毕向东老师 java 视频学习笔记

String 类常用方法总结

什么是字符串？

一个字符串就是一连串的字符，Java 定义了 String 和 StringBuffer 两个类来封装对字符串的各种操作。

String 类用于比较两个字符串、查找和抽取串中的字符或子串、字符串与其它类型之间的相互转换等。String 类对象的内容一旦被初始化就不能再改变。如果需要对字符串做很多修改，那么应该选择使用 StringBuffer 和 StringBuilder 类。

StringBuffer 类用于内容可以改变的字符串，可以将其它各种类型的数据增加、插入到字符串中，也可以转置字符串中原来的内容。一旦通过 StringBuffer 生成了最终想要的字符串，就应该使用 StringBuffer.toString 方法将其转换成 String 类，随后，就可以使用 String 类的各种方法操纵这个字符串了。

在实际开发中，如果需要频繁改变字符串的内容就需要考虑用 StringBuffer 类实现，因为其内容可以改变，所以执行性能会比 String 类更高。

String 类有十几个构造函数，其中 2 个已过时，有六十几个普通函数，1 个过时（本文参照 java API 1.6 版）。具体参照 java API 的最新版。

下面简要介绍一些常用的普通函数的用法。

一、连接字符串函数

String 类提供了连接两个字符串的方法：

第一种：String concat(String str)函数 将指定字符串连接到此字符串的结尾

用法：string1.concat(string2)，返回 string2 连接 string1 的新字符串。也可以对字符串常量使用 concat()方法，如"My name is ".concat("Zara");结果是：My name is Zara

第二种：使用'+'操作符来连接字符串，更常用。可以把其它各种类型的数据转换成字符串，并前后连接成新的字符串如：

String str="a" + " 8" + "c"，结果是："a8c"

编译时等效于

String str=new StringBuffer().append("a ").append(8).append("c").toString();

下面是一个例子:

```
public class StringDemo {
```

```
public static void main(String args[]) {
    String str = "java ";
    System.out.println("hello " + str + "123");
    }
}
```

运行结果是：hello java 123

二、equals()函数

2.1、boolean equals(Object anObject)函数将此字符串与指定的对象比较，结果为真返回 1，为假返回 0。

2.2、“==”是比较两个 String 类型引用变量指向的地址是否相等。

如下例所示：

```
public class StringTest {
    public static void main(String[] args) {
        String s1 = "abc";  // s1、s2 是 String 类型引用变量，在栈中分配。
        String s2 = "abc";  //“abc”是字符串常量，在数据区分配。
        String s3 = new String("abc"); // s3、s4 是 String 类型对象变量，在栈中分配。
        String s4 = new String("abc");  // new String("abc")在堆中分配
        System.out.println(s1==s2);  //判断 s1、s2 指向的地址是否相等：true
        System.out.println(s1.equals(s2));  //判断 s1、s2 各自的内容是否相等：true
        System.out.println(s3==s4);  //判断 s3、s4 指向的地址是否相等：false
        System.out.println(s3.equals(s4));  //判断 s3、s4 各自的内容是否相等：true
    }
}
```

运行结果是：

```
true
true
false
true
```

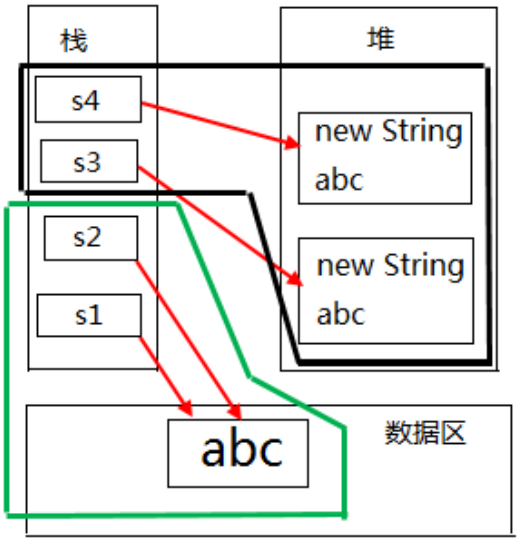

图 1 程序示意图

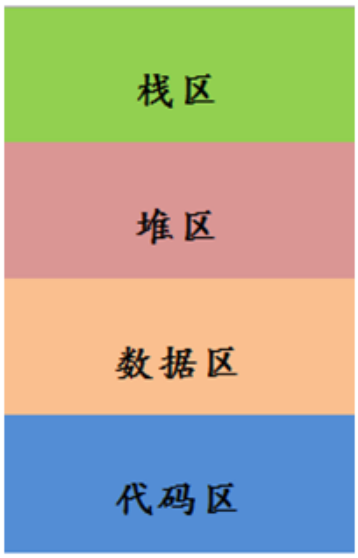

图 2 内存分配图

由图 1 可以看出，当编译器检测到 String 类型引用变量 s1、s2 都指向字符串常量“abc”时，编译器会在数据区只分配一个存储空间来保存字符常量“abc”。即 s1、s2 指向数据区的同一个地址，如图 1 绿色区域所示。String 类型的对象 s3 和 s4 指向的是各自在堆中 new 出来的存储空间，所以 s3 和 s4 指向的堆空间的地址不一样，如图 1 黑色区域所示。

三、获取

3.1、获取字符串长度函数

int length() 函数，返回此字符串的长度。如下例所示：

```
public class StringDemo {
    public static void main(String[] args) {
        String str = " str length is 16";  //定义一个引用变量，并初始化
        int len = str.length();   //获取字符串的长度
        System.out.println( "字符串长度是：" + len );//打印字符串长度信息
    }
}
```

运行结果是：

字符串长度是：16

3.2、根据位置获取字符串中对应的字符

char charAt(int index);　返回指定索引处的字符

3.3、根据字符获取在字符串中对应的位置。

int indexOf(int ch)：获取 ascii 码为 ch 的字符在字符串中第一次出现的位置。

查找成功返回位置下标，查找失败返回-1。

int indexOf(int ch, int fromIndex) ：从 fromIndex 指定位置开始，获取 ch 在字符串中出现的位置。

int indexOf(String str)：返回的是 str 在字符串中第一次出现的位置。

int indexOf(String str, int fromIndex)：从 fromIndex 指定位置开始，获取 str 在字符串中出现的位置。

int lastIndexOf(int ch)：返向索引一个字符出现位置

注意：位置值从 0 开始，从左向右计数。0 1 2 3 。。。

测试实例：

```
class StringMethodDemo  {
    public static void method_get()  {
        String str = "abcdeakpf";
        sop(str.charAt(4));  //根据索引获取字符
        //当访问到字符串中不存在的角标时会发生“StringIndex out of Bounds
        //Exception”异常。
        sop(str.indexOf(98)); //查找 ascii 码为 98 的字符的下标，小写字母 b
        sop(str.indexOf('m',3));  //从下标 3 处开始查找字符 m
        sop(str.indexOf("abc"));  //查找字符串"abc"第一次出现的位置
        sop(str.indexOf("abc",2));  //从下标 2 开始查找字符串"abc"出现的位置
        sop(str.lastIndexOf("a"));  //反向索引一个字符 a 出现位置
        // indexOf()函数如果查找到指定的内容就返回下标值，如果未找到返回-1
    }
    public static void main(String[] args)  {
        method_get();
    }
    //父类引用指向子类对象，类型提升。
    public static void sop(Object obj)  {
        System.out.print(obj+“  ”);  //用 Object 接收所以的对象（万能引用）
    }
```

}

输出结果是：

e -1 -1 0 -1 5

四、判断。

4.1、字符串中是否包含某一个子串。

boolean contains(CharSequence s);

特殊之处：indexOf(str):可以索引 str 第一次出现位置，如果返回-1.表示该 str 不在字符串中存在。所以也可以用于对指定判断是否包含。

if(str.indexOf("aa")!=-1)该方法即可以判断是否包含字符，也可获取出现的位置。

4.2、字符中是否有内容。

boolean isEmpty(): 原理就是判断长度是否为 0.

4.3、字符串是否是以指定内容开头。

boolean startsWith(str);

4.4、字符串是否是以指定内容结尾。

boolean endsWith(str);

测试实例：

```
class  StringMethodDemo  {
    public static void method_is()  {
        String str = "ArrayDemo.java";  //定义一个字符串引用变量，并赋初值
        sop(str.isEmpty());  //判断字符串 str 是否为空，空返回 true，不空返回 false
        sop(str.startsWith("Array"));  //判断字符串 str 是否是 Array 单词开头。
        sop(str.endsWith(".java"));  //判断字符串 srt 是否是.java 结尾。
        sop(str.contains("Demo"));  //判断字符串 srt 是否包含 Demo 单词
    }
    public static void main(String[] args) {
        method_is();
    }
    public static void sop(Object obj) {
        System.out.print(obj+"  ");  //0 用 Object 接收所以的对象（万能引用）
```

```
    }
}
```

输出结果是：

```
false   true   true   true
```

4.5、判断字符串内容是否相同。复写了 Object 类中的 equals 方法。

boolean equals(str); //不忽略大小写，具体见“二、equals()函数”

4.6、判断内容是否相同，并忽略大小写。

boolean equalsIgnoreCase(); //比较 2 个字符串时他会认为 A-Z 和 a-z 是一样的

测试实例：

```
class StringMethodDemo {
    public static void equals_test() {
        String s1 = "hello"; //定义变量并初始化
        String s2 = "HELLO";
        sop(s1.equals(s2)); //用 equals 比较两个字符串
        sop(s1.equalsIgnoreCase(s2)); //用 equalsIgnoreCase 比较 2 个字符串
    } // ignore ɪg'nɔːr忽略
    public static void main(String args[]) {
        equals_test();
    }
    public static void sop(Object obj) {
        System.out.print(obj+“  ”);//0 用 Object 接收所以的对象（万能引用）
    }
}
```

输出结果是：

```
false    true
```

五、转换。

5.1 将**字符数组**转成**字符串**。**重点**

构造函数：

String(char[])：把字符数组全部转换成字符串

String(char[],offset,count)：把字符数组中的一部分转成字符串。

静态方法：

static String copyValueOf(char[]); //等同于 **String(char[])**

static String copyValueOf(char[] data, int offset, int count)

//等同于 String(char[],offset,count)

测试实例：

```
class StringMethodDemo {
	public static void method_trans() {
		char[] arr = {'a','b','c','d','e','f'}; // 定义一个字符数组变量，并赋初值
		String s= new String(arr);
		String s1= new String(arr,2,3);  // 把字符数组从下标 2 开始共转换 3 个元素
		String s2 = String.copyValueOf(arr);
		String s3 = String.copyValueOf(arr,2,3);
		sop("s="+s);
		sop("s1="+s1);
		sop("s2="+s2);
		sop("s3="+s3);
	}
	public static void main(String args[]) {
		method_trans();
	}
	public static void sop(Object obj) {
		System.out.println(obj);//0 用 Object 接收所以的对象（万能引用）
	}
}
```

输出结果：

```
s=abcdef
s1=cde
s2=abcdef
```

s3=cde

5.2 将**字符串**转成**字符数组**。**重点**

char[] toCharArray():

测试实例

```
class StringMethodDemo   {
      public static void method_trans()   {
            String s1 = "HelloWorld"; //定义一个字符串引用变量，并赋初值
            char[] chs = s1.toCharArray();//把字符串转换成字符数组
            for(int x=0; x<chs.length; x++)   {
                  sop("ch ["+x+"]="+ chs [x]); //打印数组元素
            }
      }
      public static void main(String args[]) {
            method_trans();
      }
      public static void sop(Object obj) {
            System.out.print(obj+“   ”);//0 用 Object 接收所以的对象（万能引用）
      }
}
```

输出结果：

Ch[0]=H ch[1]=e ch[2]=l ch[3]=l ch[4]=o

ch[5]=W ch[6]=o ch[7]=r ch[8]=l ch[9]=d

备注：实际输出是一行，为了便于排版分成两行。

5.3 将**字节数组**转成**字符串**。

String(byte[])

String(byte[],offset,count):将字节数组中的一部分转成字符串。

5.4 将字符串转成字节数组。

byte[] getBytes():

5.5 将基本数据类型转成字符串。

字符串中特没劲的方法 valueOf()；它把基本数据型态转换成字符串。

有下列几种

static String valueOf(boolean b) : 将 boolean 变量 b 转换成字符串

static String valueOf(char c) : 将 char 变量 c 转换成字符串

static String valueOf(char[] data): 将 char 数组 data 转换成字符串

static String valueOf(char[] data, int offset, int count) :

//将 char 数组 data 的部分元素转换成字符串

static String valueOf(double d) 将 double 变量 d 转换成字符串

。。。 。。。

特殊：字符串和字节数组在转换过程中，是可以指定编码表的。

六、字符/字符串替换

String replace(oldchar,newchar);

测试实例：

```
class StringMethodDemo {
    public static void method_replace()  {
        String s = "hello java";
        String s1 = s.replace('q','n');   //把原字符串中的字符 q 替换成字符 n
        //如果要替换的字符不存在，返回的还是原串。
        String s2 = s.replace("java","world");   //把原字符串中的 java 替换成 world
        sop("s="+s);//输出源字符串
        sop("s1="+s1);
        sop("s2="+s2);
    }
    public static void main(String args[]) {
        method_replace();
    }
    public static void sop(Object obj) {
        System.out.println(obj);//0 用 Object 接收所以的对象（万能引用）
    }
}
```

输出结果：

s=hello java

s1=hello java

s2=hello world

七、字符串切割

String[] split(regex);

```
class StringMethodDemo {
	public static void	method_split() {
		String s = "张三,李四,王五";
		String s1 = "zhagnsammlisimmwangwu";
		String[] arr = s.split(","); //用逗号“，”切割字符串
		String[] arr1 = s1.split("mm"); //用逗号“mm”切割字符串
		for(int x = 0; x<arr.length; x++) {
			sop("arr["+x+"]="+arr[x]); //打印数组元素
		}
		for(int x = 0; x<arr1.length; x++) {
			sop("arr1["+x+"]="+arr1[x]); //打印数组元素
		}
	}
	public static void main(String args[]) {
		method_split();
	}
	public static void sop(Object obj) {
		System.out.println(obj);//0 用 Object 接收所以的对象（万能引用）
	}
}
```

输出结果：

arr[0]=张三

arr[1]=李四

arr[2]=王五

arr1[0]=zhagnsa

arr1[1]=lisi

arr1[2]=wangwu

八、子串 获取字符串中的一部分。

String substring(begin);

String substring(begin,end);

测试实例：

```
class StringMethodDemo {
	public static void method_sub() {
		String s = "Iloveyou";
		sop(s.substring(1));  //从指定位置开始到结尾。如果角标不存在，
                                      //会出现字符串角标越界异常。
		sop(s.substring(1,5));  //包含头，不包含尾。即只获取下标 1~下标 4 的字符
	}
	public static void main(String args[]) {
		method_sub();
	}
	public static void sop(Object obj) {
		System.out.println(obj);  //用 Object 接收所以的对象（万能引用）
	}
}
```

输出结果：

loveyou

love

九、转换 去除空格，比较。

9.1 将字符串转成大写或则小写。

String toUpperCase();

String toLowerCase();

9.2 将字符串两端的多个空格去除。

String trim();

9.3 对两个字符串进行自然顺序的比较。

```
class StringMethodDemo {
	public static void method_switch() {
		String s = "	Hello Java	";
		sop(s.toLowerCase()); //转成小写
		sop(s.toUpperCase()); //转成大写
		sop(s.trim()); //去掉空格
		//比较字符
		String s1 = "a1c";
		String s2 = "aaa";
		sop(s1.compareTo(s2));//从两个字符串的第一个字符开始比较，
		//返回两个不相等字符的差值，以后的字符就不比较了
	}
	public static void main(String args[]) {
		method_switch();
	}
	public static void sop(Object obj) {
		System.out.println(obj); //0 用 Object 接收所以的对象（万能引用）
	}
}
```

参考资料

1、毕向东老师 java 视频

2、马士兵老师 java 视频

3、Java 基础教程.pdf 作者未知

4、http://www.w3cschool.cc 菜鸟笔记

5、其他网络论坛，个人博客等

如有总结、描述错误的地方，请大家指正，为谢！