Object类的toString方法；StringBuffer修改字符串

以下是对在JAVA中toString方法的作用进行了详细的分析介绍，需要的朋友可以参考下

因为它是Object里面已经有了的方法，而所有类都是继承Object，所以“所有对象都有这个方法”。  
  
它通常只是为了方便输出，比如System.out.println(xx)，括号里面的“xx”如果不是String类型的话，就自动调用xx的toString()方法  
   
总而言之，它只是sun公司开发java的时候为了方便所有类的字符串操作而特意加入的一个方法  
**回答补充：**写这个方法的用途就是为了方便操作，所以在文件操作里面可用可不用  
**例子1：**

复制代码代码如下:

public class Orc  
{  
       public static class A  
       {  
              public String toString()  
              {  
                     return "this is A";  
              }  
       }  
       public static void main(String[] args)  
       {  
              A obj = new A();  
              System.out.println(obj);  
       }  
}

如果某个方法里面有如下句子：   
A obj=new A();  
System.out.println(obj);  
会得到输出:this is A  
  
http://files.jb51.net/file_images/article/201307/2013071811491621.jpg  
  
**例子2：**

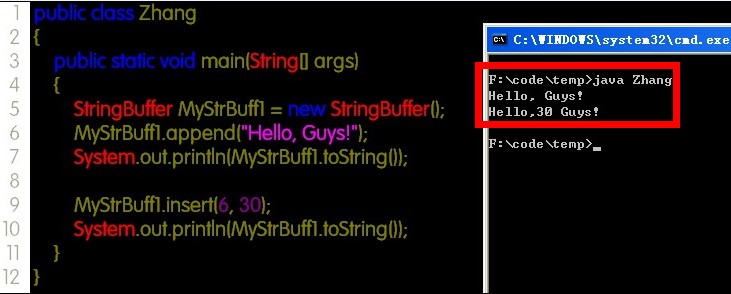
复制代码代码如下:

public class Orc  
{  
       public static class A  
       {  
              public String getString()  
              {  
                     return "this is A";  
              }  
       }  
       public static void main(String[] args)  
       {  
              A obj = new A();  
              System.out.println(obj);  
              System.out.println(obj.getString());  
       }  
}

会得到输出:xxxx@xxxxxxx的类名加地址形式  
System.out.println(obj.getString());  
会得到输出:this is A  
  
http://files.jb51.net/file_images/article/201307/2013071811491622.jpg  
  
看出区别了吗，**toString的好处是在碰到“println”之类的输出方法时会自动调用，不用显式打出来。**

复制代码代码如下:

public class Zhang  
{  
       public static void main(String[] args)  
       {  
              StringBuffer MyStrBuff1 = new StringBuffer();  
              MyStrBuff1.append("Hello, Guys!");  
              System.out.println(MyStrBuff1.toString());  
              MyStrBuff1.insert(6, 30);  
              System.out.println(MyStrBuff1.toString());  
       }  
}

**值得注意的是,** 若希望将StringBuffer在屏幕上显示出来, 则必须首先调用toString方法把它变成字符串常量, 因为PrintStream的方法println()不接受StringBuffer类型的参数.  
  


复制代码代码如下:

public class Zhang  
{  
    public static void main(String[] args)  
    {  
        String MyStr = new StringBuffer();  
        MyStr = new StringBuffer().append(MyStr).append(" Guys!").toString();  
        System.out.println(MyStr);          
    }  
}

toString()方法在此的作用是将StringBuffer类型转换为String类型.  
  


复制代码代码如下:

public class Zhang  
{  
    public static void main(String[] args)  
    {  
        String MyStr = new StringBuffer().append("hello").toString();  
        MyStr = new StringBuffer().append(MyStr).append(" Guys!").toString();  
        System.out.println(MyStr);          
    }  
}



**关于String ,StringBuffer的性能**

通过使用一些辅助性工具来找到程序中的瓶颈，然后就可以对瓶颈部分的代码进行优化。一般有两种方案：即优化代码或更改设计方法。我们一般会选择后者，因为不去调用以下代码要比调用一些优化的代码更能提高程序的性能。而一个设计良好的程序能够精简代码，从而提高性能。

下面将提供一些在JAVA程序的设计和编码中，为了能够提高JAVA程序的性能，而经常采用的一些方法和技巧。

**1．对象的生成和大小的调整。**

JAVA程序设计中一个普遍的问题就是没有好好的利用JAVA语言本身提供的函数，从而常常会生成大量的对象（或实例）。由于系统不仅要花时间生成对象，以后可能还需花时间对这些对象进行垃圾回收和处理。因此，生成过多的对象将会给程序的性能带来很大的影响。

**例1：关于String ,StringBuffer，+和append**

JAVA语言提供了对于String类型变量的操作。但如果使用不当，会给程序的性能带来影响。如下面的语句：

String name=new String("HuangWeiFeng");

System.out.println(name+"is my name");

看似已经很精简了，其实并非如此。为了生成二进制的代码，要进行如下的步骤和操作：

(1) 生成新的字符串 new String（STR\_1);

(2) 复制该字符串;

(3) 加载字符串常量"HuangWeiFeng"（STR\_2);

(4) 调用字符串的构架器（Constructor）;

(5) 保存该字符串到数组中（从位置0开始）;

(6) 从java.io.PrintStream类中得到静态的out变量;

(7) 生成新的字符串缓冲变量new StringBuffer(STR\_BUF\_1);

(8) 复制该字符串缓冲变量;

(9) 调用字符串缓冲的构架器（Constructor）;

(10) 保存该字符串缓冲到数组中（从位置1开始）;

(11) 以STR\_1为参数，调用字符串缓冲(StringBuffer)类中的append方法;

(12) 加载字符串常量"is my name"(STR\_3);

(13) 以STR\_3为参数，调用字符串缓冲(StringBuffer)类中的append方法;

(14) 对于STR\_BUF\_1执行toString命令;

(15) 调用out变量中的println方法，输出结果。

由此可以看出，这两行简单的代码，就生成了STR\_1,STR\_2,STR\_3,STR\_4和STR\_BUF\_1五个对象变量。这些生成的类的实例一般都存放在堆中。堆要对所有类的超类，类的实例进行初始化，同时还要调用类极其每个超类的构架器。而这些操作都是非常消耗系统资源的。因此，对对象的生成进行限制，是完全有必要的。

经修改，上面的代码可以用如下的代码来替换。

StringBuffer name=new StringBuffer("HuangWeiFeng");

System.out.println(name.append("is my name.").toString());

**系统将进行如下的操作：**

(1) 生成新的字符串缓冲变量new StringBuffer(STR\_BUF\_1);

(2) 复制该字符串缓冲变量;

(3) 加载字符串常量"HuangWeiFeng"(STR\_1);

(4) 调用字符串缓冲的构架器（Constructor）;

(5) 保存该字符串缓冲到数组中（从位置1开始）;

(6) 从java.io.PrintStream类中得到静态的out变量;

(7) 加载STR\_BUF\_1;

(8) 加载字符串常量"is my name"(STR\_2);

(9) 以STR\_2为参数，调用字符串缓冲(StringBuffer)实例中的append方法;

(10) 对于STR\_BUF\_1执行toString命令(STR\_3);

(11)调用out变量中的println方法，输出结果。

由此可以看出，经过改进后的代码只生成了四个对象变量：STR\_1,STR\_2,STR\_3和STR\_BUF\_1.你可能觉得少生成一个对象不会对程序的性能有很大的提高。但下面的代码段2的执行速度将是代码段1的2倍。因为代码段1生成了八个对象，而代码段2只生成了四个对象。

**代码段1：**

String name= new StringBuffer("HuangWeiFeng");

name+="is my";

name+="name";

**代码段2：**

StringBuffer name=new StringBuffer("HuangWeiFeng");

name.append("is my");

name.append("name.").toString();

因此，充分的利用JAVA提供的库函数来优化程序，对提高JAVA程序的性能时非常重要的.