**StringBuffer类与String类的区别：**

除了可以是String类表示字符串，也可以使用StringBuffer类表示字符串，但两者的 区别如下：

1. String类对象的内容一旦声明便不可改变，只能通过修改引用地址进行改变，而 StringBuffer类对象声明后可以直接改变其内容，不需要修改引用地址。
2. String类对象使用“+”进行字符串的连接，而StringBuffer类只能使用append( )方 法进行字符串的连接。
3. StringBuffer类对象引用传递后可以修改内容，但String类对象不能。

**StringBuffer类的常用操作（方法）：**

为了保证用户操作的适应性，在StringBuffer类中定义的大部分方法名称都与String是 一样的。

1. 使用append( )方法可以进行字符串的连接，此方法返回一个StringBuffer类的实例， 这样就采用代码链子的形式一直调用append( )方法；其中，append方法被重载多次， 可以连接任意的数据类型。
2. 可以使用insert( )方法，在指定的位置（下标处）插入内容。
3. 可以使用reverse( )方法进行反转操作。
4. 可以使用replace( )方法对指定范围的内容进行替换，而在String类中是使用

replaceAll( )方法。

1. 可以使用substring( )方法截取指定范围内的字符串内容或者截取从某一位置开始到 结尾的内容。
2. 可以使用delete( )方法删除指定范围的内容。
3. 可以使用indexOf( )方法查找指定的字符串是否存在，如果存在则返回其下标开始位 置，如果不存在则返回-1。
4. 使用toString( )方法可以把StringBuffer类对象变为字符串对象。

**StringBuffer类的应用场景：**

如果一个字符串要频繁修改的话，使用String类会产生很多的垃圾空间，性能很低。所 以，对于频繁修改的字符串，最好使用StringBuffer类，以提升性能。

**StringBuffer类与StringBuilder的区别：**

StringBuffer类中的提供的方法全部使用了synchronized声明，都是同步的方法，属于安 全的线程操作，而StringBuilder类中提供的方法都属于异步方法，属于不安全的线程操 作。

**Runtime类基本概念：**

Runtime类表示运行时操作类，是一个封装了JVM进程的类。

**Runtime类对象的实例化：**

Runtime类本身的构造方法是私有化的（单例设计模式），如果要取得Runtime类的实 例，只能通过类名称调用getRuntime( )方法。

**Runtime类的应用：**

1）使用Runtime类可以取得JVM内存信息，了解JVM内存的使用情况，还可以运行 垃圾回收器，释放内存。

2）使用Runtime类运行本机的可执行程序，与Process类搭配的话，还可以控制可执行 程序的运行时间。

**什么是国际化程序：**

国际化程序是指通过国际化操作让一个程序适应不同国家的语言要求。

**国际化程序的实现思路：**

在Java中要想实现国际化程序，则必须依靠一下条件完成。

1. java.util.Locale类
2. java.util.ResourceBundle类
3. 资源文件
4. java.text.MessageFormat类

·Locale类用来根据ISO编码构造语言环境，对于各个国家都有自己的ISO编码，可以在IE上查看。

·ResourceBundle类的主要作用是读取资源文件，读取资源文件时，可以直接指定资源文件的名称不需要指定文件扩展名；也可以根据Locale对象所指定的区域码来选取指定的资源文件。

·资源文件又称属性文件，是指后缀名为properties的文件。可以根据不同国家配置不同的资源文件，在资源文件中，内容的保存形式为key=value。

·MessageFormat类是Format类的子类，用于格式化资源文件中存在的占位符。

**Java国际化程序的简单实现：**

**处理动态文本：**

在资源文件中，所有的value内容都是固定的，但如果想包含一些动态文本，则必须使 用占位符在资源文件中value中清楚地表示出动态文本的位置，占位符使用“编号”的 格式出现，如“{0}”，如果需要更多的占位符，则直接在value信息的指定位置再加入 占位符“{1}”，“{2}”即可。使用占位符之后，程序可以直接通过MessageFormat类名 调用format( )方法 进行占位符的格式化，为占位符设置内容。

**使用类代替资源文件：**

一般使用资源文件根据key=value的格式存储资源信息。但在Java中，也可以使用类来 保存所有的资源信息，但此种做法并不多见，主要还是以资源文件为主。

如果使用类保存资源信息，需要满足一下条件：

1. 使用Object类型二维数组的形式保存资源信息；
2. 需要继承java.util.ListResourceBundle类，并覆写其中的getContents( )方法，返回 Object类型的二维数组。

**资源文件的优先级：**

如果存在多个资源文件，如Message.properties、Message\_zh\_CN.properties、 Message\_zh\_CN.class。此时，只会使用一个。即

类资源文件优先、然后是带有区域码的资源文件优先。

**什么是System类：**

System类表示系统类，是一些与系统相关的属性和方法的集合。而且在System类中， 所有的属性和方法都是静态的，可以通过System类名称直接调用。

**System类常用操作（方法）：**

System类的常用操作有：系统退出、数组复制、取得本机全部属性或指定属性、运行 垃圾回收器、计算程序的执行时间。

1. 可以使用System类中getProperty( )方法取得本机的全部属性或者根据key取得指定 属性。
2. 可以利用System类中的currentTimeMillis( )方法取得当前毫秒时间，利用结束时间 减去开始时间，以此来计算一个程序的执行了多长时间。

**垃圾对象的回收：**

当一个对象不再指向堆内存空间时，此对象就是一个垃圾对象。但如果一个对象要在回 收前进行某些操作，此时在该对象所属类中覆写Object类中的finalize( )方法（需要 抛出Throwable），然后在此方法中写入需要执行的操作即可。

**对象的生命周期：**

声明对象，实例化对象。当对象不再使用时会被当做垃圾收集，之后对象被终结、最终 被程序卸载。

**日期操作类：**

在Java中有四种方法可以取得日期：

1. java.util.Date类
2. java.util.Calendar类与java.util.GregorianCalendar类
3. java.text.DateFormat类
4. java.text.SimpleDateFormat类

**Date类：**

Date类是取得日期操作最方便的类，直接输出Date类实例即可。但是通过Date类取得 的日期并不符合日常的日期格式，而且也不能精确到毫秒。

**Calendar类：**

Calendar类表示日历类，通过Calendar类取得时间可以精确到毫秒，并且通过Calendar 类可以自定义取得日期的元素。但Calendar类是一个抽象类，需要通过子类 java.util.Gregorian类进行实例化。

**DateFormat类：**

DateFormat类表示日期格式化操作类，由于Date类取得日期无法符合习惯，所以此时 可以通过DateFormat类对Date类对象进行格式化。但此类是一个抽象类，需要通过类 名称调用类中的getDateInstance( )或getDateTimeInstance( )方法实例化，然后调用 DateFormat类中的format( )方法对一个Date类对象进行格式化。

**SimpleDateFormat类：**

1. SimpleDateFormat类是DateFormat类的一个子类，使用此类可以根据自己定义的模 板对日期进行格式化。所以首先需要定义一个格式化的模板，然后根据模板格式化 日期类对象，得到想要的格式日期。
2. 在开发中，用户所输入的各个数据都是以String方式进行接收的，所以在输入日期 的时候就可以使用SimpleDateFormat类将包含日期的字符串转换为日期。

**Math类：**

Math类是数学操作类，提供了一系列的数学操作方法，包括求最大值、最小值、绝对 值等。在Math类中的属性和方法都是静态的，可以直接由类名称调用。

当使用Math类round( )四舍五入时，如果四舍五入的数是负数，小数部分大于0.5才进 位，小于或等于则不进位。

**Random类：**

Random类表示随机数类，使用此类可以产出随机数。

**NumberFormat类：**

1）当遇到的数字较大时，可以使用NumberFormat类实现数字的格式化显示，这样就 可以方便使用者阅读数字。

2）NumberFormat类表示数字格式化类，可以根据本机环境或者指定的Locale对数字 进行格式化显示，以方便阅读。

3）NumberFormat类是一个抽象类，需要调用内部的getInstance( )方法进行实例化。

**DecimalFormat类：**

DecimalFormat类是NumberFormat类的子类，NumberFormat类是根据本机环境或指定 的Locale对数字进行格式化显示，而DecimalFormat类可以根据自己定义的模板格式化 数字。

（利用此类也可以巧妙的精确小数的位数）

**BigInteger类：**

BigInteger类表示大整数类，定义在java.math包中，如果在操作时一个整型数据超过了 整数的最大类型long，数据无法装入时，此时可以使用BigInteger类进行操作。

**BigDecimal类：**  
 如果需要精确小数的位数的话，可以使用BigDecimal类，并且BigDecimal类也可以进 行大数的操作。此类定义在java.math类。

**对象克隆技术：**

一个对象要想被克隆，则该对象所属的类必须满足两个条件：

1）实现Cloneable接口。此接口没有任何的方法定义，是一个标识接口，表示此类对 象可以被克隆。

2）覆写Object类中clone( )方法（还需要抛出异常）。此方法在Object类中的访问权限 是protected，覆写应扩大访问权限以方便外部调用，并且在该方法中调用Object 类中Clone( )方法。

克隆对象时，只需要使用原始对象调用clone( )方法，同时向下转型，因为clone( )方法 返回的是Object类型。

**Arrays类：**

Array类是数组的操作类，定义在java.util包中，主要功能是实现数组排序、数组元 素的查找，数组内容的填充，判断两个数组是否相等，将数组变为字符串的形式。

**比较器的引入：**

在java.util.Arrays类的sort( )方法是一个可以对数组进行排序的方法，此方法被重载多 次，可以对任意类型的数组排序，其中也包含着对对象数组排序的操作支持，但所排序 的数组所属的类必须实现Comparable接口。也就是比较器。

**Comparable接口：**

Comparable接口表示比较器，只要是实现了此接口的类都可以使用Arrays类中sort( ) 方法进行排序。Comparable接口采用了泛型技术，在Comparable接口中只存在一个 compareTo( )方法，实现Comparable接口需要覆写此方法，在此方法中制定排序规则， compareTo( )返回一个int值，但只会有三种结果：

1）1：大于

2）-1：小于

3）0：等于

**比较器的排序原理：**

比较器的排序原理是数据结构中的二叉树排序，二叉树排序的基本原理是：将第一个内 容作为根节点保存，如果后面的值比根节点的值小则放在根节点的左子树，如果后面的 值比根节点的值大，则放在根节点的右子树。最后根据中序遍历的原理取出。

**另一种比较器：**  
 如果一个类已经开发完成，但是忘记实现了Comparable接口，此时肯定是无法排序的， 所以再Java中又定义了另一种比较器，即Comparator接口，该接口定义在java.util包 中。

与Comparable不同的是，使用此接口需要单独创建一个类实现该接口，并指定好需 要排序的泛型类型。然后覆写其中compare( )方法，此方法需要接受两个参数。然后在 此方法中编写排序规则。此方法也是返回一个int值，并且也只有三种可能，即1、-1、 0。

**Comparable接口与Comparator接口的区别：**

Comparable接口与Comparator接口都是比较器接口，都可以用来为对象数组制定排序 规则。

但Comparable接口是在定义类时同时实现的接口，而Comparator接口是在类定义好需 要单独的创建一个类来指定比较规则，属于挽救的比较操作，是一种补救的做法。

所以，优先使用Comparable接口。

**什么是观察者设计模式：**  
 观察者设计模式是一种类似于数据监听的操作，即当某些数据发生变化后，可以立刻引 起响应。在Java中，使用Observable类和Observer接口实现观察者设计模式。

**实现观察者设计模式：**

在Java中，使用Observable类和Observer接口实现观察者设计模式。它们都定义在 java.util包中。

其中，

1）被观察的类必须继承Observerable类，通过addObsever( )方法添加观察者，通过 setChanged( )方法设置变化点，通过notifyObservers( )通知所有的观察者某数据发生 了变化。

2）每一个观察者所属的类必须实现Observer接口，并覆写update( )方法。

**正则表达式：**  
 利用正则表达式（Regular Expression）可以执行更加复杂的字符串的拆分、验证、 替 换等操作，与正则表达式有关的类定义在java.util.regex包中。

**Pattern类和Matcher类：**

要想在程序中应用正则表达式，则需要依靠Pattern类和Matcher类，这两个类都定义 在java.util.regex包中。

1. Pattern类用来制定正则规范
2. Matcher类用来执行规范，验证一个字符串是否符合其规范。

**String类对正则表达式的支持：**

在String类中有三个方法是支持正则表达式的。这三个方法的功能是用来验证字符串是 否符合其规范（matches( )方法）、按照规范拆分字符串（split( )方法）、字符串替换 （replaceAll( )方法）。

|  |  |
| --- | --- |
| public boolean matches(String regex) | 字符串验证 |
| public Sring[] split(String regex) | 字符串拆分 |
| public String replaceAll(String regex,String s) | 字符串替换 |

**定时调度：**

Java为了解决定时操作提供了Timer类和TimerTask类。这两个类都定义在java.util包 中。

1. Timer类是一种线程设施类，可以用来安排某一任务在某一时间点或者某一段时间 后执行一次或定期重复执行多次。
2. TimerTask类表示具体的任务操作类。是一个抽象类，在使用时需要建立一个类 来继承此类，并覆写run( )方法，run( )方法中为具体的任务操作。

两者必须配合使用。