链滴

Java 核心 API_02

作者：whoms

原文链接：https://ld246.com/article/1637749459506

来源网站：链滴

<h2 id="1--Objcet类">1. Objcet 类</h2>

概述

<blockquote>
<code>java.lang.Object</code> 类是 Java 语言中的根类，是所有类的父类。它中描述的所
方法子类都可以使用。在对象实例化时，最终找到的父类就是 Object。
</blockquote>
<blockquote>
如果一个类没有特别指定父类，那么默认继承 Object 类。
</blockquote>

toString 方法

<blockquote>
toString 方法返回该对象的字符串表示，字符串内容就是对象的类型 +@+ 内存地址值，在开
中经常需要重写。
</blockquote>

equals 方法

<blockquote>
equals 方法判断两个对象是否相同。相同有默认和自定义两种方式。

默认地址比较

<blockquote>
如果没有重写 equals 方法，Object 类中默认进行 <code>==</code> 运算符进行比较，只
不是同一个对象，结果必然为 false。
</blockquote>

对象内容比较

<blockquote>
如果希望对象的内容进行比较，则可以重写 equals 方法
</blockquote>
</blockquote>

Objects 类

<blockquote>
在 JDK7 中添加了一个 Objects 工具类，它提供了一些方法来操作对象，他由一些静态的使用方
组成，用于计算对象的 hashcode、返回对象的字符串表示形式、比较两个对象。
在比较两个对象是，Object 的 equals 方法容易抛出空指针异常，而 Objects 类中优化了该问
。
</blockquote>
<h2 id="2-String-StringBuilder-StringBuffer">2.String、StringBuilder、StringBuffer</h2>

三个类的区别

<blockquote>
String 类创建字符串后将被视为常量，不可修改，字符串的拼接也只是创建了新的字符串而已。
率低，但是代码复用性高。
StringBuilder 类创建字符串效率高，但是线程不安全。

StringBuffer 类创建字符串效率比 StringBuilder 第一点，但是线程安全。
</blockquote>
<h2 id="3--System">3. System</h2>
<blockquote>
<code>java.lang.System</code> 是 Java 中的系统类，主要用于获取系统的属性数据，没有造方法。类中提供了大量的静态方法，可以获取系统的相关信息或系统级操作。
</blockquote>

currentTimeMillis

<blockquote>
获取当前时间的毫秒值，有了毫秒值可以将其转换为 Date 对象、Calendar 对象等进行日期运。
</blockquote>

arraycopy

<blockquote>
将数组中指定的数据拷贝到另一个数组中，拷贝的动作是系统级的，性能很高。
<code>public static void arraycopy(Object src, int srcPos, Object dest, int destPos, int length)</code>
</blockquote>

exit

<blockquote>
exit 用于退出 Jvm，需要注意的是：0 或者非 0 的数据都可以退出 Jvm,对于用户而言没有任何别，对于 windows 是有作用的，因为如果传非 0 对于 windows 而言是异常终止的，如果是正版的作系统，对于异常退出的软件，需要把这些异常退出的软件信息做成报告发送给微软，微软就可以针这些问题对系统做出一些修改。
</blockquote>

gc

<blockquote>
在 Java 程序运行时，如果一个对象没有被引用，或者被指向了 null，那么这个对象就被标记为垃圾”，jvm 会根据某种算法来定时清除这些对象来释放内存；System.gc()方法就是告诉 jvm 的垃回收期该清理垃圾了，但 gc 方法只是建议 jvm 清理垃圾，jvm 不一定以立即去清理垃圾。
</blockquote>
<blockquote>
finalize()是 Object 类的一个方法，如果一个对象被垃圾回收器回收时，会先调用对象的该方法
</blockquote>
<h2 id="4--包装类">4. 包装类</h2>

装箱与拆箱

<blockquote>
装箱：从基本类型装换为对应的包装类对象
拆箱：从包装类对象转换为对应的基本类型
</blockquote>

<pre><code class="language-java highlight-chroma">//装箱
Integer <s
an class="highlight-n">i = <span class="highlig
t-k">new Integer(</
pan>1) <span class
"highlight-c1">//使用构造函数
Integer <s
an class="highlight-n">i1 = <span class="highli
ht-n">Integer.valu
Of(1<span c
ass="highlight-o">) //使用方法
//拆箱
int <span
lass="highlight-n">num = <span class="highlig
t-n">i.intValue</sp
n>();
</code></pre>
<blockquote>
从 JDK5 开始，基本类型的装箱与拆箱可以自动完成，自动装箱（拆箱）就是把装箱（拆箱）代
封装起来，自动执行。
</blockquote>

基本类型与字符串之间的转换

<blockquote>

基本类型转 String

<blockquote>
基本类型与空白字符串相连即可
</blockquote>

String 转对应的基本类型

<blockquote>
除 Character 类以外，其他所有的包装类都具有 parseXXX 静态方法可以将字符串参数转为对
的基本类型。
</blockquote>
<blockquote>
如果字符串参数的内容无法正确转为对应的基本类型，则抛出异常 <code>java.lang.NumberFo
matException</code>
</blockquote>
</blockquote>