#常用基础类 Scanner类

> 基于正则表达式的本文扫描器？使用其**构造器**可以接收文件、输入流、字符串作为数据源，解析出数据。 提供了四个方法：
>
> - boolean hasNextXXX()和hasNext():判断是否有下一个输入，带XXX的方法为判断是否为XXX类型，XXX代表基本数据类型。
> - nextXXX()和next():获取下一个输入项，XXX与上面是一个意思。如果输入不同于XXX的类型数据会报InputMisMatchException错误。
> - hasNextLong():判断是否有下一行
> - nextLine():返回输入下一行字符串。
> - useDelimiter(Sting s):填入正则表达式，设置分隔符；**默认是空白（空格、TAB、回车）**

##与运行平台进行交互的类 System类

> 作用：1.提供标准输入、标准输出和错误输出的类变量；2.静态方法访问环境变量、系统属性；3.加载文件和动态链接库（native方法）； 访问环境变量和系统属性：getenv("指定环境变量名");getProperties()(获取所有属性，返回类型为Properties);getProperty("指定属性名"); **获取系统时间**：currentTimeMillis()(毫秒)和nanoTime()(纳秒)，受底层操作系统影响，不一定准确。 System.in/out/err,三个变量分别代表了标准输入（键盘）、标准输出（显示器）和错误输出流； out和err都可以使用System.out.print()/System.err.print()把结果输出到显示器上（因为本身就是数据流），但out输出会有缓存（输出会有延迟，积累一定数量或时间后才输出），**且能重定向？**，但err不会。 提供System.identityHashCode()方法，返回对象根据地址计算的hashCode，当hashcode（）方法被重写后就可以用这个方法判断是否为同一对象。

Runtime类

> 代表运行时环境，每个java程序都有与之相对应的Runtime实例；应用程序只能**通过Runtime.getRuntime()来获取Runtime实例。 Runtime提供gc()和runFinalization()通知系统进行垃圾回收、清理系统资源。 提供方法来加载文件和动态链接库？ 可以获得JVM的相关信息。 可以单独启动一个进程来执行操作系统的指令。**

Object类

> equals()/getClass()/hashCode()/toString():实例方法，分别用于判断是否相等/获取Class对象/返回hashcode值/返回字符串值（默认为“运行时类名@十六进制hashcode值”） protected void finalize()：**注意是protected修饰**，所以只能在类里面使用，让垃圾回收器在没有引用指向该对象时回收资源。protected native Object clone()方法：由于**protected**，只能在子类调用或重写。克隆出一个本类的副本。

clone方法的详细用法：

> 自定义一个类，**实现Cloneable接口**，该接口没有方法，仅仅只是标记说明该类可以进行自我克隆; 自定义类实现一个克隆的方法，**该方法里通过super.clone()（需要进行强制类型转换，因为返回值为Object）**得到该对象的副本。

```
//继承接口
public class TestSc implements Cloneable{
    //自定义一个方法
```

```
    public TestSc clone() throws CloneNotSupportedException {
        //强制类型转换
        return (TestSc)super.clone();
    }
}
```

注意点

> clone方法克隆出一个新的对象，对象里面的成员变量都是**简单复制其值，所以其副本的引用类型变量则会和原对象里的变量指向同一对象**

Objects工具类

> 提供一些**空指针安全的方法**操作对象。如：直接使用object的toString()方法，容易出现NullPointerException（空指针异常）；但使用Objects.toString(obj)时，obj为空返回null;

String/StringBuffer/StringBuilder类

> 都实现了CharSequence接口。 String是一个不可变类； StringBuffer/StringBuilder都是序列可变的字符串，**前者是线程安全的，后者是线程不安全的**。生成最终想要的序列后，使用toString()方法输出。StringBuffer/StringBuilder有**append()（追加）,insert()（插入）,reverse()（反转）,setCharAt(),setLength()，replace(int,int,String)（替换）,delete(int,int)（删除）**等方法改变字符串序列。

String类

> 构造器：String(),参数可使用char[]数组，StringBuffer/StringBuilder。 int CompareTo(String):对比两字符串，**相同返回0**，不相同返回第一个不相等的字符差或者长度差。 boolean contentEquals(StringBuffer):与StringBuffer对象进行比较。 String concat(String):将两个String对象连接在一起。 int indexOf():找出某字符在字符串中第一次出现的位置。 **int length()**:**使用方法不是属性获取长度**。 **String[] split(分隔字符)**：获取按分隔字符分隔的字符串数组。 **substring(int beginIndex, int endIndex)**,获取字符串中部分字符串； String字符串变成小写使用toLowerCase()方法；**字符Char**变成小写使用Character.toLowerCase（char）静态方法；大写是toUpperCase()

StringBuffer/StringBuilder类

> 使用**length**属性获取长度。

Math类

> **所有方法都是类方法**。 取整运算： 最大整数：Math.floor()；最小整数：Math.ceil();四舍五入取整：Math.round() 大小相关运算： 最大值：Math.max();最小值:Math.min();随机数：Math.random();

Random随机数类

> 两个构造器，一个使用默认的种子（当前系统时间），另一个需要显式传入long型整数的种子。 比Math.random()提供更多的方法来生成各种伪随机数，生成浮点类型，整数类型，指定生成随机数的范围。 使用nextXXX()**实例方法**来获取各种类型的随机数； **只要两个Random对象的种子相同，方法调用顺序相同，则输出结果相同，所以是伪随机 通常使用当前时间作为种子：Random random = new Random(System.currentTimeMillis());** ThreadLocalRandom类可以在并发访问的环境下，减少多

> 线程资源竞争。使用**ThreadLocalRandom.current()**获取该类对象，然后调用各种**nextXXX()**方法来获取伪随机数

BigDecimal类

> 为了精确表示与计算浮点数，使用BigDecimal类;
>
> 1. 一般使用BigDecimal(String)的**字符串做参数的**构造器,因为使用double类型参数构造器时，传入的值不是精确地（double本身不精确）；
> 2. 如果需要使用double类型数据来创建对象，则通过**BigDecimal.valueOf(double value)**来创建对象。
> 3. 拥有：add()加法、subtract()减法、multiply()乘法、divide()除法来进行浮点数运算。

时间类：Date、Calendar类

正则表达式：