首先String不属于8种基本数据类型，String是一个对象,因为对象的默认值是null,所以String的默认值也是null;但它又是一种特殊的对象，有其它对象没有的一些特性

1. 首先String不属于8种基本数据类型，String是一个对象。   
  
因为对象的默认值是null,所以String的默认值也是null;但它又是一种特殊的对象，有其它对象没有的一些特性。   
  
2. new String()和new String("")都是申明一个新的空字符串，是空串不是null;   
  
**3. String str="kvill";**  
String str=new String ("kvill");的区别：   
在这里，我们不谈堆，也不谈栈，只先简单引入常量池这个简单的概念。   
常量池(constant pool)指的是在编译期被确定，并被保存在已编译的。class文件中的一些数据。它包括了关于类、方法、接口等中的常量，也包括字符串常量。   
  
看例1:

复制代码 代码如下:

String s0="kvill";   
String s1="kvill";   
String s2="kv" + "ill";   
System.out.println( s0==s1 );   
System.out.println( s0==s2 );

结果为：   
true   
true   
首先，我们要知结果为道Java会确保一个字符串常量只有一个拷贝。   
因为例子中的s0和s1中的"kvill"都是字符串常量，它们在编译期就被确定了，所以s0==s1为true;而"kv"和"ill"也都是字符串常量，当一个字符串由多个字符串常量连接而成时，它自己肯定也是字符串常量，所以s2也同样在编译期就被解析为一个字符串常量，所以s2也是常量池中"kvill"的一个引用。   
所以我们得出s0==s1==s2;   
用new String() 创建的字符串不是常量，不能在编译期就确定，所以new String() 创建的字符串不放入常量池中，它们有自己的地址空间。   
看例2:

复制代码 代码如下:

String s0="kvill";   
String s1=new String("kvill");   
String s2="kv" + new String("ill");   
System.out.println( s0==s1 );   
System.out.println( s0==s2 );   
System.out.println( s1==s2 );

结果为：   
false   
false   
false   
例2中s0还是常量池中"kvill"的应用，s1因为无法在编译期确定，所以是运行时创建的新对象"kvill"的引用，s2因为有后半部分 newString("ill")所以也无法在编译期确定，所以也是一个新创建对象"kvill"的应用;明白了这些也就知道为何得出此结果了。   
  
**4. String.intern()：**   
  
再补充介绍一点：存在于。class文件中的常量池，在运行期被JVM装载，并且可以扩充。String的intern()方法就是扩充常量池的一个方法;当一个String实例str调用intern()方法时，Java查找常量池中是否有相同Unicode的字符串常量，如果有，则返回其的引用，如果没有，则在常量池中增加一个Unicode等于str的字符串并返回它的引用;看例3就清楚了   
例3:

复制代码 代码如下:

String s0= "kvill";   
String s1=new String("kvill");   
String s2=new String("kvill");   
System.out.println( s0==s1 );   
System.out.println( "\*\*\*\*\*\*\*\*\*\*" );   
s1.intern();   
s2=s2.intern(); //把常量池中"kvill"的引用赋给s2   
System.out.println( s0==s1);   
System.out.println( s0==s1.intern() );   
System.out.println( s0==s2 );

结果为：   
false   
\*\*\*\*\*\*\*\*\*\*   
false //虽然执行了s1.intern()，但它的返回值没有赋给s1   
true //说明s1.intern()返回的是常量池中"kvill"的引用   
true   
最后我再破除一个错误的理解：   
有人说，"使用String.intern()方法则可以将一个String类的保存到一个全局String表中，如果具有相同值的 Unicode字符串已经在这个表中，那么该方法返回表中已有字符串的地址，如果在表中没有相同值的字符串，则将自己的地址注册到表中"如果我把他说的这个全局的String表理解为常量池的话，他的最后一句话，"如果在表中没有相同值的字符串，则将自己的地址注册到表中"是错的：   
看例4:

复制代码 代码如下:

String s1=new String("kvill");   
String s2=s1.intern();   
System.out.println( s1==s1.intern() );   
System.out.println( s1+" "+s2 );   
System.out.println( s2==s1.intern() );

结果为：   
false   
kvill kvill   
true   
在这个类中我们没有声名一个"kvill"常量，所以常量池中一开始是没有"kvill"的，当我们调用s1.intern()后就在常量池中新添加了一个"kvill"常量，原来的不在常量池中的"kvill"仍然存在，也就不是"将自己的地址注册到常量池中"了。   
s1==s1.intern()为false说明原来的"kvill"仍然存在;   
s2现在为常量池中"kvill"的地址，所以有s2==s1.intern()为true.   
  
**5. 关于equals()和==:**   
  
这个对于String简单来说就是比较两字符串的Unicode序列是否相当，如果相等返回true;而==是比较两字符串的地址是否相同，也就是是否是同一个字符串的引用。   
  
**6. 关于String是不可变的**   
  
这一说又要说很多，大家只要知道String的实例一旦生成就不会再改变了，比如说：String str="kv"+"ill"+" "+"ans";   
就是有4个字符串常量，首先"kv"和"ill"生成了"kvill"存在内存中，然后"kvill"又和" " 生成 "kvill "存在内存中，最后又和生成了"kvill ans";并把这个字符串的地址赋给了str,就是因为String的"不可变"产生了很多临时变量，这也就是为什么建议用StringBuffer的原因了，因为StringBuffer是可改变的。