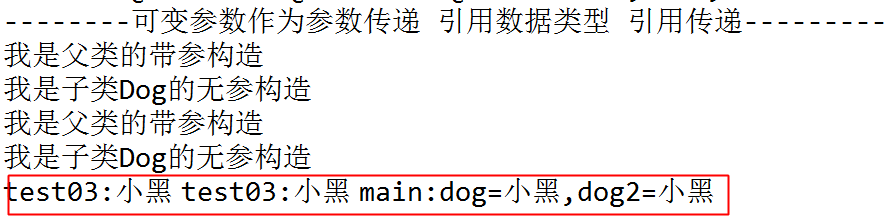
### 3类可变参数作为传递：【引用传递】



测试:



测试结果:



**可变参数作为参数，到底是如何传递要取决于可变参数的数据类型。**

## 总结：值传递与引用传递

值传递：

(形式参数类型是基本数据类型)：

方法调用时，实际参数把它的值传递给对应的形式参数，形式参数只是用实际参数的值初始化自己的存储单元内容，是两个不同的存储单元，所以方法执行中形式参数值的改变不影响实际参数的值。

引用传递

(形式参数类型是引用数据类型参数)：

也称为传地址。方法调用时，实际参数是对象(或数组)，这时实际参数与形式参数指向同一个地址，在方法执行中，对形式参数的操作实际上就是对实际参数的操作，这个结果在方法结束后被保留了下来，所以方法执行中形式参数的改变将会影响实际参数。

# Math

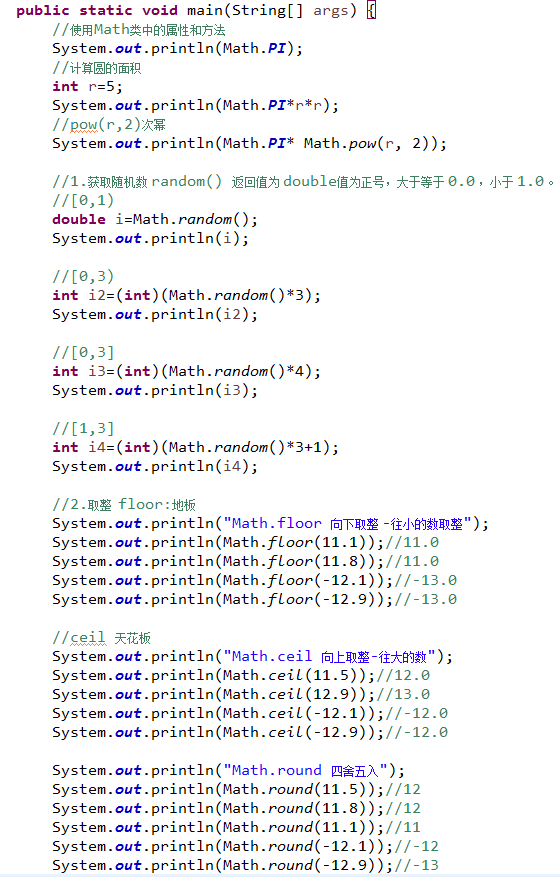
Math 类包含用于执行基本数学运算的方法，如初等指数、对数、平方根和三角函数。

|  |  |
| --- | --- |
| **字段摘要** | |
| static double | **[E](mk:@MSITStore:E:\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/lang/../../java/lang/Math.html" \l "E)**            比任何其他值都更接近 *e*（即自然对数的底数）的 double 值。 |
| static double | **[PI](mk:@MSITStore:E:\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/lang/../../java/lang/Math.html" \l "PI)**            比任何其他值都更接近 *pi*（即圆的周长与直径之比）的 double 值。 |

常见的方法

|  |  |
| --- | --- |
| static double | **[ceil](mk:@MSITStore:E:\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/lang/../../java/lang/Math.html" \l "ceil(double))**(double a)            返回最小的（最接近负无穷大）double 值，该值大于等于参数，并等于某个整数。 |
| static double | **[floor](mk:@MSITStore:E:\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/lang/../../java/lang/Math.html" \l "floor(double))**(double a)            返回最大的（最接近正无穷大）double 值，该值小于等于参数，并等于某个整数。 |
| static long | **[round](mk:@MSITStore:E:\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/lang/../../java/lang/Math.html" \l "round(double))**(double a)            返回最接近参数的 long。 |
| static double | **[random](mk:@MSITStore:E:\\API帮助文档\\jdk1.6\\JDK_API_1_6.CHM::/java/lang/../../java/lang/Math.html" \l "random())**()     返回带正号的 double 值，该值大于等于 0.0 且小于 1.0。 |

**求随机数:取值范围0~1 ,1取不到**



随机数的应用:

1. 某些业务中可能有所谓的幸运抽奖，随机数的匹配(产生一个随机数，刚好和你的生日匹配)

2. 模拟彩票生成算法，猜数字游戏

3. 人物猜拳的游戏;

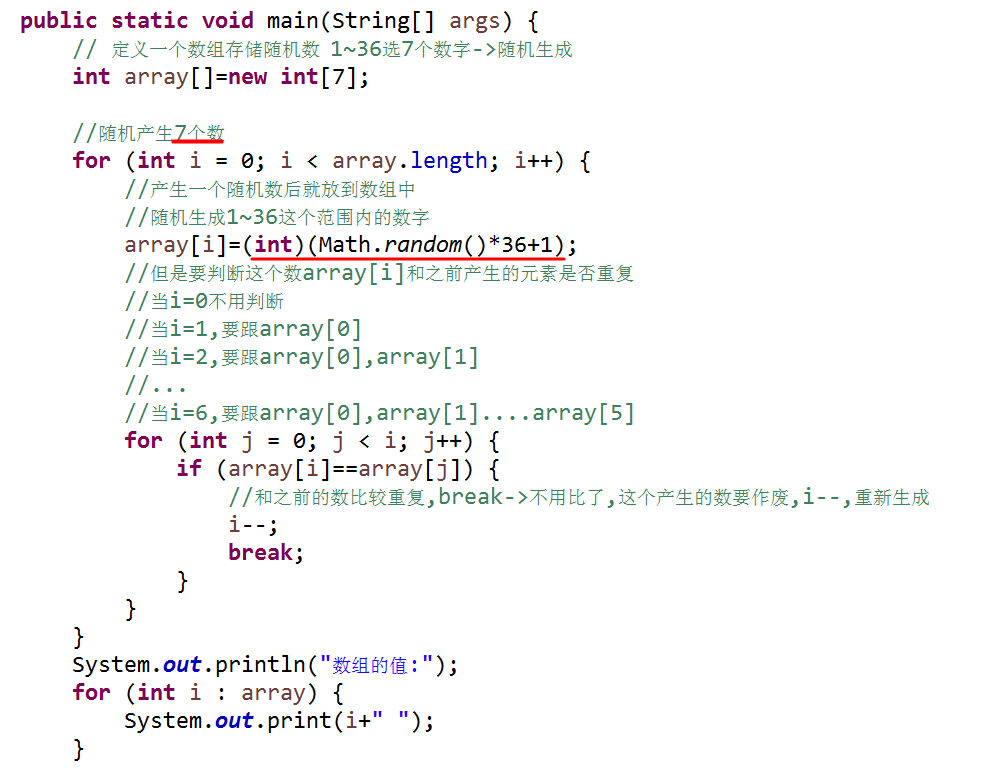
## 简单的模拟彩票生成器

36选7,1到36个球，每次产生一个，一共产生7个号码，不能重复；

产生1~36数字:

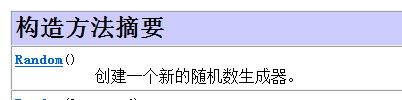
生成:1~36的随机数:

(int)(Math.radom()\*36+1)



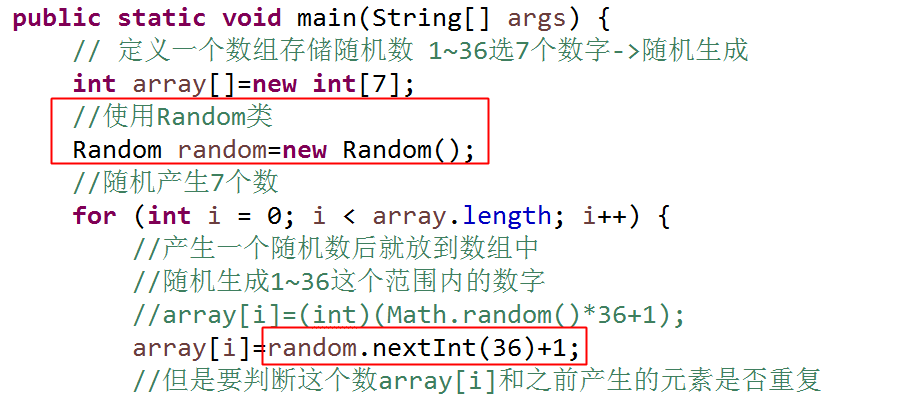
# Random类

此类的实例用于生成伪随机数流



## 直接使用nextInt()指定特定范围的值,不用再自己\*多少





## 【面试题】Math类中的取整三个函数：

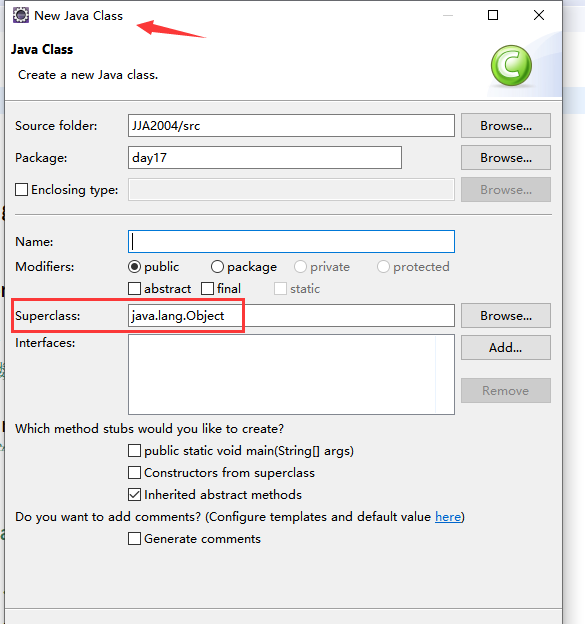
取整函数（ceil（向上取整）、floor（向下取整）、round（四舍五入））

## 【面试题】Java 中的 Math. round(-1. 5) 等于多少？

等于 -1，因为在数轴上取值时，中间值（0.5）向右取整，所以正 0.5 是往上取整，负 0.5 是直接舍弃。

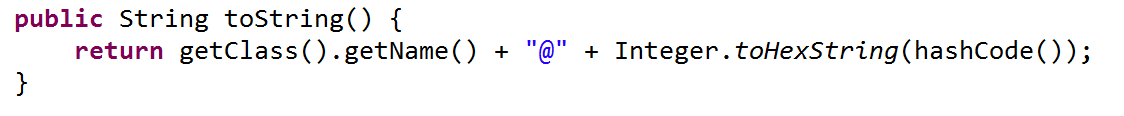
# Object类

类 Object 是类层次结构的根类。每个类都使用 Object 作为超类。所有对象（包括数组）都实现这个类的方法。

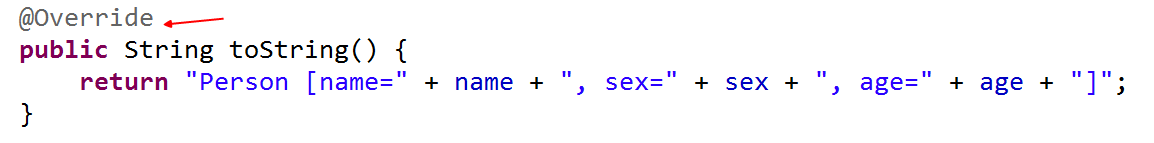


## 重写父类的toString()、equals()方法

toString():返回的是类名+@+哈希地址（唯一特定的地址）。通常，ToString方法会返回一个“以文本方式表示”此对象的字符串。结果应是一个简明但易于读懂的信息表达式。



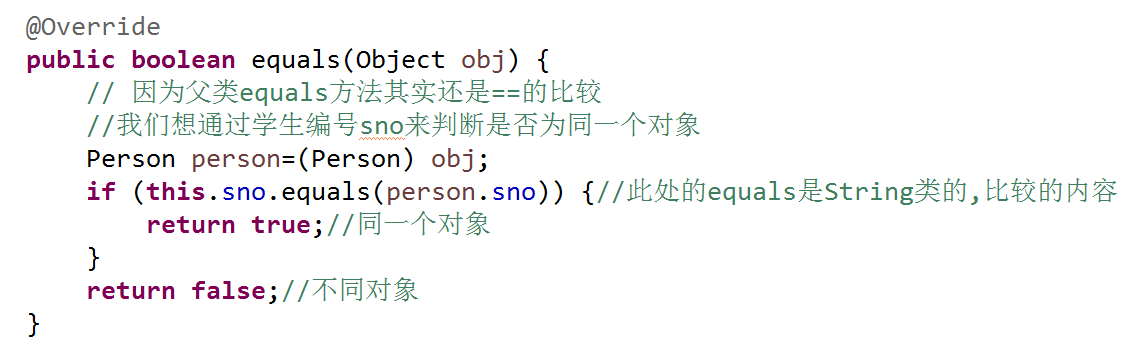
重写:



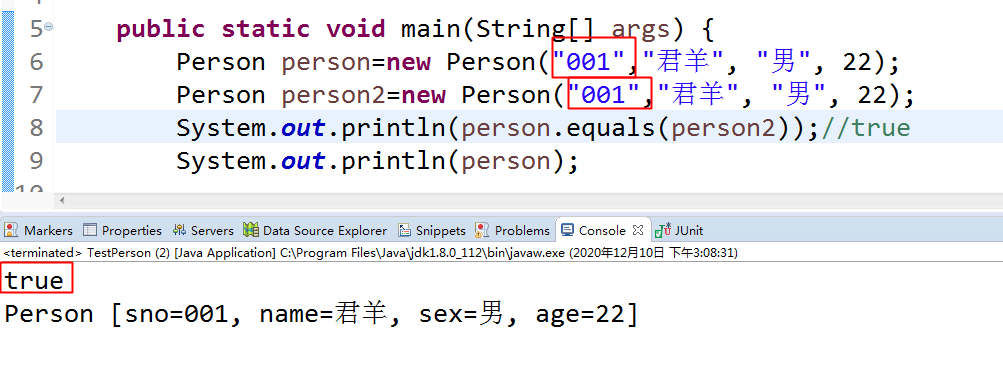
Ctrl+o 列出类的所有方法

equals()用于判断两个对象是否相等

重写父类的equals()



测试类:



## 课堂练习：

定义一个员工类，属性：编号，姓名，学历，岗位

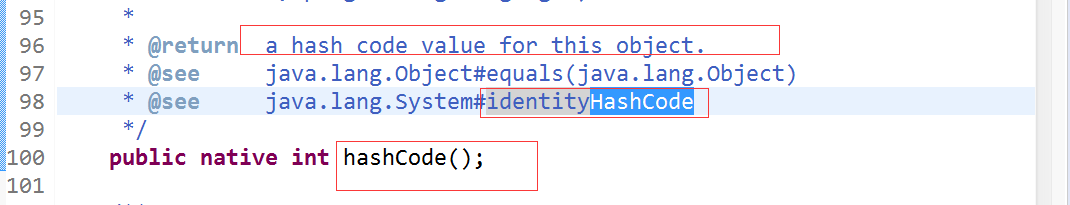
实现以下功能： 1、能够根据编号判断是否是同一个员工

2、重写toString()方法，显示出 姓名：xxx，学历：本科，岗位：高级工程师

主要目的：equals和toString重写

## hashCode

hashCode()被设计是用来使得哈希容器能高效的工作,相当于每个对象的身份证号码。

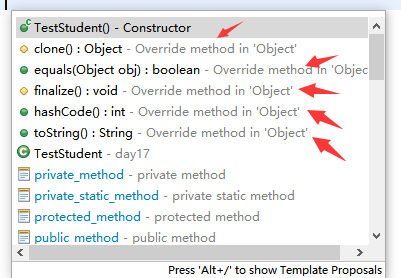


hashCode是java.lang.Object.hashCode() 或者 java.lang.System.identityHashCode(obj) 会返回的值。他是一个对象的身份标识。官方称呼为：**标识哈希码（** identity hash code）

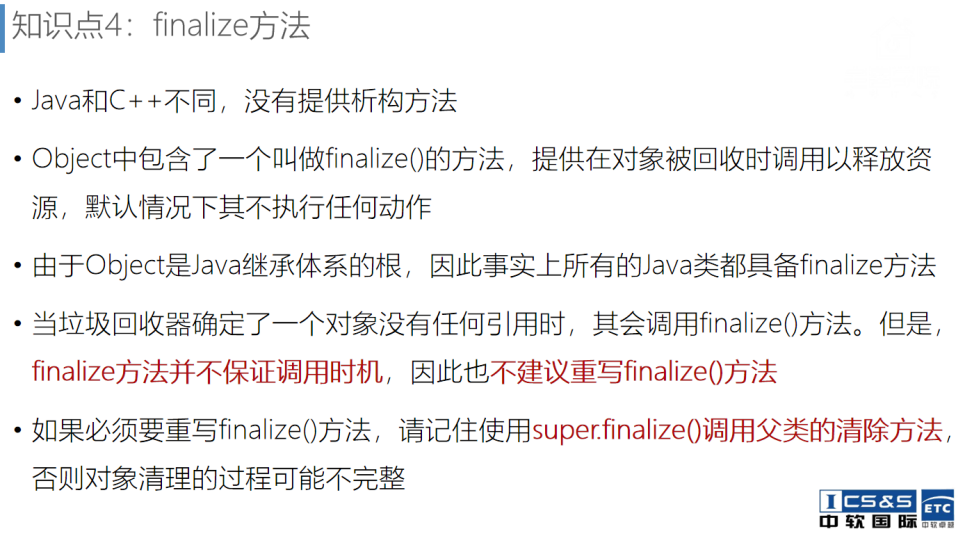
hashCode()这个方法是Java根类Object的方法，因此所有继承自Object类的子类的对象都可以调用该方法计算出本对象的哈希值，默认实现是根据对象的内存地址进行计算。

## 【面试题】子类中可以重写父类的哪些方法？

Alt+/



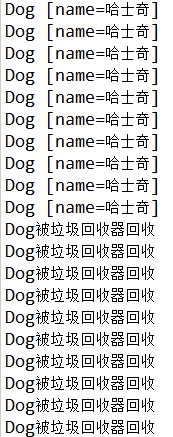
## finalize()











## 【面试题】final finalize() finally有什么不同?

回答方式:就是每个关键字或方法的应用场景.

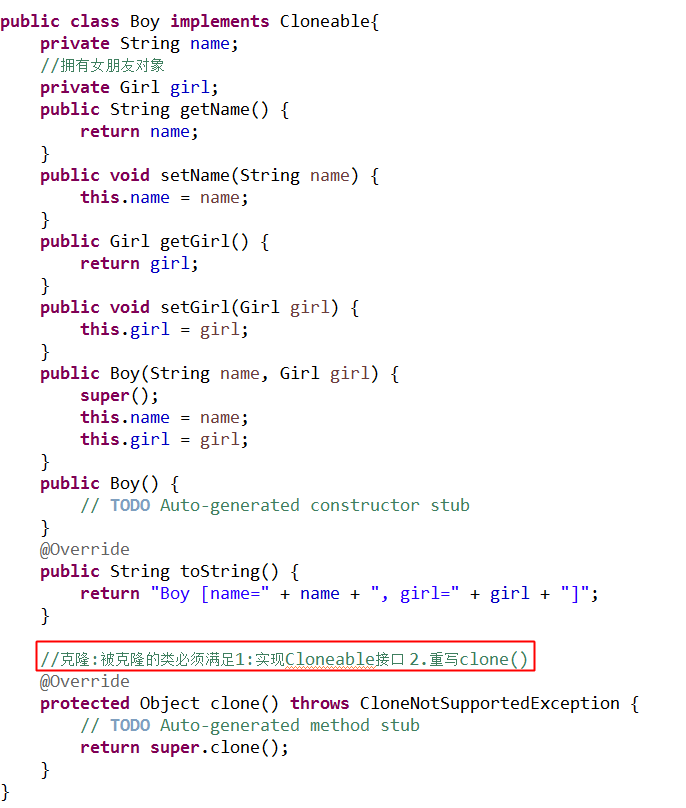
## finalize()小结

1. finalize()每个对象都有.
2. finalize()不需要程序员去调用,由java虚拟机(JVM)自动调用
3. Java对象如果没有更多的引用去指向它,则该java对象成为垃圾数据,等待垃圾回收器的回收,垃圾回收器在回收这个java对象就会自动调用该对象的finalize()

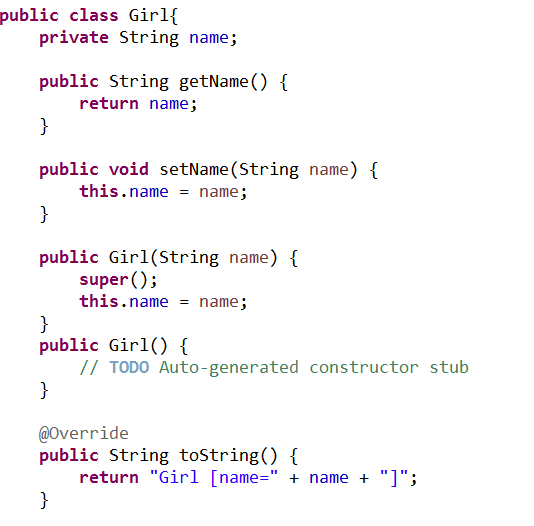
## Object克隆的方法(了解):

使用clone的方法时候,要保证我们的那个对象对应的类,实现了Cloneable的接口才行;

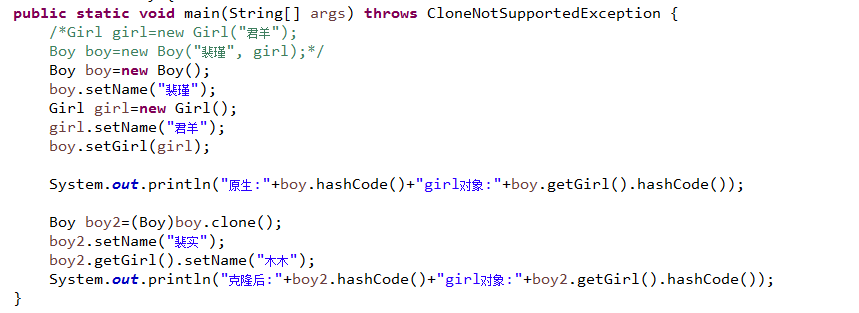
### 浅克隆:

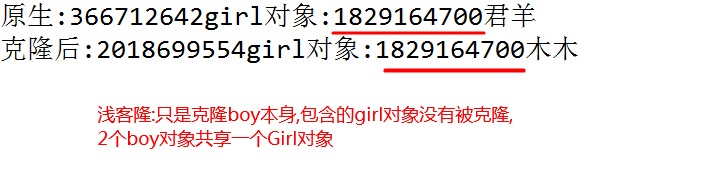


女朋友:



测试类:

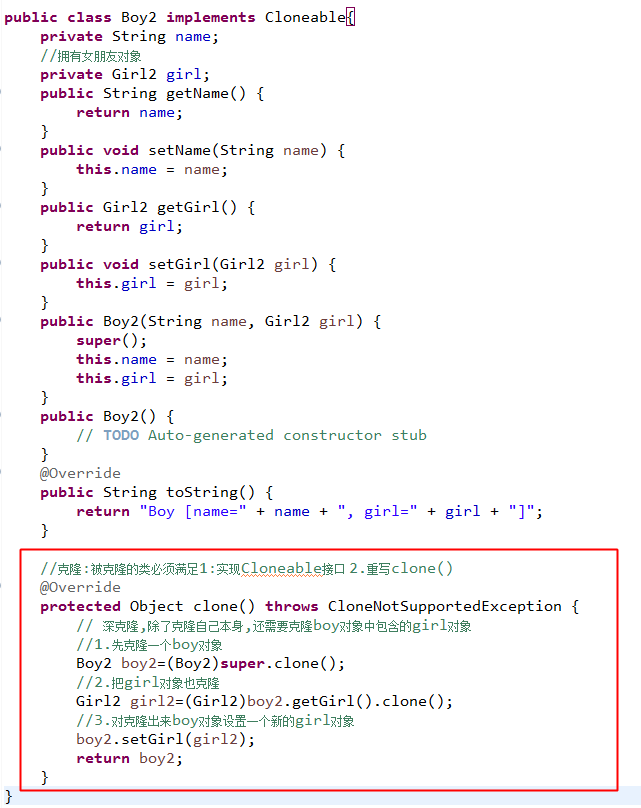


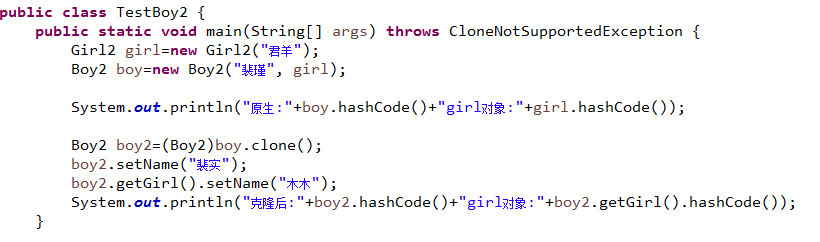


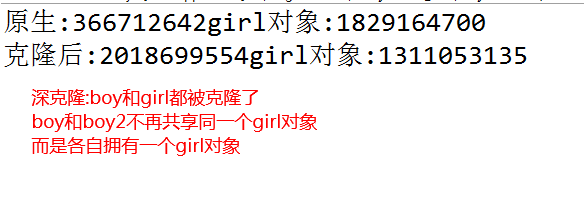
以上称为浅克隆,看到的结果,Boy中引用了一个Girl,当girl的某个属性修改的的时候,每个boy对象中的girl信息都是一致的;

我们的想法一定是希望修改了某个boy属性girl的name的时候,其他的boy对象的girl属性不变;

### 深克隆







## 【面试题】谈谈Java的克隆?

浅克隆：创建一个新对象，新对象的属性和原来对象完全相同，对于 引用的其他对象属性，仍指向原有属性所指向的对象的内存地址。(例如：Boy类中有一个Girl数据类型的对象属性，浅克隆的时候不会克隆该Girl对象，而是所有的Boy类的对象共享同一个Girl实例。)

深克隆：创建一个新对象，属性中引用的其他对象也会被克隆，不再指向原有对象地址。  
(例如：Boy类中有一个Girl数据类型的对象属性，深克隆表示同时也会克隆该Girl对象，每一个Boy类的对象都各自拥有一个Girl实例。)  
总之深浅克隆都会在堆中新分配一块区域，区别在于 对象属性引用的对象是否需要进行克隆（递归性的）。  
用处：当你的对象要求为不可变的时候,在传对象的时候,可以考虑clone。例如方法的参数传递，传递对象的时候是引用传递，如果不希望改变原始对象的数据，则就可以使用克隆。

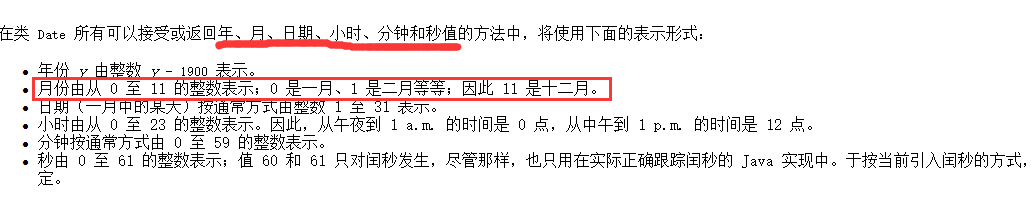
# JDK API 工具包

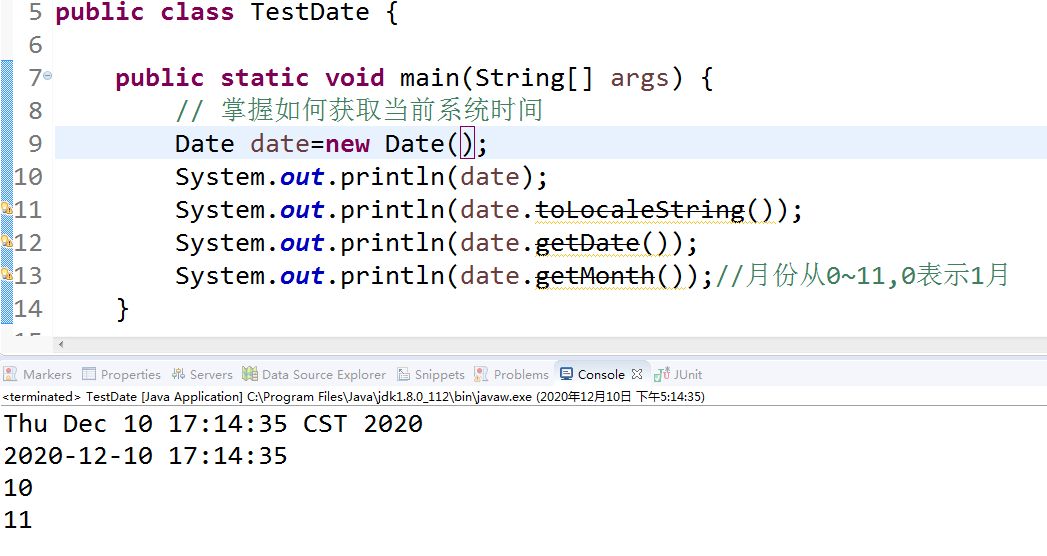
## Java.util.Date:

操作日期时间类

操作日期和时间.只需要掌握获取当前系统日期时间方法



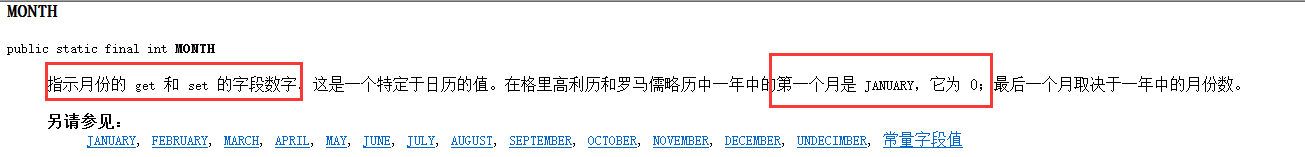




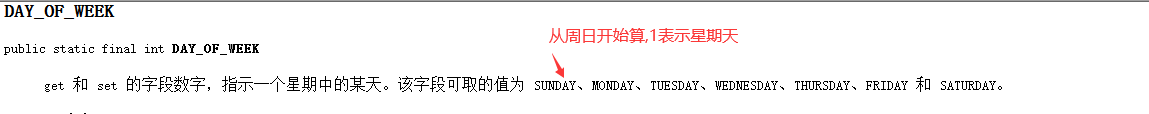
## Java.util.Calendar:

推荐使用的日期时间类

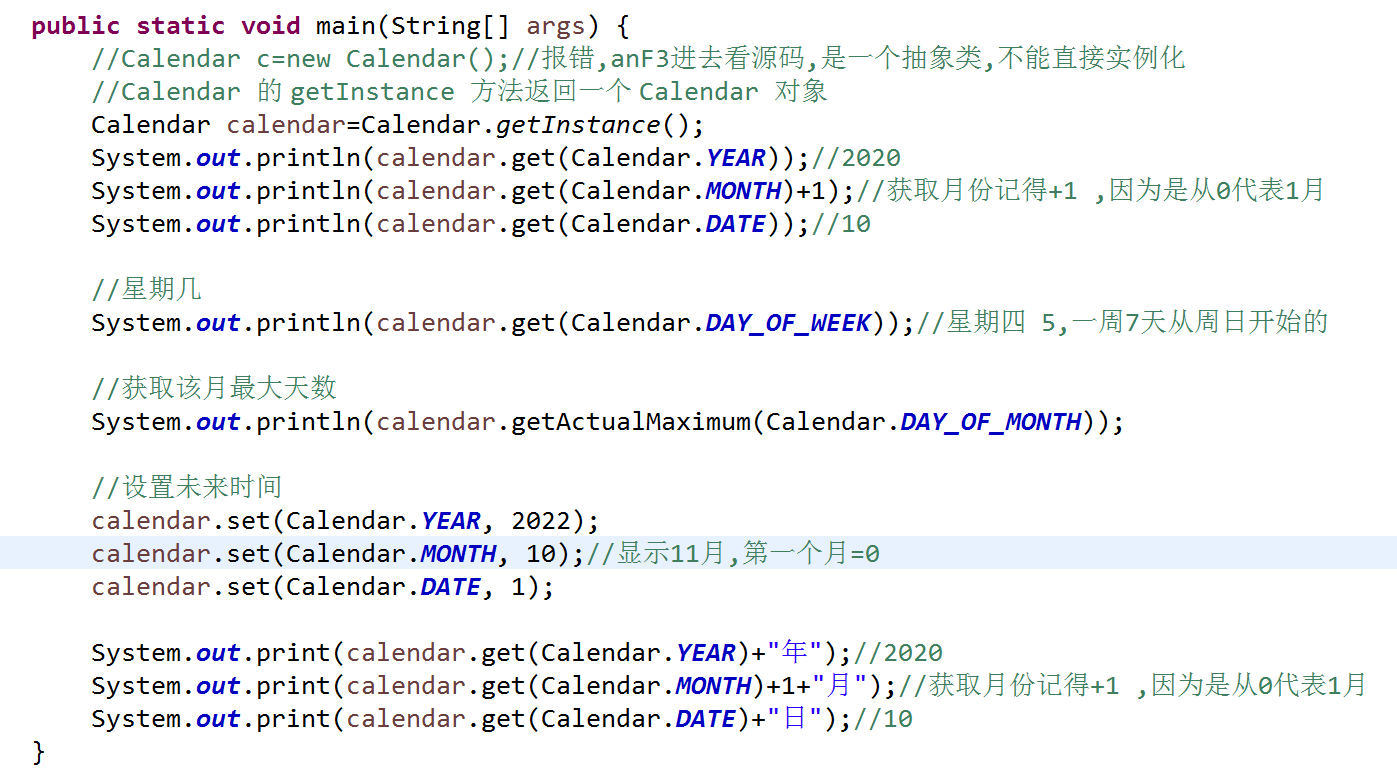








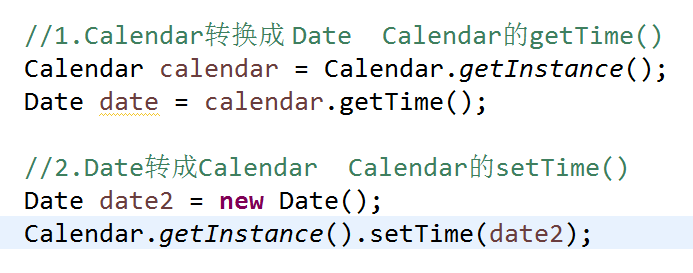




## 问题:Date和Calendar是否可以转换?

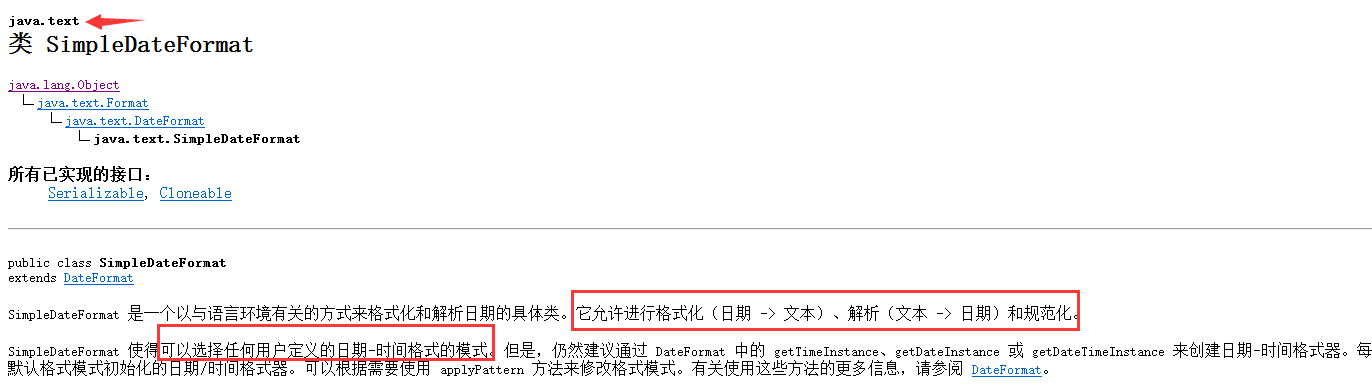






## SimpleDateFormat(辅助):

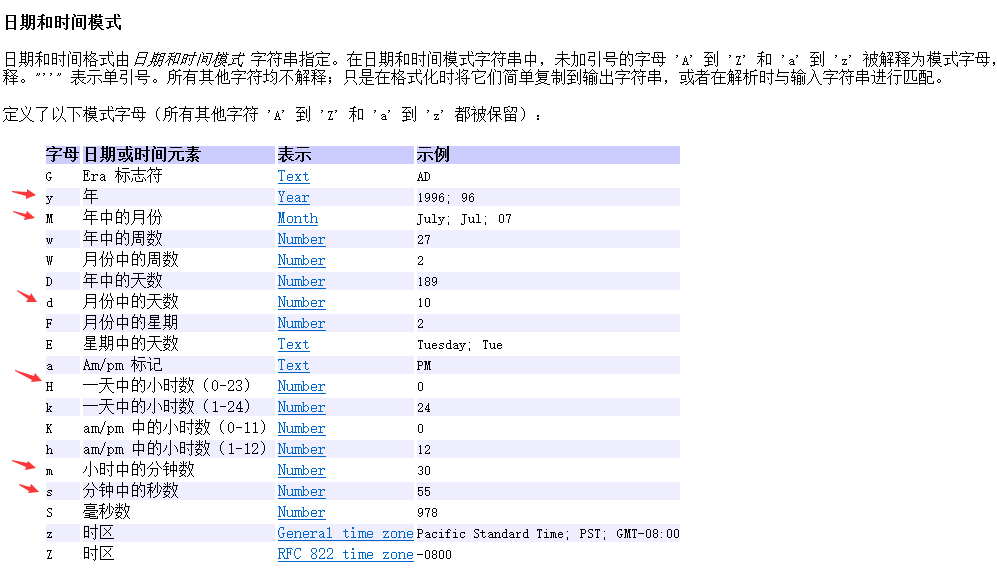
对日期时间进行格式化输出:



yyyy年 MM月dd日 HH:mm:ss

**上面红色的是格式,不能乱写,根据下面格式表写的,蓝色的字体是用户随意选择要显示的格式**

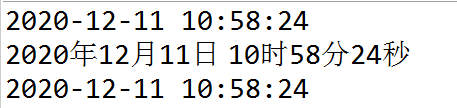
**格式:**











# 总结:

1. 方法的参数传递:String:值传递
2. 可变参数:不确定参数个数用可变参数作为参数, 语法:数据类型... 参数名

特点:1.方法定义时只能写一个可变参数, 2.放最后

1. 可变参数作为参数传递:主要是由参数类型决定是值传递还是引用传递
2. 方法中,引用名出现=指的是 指向发生变化,与原本地址上的内容无关,不会修改内容
3. Math: Math.属性/方法() :3个取整的方法 random()
4. Random类:生成随机数:nextInt(n)范围:0~n,n取不到
5. Object:所有类的超类,toString equals 重写的方法
6. 克隆:浅克隆:2个前提:1.实现克隆接口2.重写clone方法

浅克隆:只能克隆对象本身,

深克隆:克隆对象本身+克隆包含的对象

# 作业

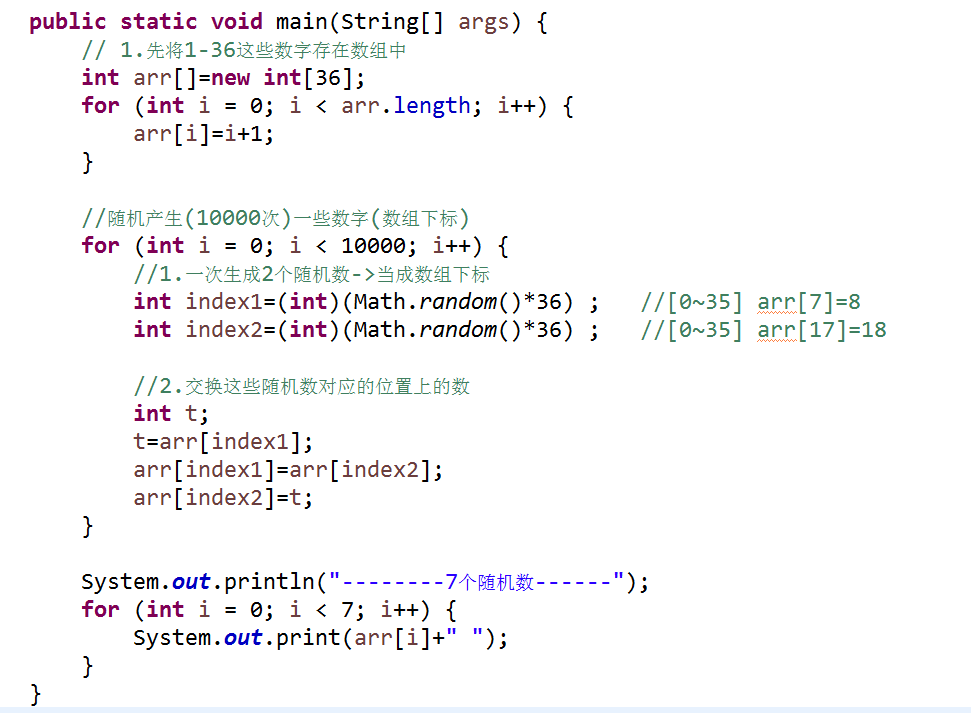
1. 随机数：猜数字游戏 【1-10】Math.random() Random类



1. 彩票生成器:先将1-36这些数字存在数组中,随机产生(10000次)一些数字(数组下标),交换数组元素的位置

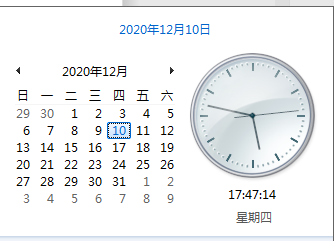
//从而达到把数组中的元素的顺序打乱

//最后从数组中取前7个数



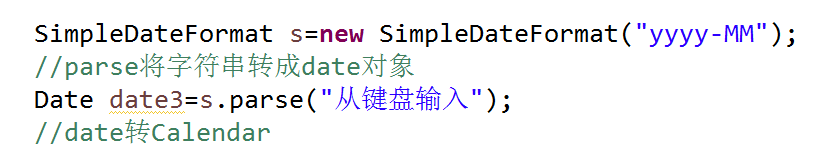
3.综合作业

从控制台输入一个时间值:格式自定义,建议格式:2020-12,然后根据用户的输入,我们打印这个月的日历出来



要解决的问题:

1. 从控制台输入进来的字符串,如何转换成Date(Calandar)格式.可以SimpleDateFormat 把Date转Calandar

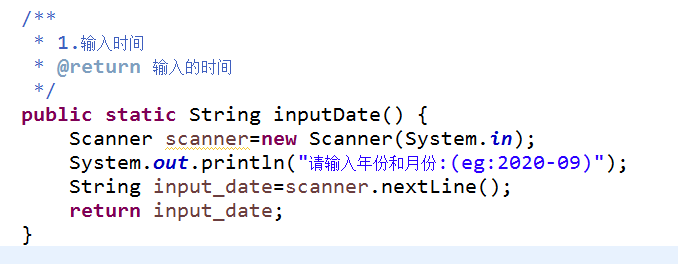


1. 如何确定这个月有几天,并打印出来
2. 确定这个月的第一天是周几

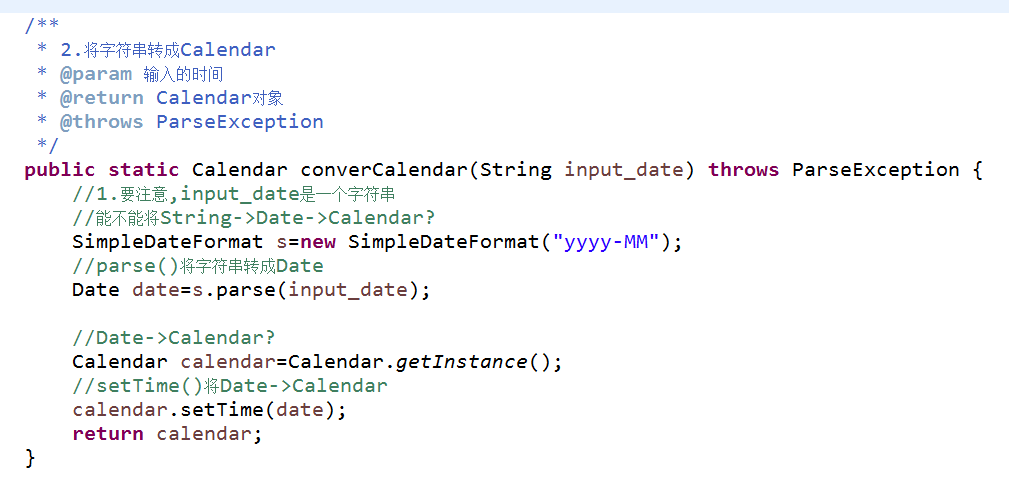


改进:用方法

1. 输入时间



1. 将字符串转成Calendar



1. 打印功能



1. 在main中调用

