# System

System 类包含一些有用的类字段和方法。它不能被实例化。

在 System 类提供的设施中，有标准输入、标准输出和错误输出流；对外部定义的属性和环境变量的访问；加载文件和库的方法；还有快速复制数组的一部分的实用方法。

|  |  |
| --- | --- |
| **字段摘要** | |
| static [PrintStream](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/io/PrintStream.html) | [**err**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#err)  “标准”错误输出流。 |
| static [InputStream](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/io/InputStream.html) | [**in**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#in)  “标准”输入流。 |
| static [PrintStream](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/io/PrintStream.html) | [**out**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#out)  “标准”输出流。 |

|  |  |
| --- | --- |
| **方法摘要** | |
| static void | [**arraycopy**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#arraycopy(java.lang.Object, int, java.lang.Object, int, int))([Object](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Object.html) src, int srcPos, [Object](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Object.html) dest, int destPos, int length)  从指定源数组中复制一个数组，复制从指定的位置开始，到目标数组的指定位置结束。 |
| static long | [**currentTimeMillis**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#currentTimeMillis())**()  返回以毫秒为单位的当前时间。** |
| static void | [**exit**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#exit(int))**(int status)  终止当前正在运行的 Java 虚拟机。** |
| static void | [**gc**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#gc())**()**  运行垃圾回收器。 |
| static [String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) | [**getenv**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#getenv(java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **name)  获取指定的环境变量值。** |
| static [Properties](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Properties.html) | [**getProperties**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#getProperties())**()  确定当前的系统属性。** |
| static [String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) | [**getProperty**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#getProperty(java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **key)  获取指定键指示的系统属性。** |
| static [String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) | [**getProperty**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#getProperty(java.lang.String, java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **key,** [**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **def)  获取用指定键描述的系统属性。** |
| static int | [**identityHashCode**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#identityHashCode(java.lang.Object))**(**[**Object**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Object.html) **x)  返回给定对象的哈希码，该代码与默认的方法 hashCode() 返回的代码一样，无论给定对象的类是否重写 hashCode()。** |
| static void | [**load**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#load(java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **filename)  从作为动态库的本地文件系统中以指定的文件名加载代码文件。** |
| static void | [**loadLibrary**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#loadLibrary(java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **libname)  加载由 libname 参数指定的系统库。** |
| static [String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) | [**mapLibraryName**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#mapLibraryName(java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **libname)  将一个库名称映射到特定于平台的、表示本机库的字符串中。** |
| static long | [**nanoTime**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#nanoTime())**()  返回最准确的可用系统计时器的当前值，以毫微秒为单位。** |
| static void | [**setErr**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#setErr(java.io.PrintStream))**(**[**PrintStream**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/io/PrintStream.html) **err)  重新分配“标准”错误输出流。** |
| static void | [**setIn**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#setIn(java.io.InputStream))**(**[**InputStream**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/io/InputStream.html) **in)  重新分配“标准”输入流。** |
| static void | [**setOut**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#setOut(java.io.PrintStream))**(**[**PrintStream**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/io/PrintStream.html) **out)  重新分配“标准”输出流。** |
| static void | [**setProperties**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#setProperties(java.util.Properties))**(**[**Properties**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Properties.html) **props)  将系统属性设置为 Properties 参数。** |
| static [String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) | [**setProperty**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/System.html#setProperty(java.lang.String, java.lang.String))**(**[**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **key,** [**String**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) **value)** |

获取系统属性信息：Properties getProperties();

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Properties prop = System.*getProperties*();

// 因为Properties是Hashtable的子类，也就是Map集合的一个子类对象。

// 那么可以通过map的方法取出该集合中的元素。

// 该集合中存储都是字符串。没有泛型定义。

// 如何在系统中自定义一些特有信息呢？

System.*setProperty*("mykey", "myvalue");

// 获取指定属性信息。

String value = System.*getProperty*("os.name");

System.*out*.println("value=" + value);

// 可不可以在jvm启动时，动态加载一些属性信息呢？

String v = System.*getProperty*("haha");

System.*out*.println("v=" + v);

// 获取所有属性信息。

**for** (Object obj : prop.keySet()) {

value = (String) prop.get(obj);

System.*out*.println(obj + "=" + value);

}

}

}

能获取到如下一些信息：

java.runtime.name=Java(TM) SE Runtime Environment

sun.boot.library.path=D:\software\jdk\jdk6\jre\bin

java.vm.version=20.10-b01

java.vm.name=Java HotSpot(TM) 64-Bit Server VM

user.country=CN

user.dir=G:\develop\workroom\eclipse\_most\_new\JavaDemo

java.runtime.version=1.6.0\_35-b10

os.name=Windows 7

sun.jnu.encoding=GBK

user.home=C:\Users\Administrator

file.encoding=UTF-8

java.class.path=G:\develop\workroom\eclipse\_most\_new\JavaDemo\bin

user.name=Administrator

java.home=D:\software\jdk\jdk6\jre

user.language=zh

java.version=1.6.0\_35

file.separator=\

注：

// 可不可以在jvm启动时，动态加载一些属性信息呢？

String v = System.*getProperty*("haha");

System.*out*.println("v=" + v);

这里的v=null，但是可以在运行的时候加上这个参数，如：



这样就可以看到输出v=qqqqq，

这个命令选项，看java命令的帮助即可知道，在命令行输入java查看帮助，如下：



# Runtime

每个 Java 应用程序都有一个 Runtime 类实例，使应用程序能够与其运行的环境相连接。可以通过 getRuntime 方法获取当前运行时。

应用程序不能创建自己的 Runtime 类实例。

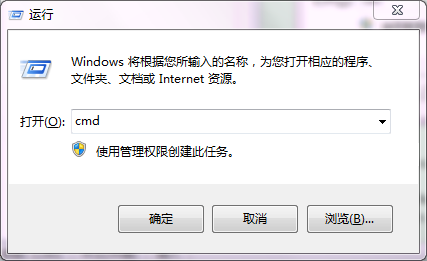
这个类没有构造函数，并且很多方法都不是静态的，像这样的类一般会提供一个静态方法返回它自己的实例。

|  |  |
| --- | --- |
| [Process](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Process.html) | [**exec**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#exec(java.lang.String))([String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) command)  在单独的进程中执行指定的字符串命令。 |
| void | [exit](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#exit(int))(int status)  通过启动虚拟机的关闭序列，终止当前正在运行的 Java 虚拟机。 |
| long | [freeMemory](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#freeMemory())()  返回 Java 虚拟机中的空闲内存量。 |
| void | [gc](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#gc())()  运行垃圾回收器。 |
| static [Runtime](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html) | [getRuntime](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#getRuntime())()  返回与当前 Java 应用程序相关的运行时对象。 |
| void | [loadLibrary](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#loadLibrary(java.lang.String))([String](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/String.html) libname)  加载具有指定库名的动态库。 |
| long | [maxMemory](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#maxMemory())()  返回 Java 虚拟机试图使用的最大内存量。 |
| long | [totalMemory](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#totalMemory())()  返回 Java 虚拟机中的内存总量。 |

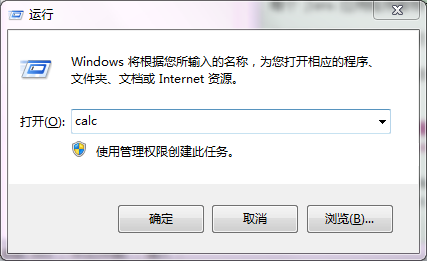
exec是execute的缩写，Windows下的可执行文件扩展名：qq.exe中的exe也是execute的缩写

cmd是command的缩写

打开命令行：



打开计算机器：



在代码中也可以：

public static void main(String[] args) throws IOException {

Runtime r = Runtime.*getRuntime*();

Process p = r.exec("calc"); // 只要是在命令行能执行的命令都可以填到这里执行，如：“C:\\soft\\qq.exe”

// Thread.sleep(4000);

// p.destroy();

}

还有可以这样用：Process p = r.exec("notepad.exe SystemDemo.java");用记事本打开一个文件，可以举一反三，可以指定对应的程序来打开对应的文件。

Process

[ProcessBuilder.start()](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/ProcessBuilder.html#start()) 和 [Runtime.exec](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Runtime.html#exec(java.lang.String[], java.lang.String[], java.io.File)) 方法创建一个本机进程，并返回 Process 子类的一个实例，该实例可用来控制进程并获得相关信息。Process 类提供了执行从进程输入、执行输出到进程、等待进程完成、检查进程的退出状态以及销毁（杀掉）进程的方法。

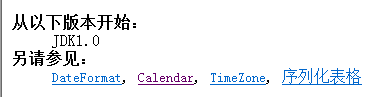
可看到Process这个类是抽象的，就不能创建实例，这个类有构造函数，则这个构造函数肯定是给子类用的。Process中的方法全是抽象的，这时候会去想有没有实现它的子类可以用，发现没有，这是因为进程是程序一运行的时候就创建了，不是由我们去创建的，是由底层去创建的，所以定义成抽象的，由底层去实现，不需要我们去实现，我们也不知道怎么去实现。

# Date

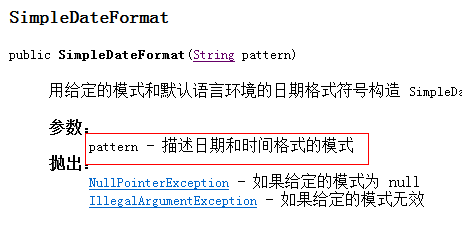
Date date = new Date();

syso(date);

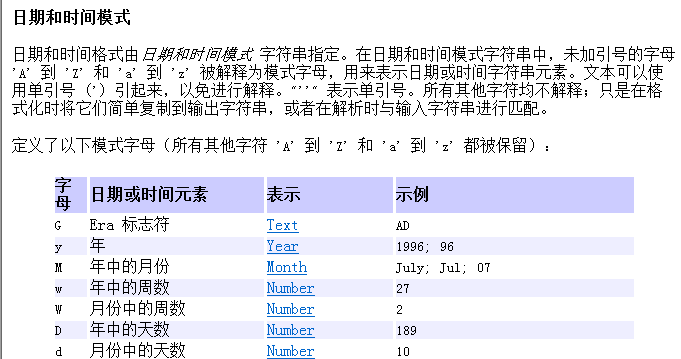
如上语句，输出的日期格式是外国的，我们看不懂，查看Date中的很多方法都已经过时了，那怎样可以把Date的格式转换为我们想要的格式呢？这时候可以想想和这个类相关的一些类，这时可以看这个类中的“另请参见”，如下：



如这Date类相关的类都在另请参见中，我们看到有一个DateFormat，明显就是日期格式化的类，打开这个类发现是一个抽象类，所以找它的子类，发现有一个[SimpleDateFormat](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/text/SimpleDateFormat.html)子类，打开它，查看它的构造函数：



Pattern参数用于描述日期和时间格式的模式，仔细查看这个类，会看到有“日期和时间模式”的相关说明，截图如下：

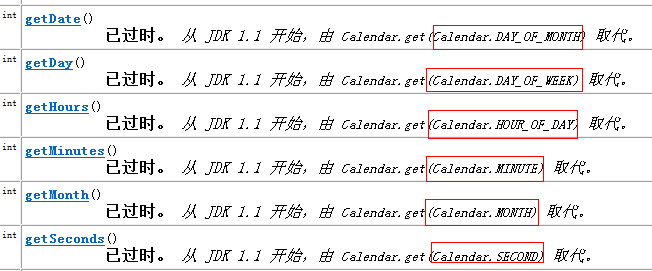


# Calendar

如果想要的不只是显示日期，只想获取单独的年或者月份，这时怎么办？虽然Date中有相应的方法，但是都是过时的，如下：



文档说由Calendar.get(Calendar.YEAR)取代了。其他的很多方法都是由Caleandar 取代了。获取年月日时分妙时，具体要传哪个Calendar的静态字段，其实去看Date对象中对应的方法就有答案了，如下：



发现Caleandar是一个抽象类，而且它有子类[GregorianCalendar](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/GregorianCalendar.html)，直接new 这个子类是没有问题的。同时Caleandar类还提供了静态方法返回子类对象，所以也可以用这些方法来获取Calendar对象，如下：

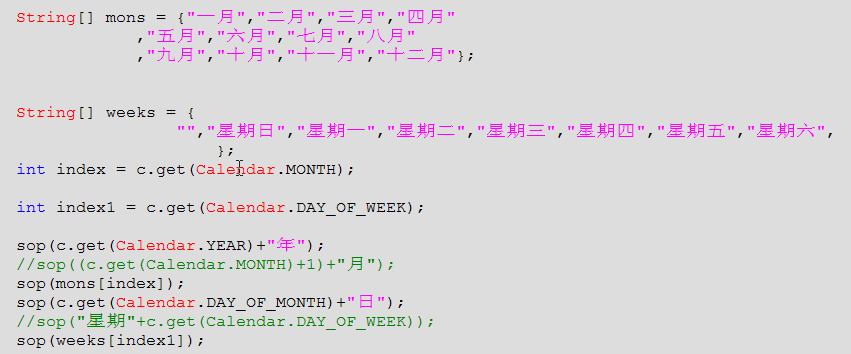
|  |  |
| --- | --- |
| static [Calendar](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html) | [**getInstance**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html#getInstance())()  使用默认时区和语言环境获得一个日历。 |
| static [Calendar](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html) | [**getInstance**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html#getInstance(java.util.Locale))([Locale](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Locale.html) aLocale)  使用默认时区和指定语言环境获得一个日历。 |
| static [Calendar](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html) | [**getInstance**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html#getInstance(java.util.TimeZone))([TimeZone](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/TimeZone.html) zone)  使用指定时区和默认语言环境获得一个日历。 |
| static [Calendar](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html) | [**getInstance**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Calendar.html#getInstance(java.util.TimeZone, java.util.Locale))([TimeZone](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/TimeZone.html) zone, [Locale](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/Locale.html) aLocale)  使用指定时区和语言环境获得一个日历。 |

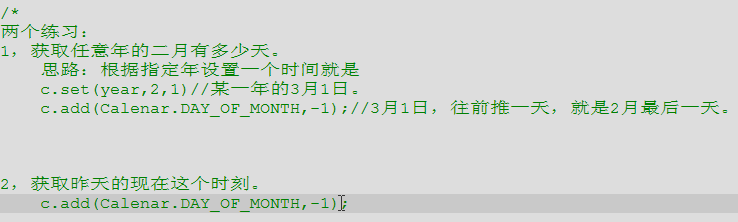
其实这里的getInstance返回的就是[GregorianCalendar](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/util/GregorianCalendar.html)对象。

外国中的星期日是第一天，所以在获取星期几时，如果是星期日返回1，星期一返回2，星期2返回3。。。。

获取月份是1月返回的是0，2月返回的是1。。。。

所以如果要用Calendar来拼接年月日的话可以使用查表法，如下：





一月是0，所以上面设置3月是传的是2。

练习：2015年2月26日开学，3月10日毕业，周六日不上课，请计算总共上了几天课？

Calendar calendar = Calendar.*getInstance*();

calendar.set(2015, 1, 26);

**int** dayCount = 0;

**do** {

**int** week = calendar.get(Calendar.*DAY\_OF\_WEEK*);

**if** (week != 1 && week != 7) {

// 如果不是星期日和星期六

dayCount++;

}

calendar.add(Calendar.*DAY\_OF\_MONTH*, 1);

**if** (calendar.get(Calendar.*MONTH*) == 2 && calendar.get(Calendar.*DAY\_OF\_MONTH*) == 10) {

// 如果是3月10号则退出循环

**break**;

}

} **while**(**true**);

System.*out*.println("上课天数：" + dayCount);

# Math

[**ceil**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Math.html#ceil(double))(double a)返回大于或等于参数的最小整数，如ceil(5.23)

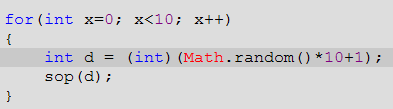
[**floor**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Math.html#floor(double))(double a) 返回小于或等于参数的最大整数，如ceil(5.23)

[**round**](mk:@MSITStore:G:\1JavaWeb资料\jdk6.ZH_cn.chm::/j2se6/api/java/lang/Math.html#round(double))(double a) 四舍五入

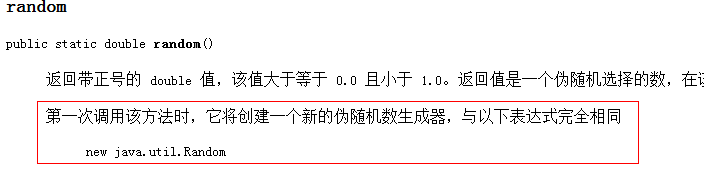
 2的3次方

随机数，随机买彩票时可使用，示例如下：

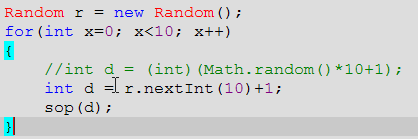
0 ~ 10的随机数：



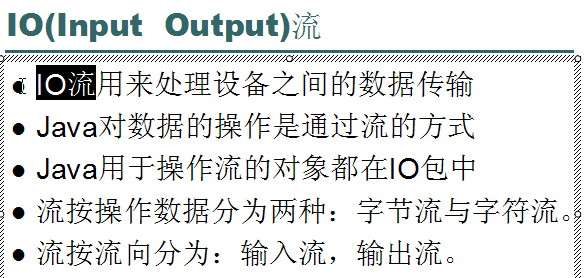
查看Math.random()方法，如下：



用Random类实现上面的模拟0 ~ 10随机数的例子：



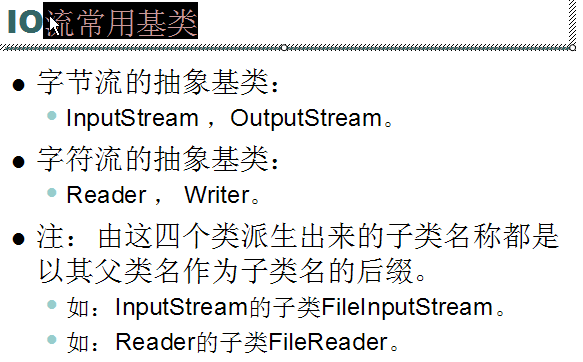
# IO流



为了统一所有的码表，创建了Unicode码表，对这个表进行优化后就形成了UTF-8码表。区别：Unicode无论什么字符都用两个字节表示，UTF-8的话如果1个字节能表示则用1个字节表示，1个不够用就用两2个字节，再不够就用3

个字节。

为什么要有字符流？GBK码表中有中文，UTF-8码表中也有中文，但是相同的中文字符在这两个码表中是不一样的，比如GBK码表中的某些汉字的编码可能与UTF-8中的某些日本字的编码相同，如用GBK码表保存的汉字，保存的肯定是与汉字对应的GBK编码，如果这个保存的文件放到UTF-8码表的机器上读取，则会出现乱码，为了解决这个问题Java就基于字节流创建了一个字符流，字符流里面就融合了各种码表，也就是说你可以指定使用GBK还是使用UTF-8来读取。只要文字才用到编码表，如果是图片、音频、视频这些就字节流。

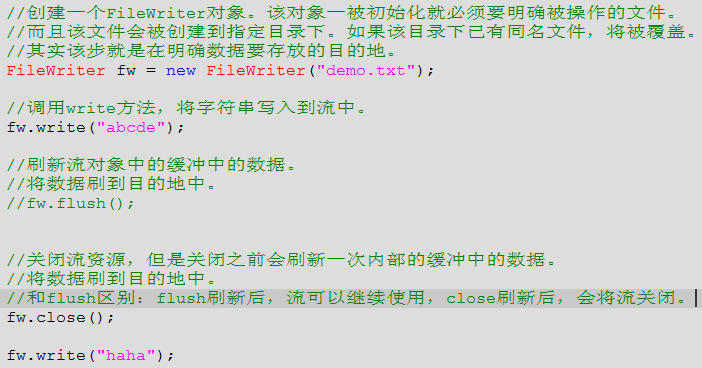


学习IO体系也是像学习集合一样，先看顶层类，学会了顶层的其它的就都容易了。

因为字符操作比较常见，所以这里先学字符流；

# Writer

覆盖的原理：把原来的文件删除，再创建新的文件。



new FileWriter(“Demo.txt”)会调用Windows的创建文件的方法来创建一个Demo.txt文件，调用的是Windows的资源

Java写数据与Windows的写数据的方式是不一样的，所以调用Java的IO读写数据其实是调用Windows对应的IO资源去实现的，因此读写操作完成之后要记得释放掉这些Windows资源，也就是调用close方法。

# IO异常处理方式

class FileWriterDemo2 {

public static void main(String[] args) {

FileWriter fw = null;

Try {

fw = new FileWriter("demo.txt"); // 如果这里初始化失败则fw为null，所以finally中调用时需要判断是否为空

fw.write("abcdefg");

} catch (IOException e){

System.out.println("catch:"+e.toString());

} finally {

try {

if(fw!=null)

fw.close();

} catch (IOException e) {

System.out.println(e.toString());

}

}

}

}

# 在已有文件的基础上续写

前面的例子中demo.txt会覆盖已存在的文件，如果想把内容添加到已存在的文件的末尾，则使用这个构造函数即可：new FileWriter(“demo.txt”, true);

# 写入换行

FileWriter fw = new FileWriter("demo.txt",true);

fw.write("nihao\r\nxiexie");

注：Windows中的换行是用\r\n来表示的，而Linux中是用\n表示的，所以如果只写入\n，则在Windows中无法实现换行，因为无法识别，而且会显示成一个乱码，如下：



如果用一些专业的文本编辑软件则能识别\n是换行，如Notepad++，用这个打开上面的demo.txt可成功识别\n为换行。如果想要程序通用的话，可使用BufferedWriter类的newLine方法，这样可以自动判断是什么系统而写入“\n”还是“\r\n”。

# FileReader

读取的第一种方式：

//创建一个文件读取流对象，和指定名称的文件相关联。

//要保证该文件是已经存在的，如果不存在，会发生异常FileNotFoundException

FileReader fr = new FileReader("demo.txt");

//调用读取流对象的read方法。

//read():一次读一个字符。而且会自动往下读。

int ch = 0;

while((ch=fr.read())!=-1)

{

System.out.println(

}

/\*

while(true)

{

int ch = fr.read();

if(ch==-1)

break;

System.out.println("ch="+(char)ch);

}

\*/

fr.close();

读取的第二种方式：

FileReader fr = new FileReader("demo.txt");

//定义一个字符数组。用于存储读到字符。

//该read(char[])返回的是读到字符个数。

char[] buf = new char[1024]; // 数组长度一般定义为1024的整数倍，这里1024则定义了buf的容量为2K，因为一个字符是两个字节，2 \* 1024 = 2K。

int num = 0;

while((num=fr.read(buf))!=-1)

{

System.out.println(new String(buf,0,num));

}

fr.close();

// 注：把字符读到buf中的时候，每次都是存到这个数组中，当第二次读的时候，如果只读到了1个字母，则存到数组中的第一个位置，但是数组往后位置已然存在数据，那是第一次读的时候保存的，所以我们需要知道读取了几个字符，读了几个就从数组中取几个。

# FileReader实例

//读取一个.java文件，并打印在控制台上。

import java.io.\*;

class FileReaderTest

{

public static void main(String[] args) throws IOException

{

FileReader fr = new FileReader("DateDemo.java");

char[] buf = new char[1024];

int num = 0;

while((num=fr.read(buf))!=-1)

{

System.out.print(new String(buf,0,num));

}

fr.close();

}

}

//将C盘一个文本文件复制到D盘。

/\*

复制的原理：

其实就是将C盘下的文件数据存储到D盘的一个文件中。

步骤：

1，在D盘创建一个文件。用于存储C盘文件中的数据。

2，定义读取流和C盘文件关联。

3，通过不断的读写完成数据存储。

4，关闭资源。

\*/

import java.io.\*;

class CopyText

{

public static void main(String[] args) throws IOException

{

copy\_2();

}

public static void copy\_2()

{

FileWriter fw = null;

FileReader fr = null;

try

{

fw = new FileWriter("SystemDemo\_copy.txt");

fr = new FileReader("SystemDemo.java");

char[] buf = new char[1024];

int len = 0;

while((len=fr.read(buf))!=-1)

{

fw.write(buf,0,len);

}

}

catch (IOException e)

{

throw new RuntimeException("读写失败");// 在这里也可以把创建的文件删除，因为复制失败了

}

finally

{

if(fr!=null)

try

{

fr.close();

}

catch (IOException e)

{

}

if(fw!=null)

try

{

fw.close();

}

catch (IOException e)

{

}

}

}

//从C盘读一个字符，就往D盘写一个字符。这样的方式不好，因为硬盘的读取磁头指针就会找到数据所在的地方读一个数据，再转到目的地去写一个数据。应该一次读多点数据，然后在写的时候也一次写多点数据，这样可以减少磁头在读和写之间的来回切换。

public static void copy\_1()throws IOException

{

//创建目的地。

FileWriter fw = new FileWriter("RuntimeDemo\_copy.txt");

//与已有文件关联。

FileReader fr = new FileReader("RuntimeDemo.java");

int ch = 0;

while((ch=fr.read())!=-1)

{

fw.write(ch);

}

fw.close();

fr.close();

}

}