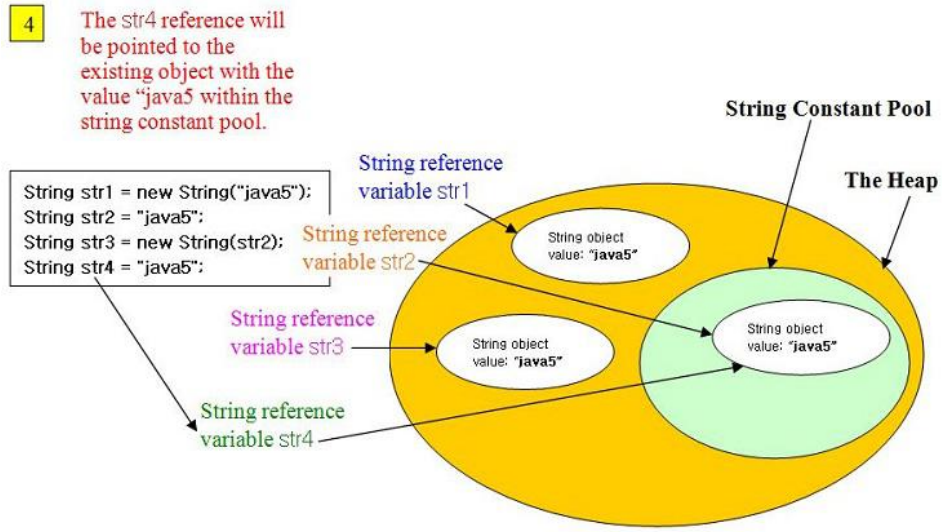
https://stackoverflow.com/questions/3052442/what-is-the-difference-between-text-and-new-stringtext




```
/*
        String s1 = "abc";//s1 是一个类类型变量， "abc"是一个对象。
                                //字符串最大特点：一旦被初始化就不可以被改变。

        String s2 = new String("abc");

        //s1 和 s2 有什么区别?
        //s1 在内存中有一个对象。
        //s2 在内存中有两个对象。
        System.out.println(s1==s2);
        System.out.println(s1.equals(s2));//String 类复写了 Object 类中 equals 方法，
                                                //该方法用于判断字符串是否相同。



/*

String 类适用于描述字符串事物。
那么它就提供了多个方法对字符串进行操作。

常见的操作有哪些?
"abcd"

1，获取。
    1.1 字符串中的包含的字符数，也就是字符串的长度。
        int length():获取长度。
    1.2 根据位置获取位置上某个字符。
        char charAt(int index):
```

1.3 根据字符获取该字符在字符串中位置。

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

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) ：

2，判断。

2.1 字符串中是否包含某一个子串。

boolean contains(str):

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)

而且该方法即可以判断，有可以获取出现的位置。

2.2 字符中是否有内容。

boolean isEmpty(): 原理就是判断长度是否为 0.

2.3 字符串是否是以指定内容开头。

boolean startsWith(str);

2.4 字符串是否是以指定内容结尾。

boolean endsWith(str);

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

boolean equals(str);

2.6 判断内容是否相同，并忽略大小写。

boolean equalsIgnoreCase();

3，转换。

3.1 将字符数组转成字符串。

构造函数：String(char[])

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

静态方法：

static String copyValueOf(char[]);

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

static String valueOf(char[]):

3.2 将字符串转成字符数组。**

char[] toCharArray():

3.3 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。

3.4 将字符串转成字节数组。

byte[] getBytes():.....IO ?

3.5 将基本数据类型转成字符串。

static String valueOf(int)

static String valueOf(double)

//3+"";//String.valueOf(3);

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。

4，替换

String replace(oldchar,newchar);

5，切割

String[] split(regex);

6，子串。获取字符串中的一部分。

String substring(begin);

String substring(begin,end);

7，转换，去除空格，比较。

7.1 将字符串转成大写或则小写。

String toUpperCase();

String toLowerCase();

7.2 将字符串两端的多个空格去除。

String trim();

7.3 对两个字符串进行自然顺序的比较。

int compareTo(string);

*/

```
class  StringMethodDemo
{

    public static void method_7()
```

```
{
	String s = "    Hello Java    ";
	sop(s.toLowerCase());
	sop(s.toUpperCase());
	sop(s.trim());

	String s1 = "a1c";
	String s2 = "aaa";

	sop(s1.compareTo(s2));
}
public static void method_sub()
{
	String s = "abcdef";

	sop(s.substring(2));//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(2,4));//包含头，不包含尾。s.substring(0,s.length());
}

public static void  method_split()
{
	String s = "zhagnsa,lisi,wangwu";

	String[] arr  = s.split(",");

	for(int x = 0; x<arr.length; x++)
	{
		sop(arr[x]);
	}
}

public static void method_replace()
{
	String s = "hello java";

	//String s1 = s.replace('q','n');//如果要替换的字符不存在，返回的还是原串。


	String s1 = s.replace("java","world");
	sop("s="+s);
	sop("s1="+s1);
}

public static void method_trans()
```

```
{
	char[] arr = {'a','b','c','d','e','f'};

	String s= new String(arr,1,3);

	sop("s="+s);

	String s1 = "zxcvbnm";

	char[] chs = s1.toCharArray();

	for(int x=0; x<chs.length; x++)
	{
		sop("ch="+chs[x]);
	}
}
public static void method_is()
{
	String str = "ArrayDemo.java";

	//判断文件名称是否是 Array 单词开头。
	sop(str.startsWith("Array"));
	//判断文件名称是否是.java 的文件。
	sop(str.endsWith(".java"));
	//判断文件中是否包含 Demo
	sop(str.contains(".java"));



}



public static void method_get()
{
	String str = "abcdeakpf";

	//长度
	sop(str.length());

	//根据索引获取字符。
	sop(str.charAt(4));//当访问到字符串中不存在的角标时会发生 StringIndexOutOfBoundsException。



	//根据字符获取索引
	sop(str.indexOf('m',3));//如果没有找到，返回-1.
```

```
		//反向索引一个字符出现位置。
		sop(str.lastIndexOf("a"));


	}
	public static void main(String[] args)
	{
		method_7();
//		method_trans();
//		method_is();
//		method_get();
		/*
		String s1 = "abc";
		String s2 = new String("abc");

		String s3 = "abc";
		System.out.println(s1==s2);
		System.out.println(s1==s3);
		*/
	}

	public static void sop(Object obj)
	{
		System.out.println(obj);
	}


}
```

/*

StringBuffer 是字符串缓冲区。

是一个容器。

特点：

1，长度是可变化的。

2，可以字节操作多个数据类型。

3，最终会通过 toString 方法变成字符串。

C create U update R read D delete

1，存储。

StringBuffer append():将指定数据作为参数添加到已有数据结尾处。

StringBuffer insert(index,数据):可以将数据插入到指定 index 位置。

2，删除。

StringBuffer delete(start,end):删除缓冲区中的数据，包含 start，不包含 end。

StringBuffer deleteCharAt(index):删除指定位置的字符。

3，获取。

char charAt(int index)

int indexOf(String str)

int lastIndexOf(String str)

int length()

String substring(int start, int end)

4，修改。

StringBuffer replace(start,end,string);

void setCharAt(int index, char ch) ;

5，反转。

StringBuffer reverse();

6，

将缓冲区中指定数据存储到指定字符数组中。

void getChars(int srcBegin, int srcEnd, char[] dst, int dstBegin)

JDK1.5 版本之后出现了 StringBuilder.

StringBuffer 是线程同步。

StringBuilder 是线程不同步。

以后开发，建议使用 StringBuilder (单线程使用，减少一步锁的判断)

升级三个因素：

1，提高效率。

2，简化书写。

3，提高安全性。

*/

```
*
基本数据类型对象包装类。

byte  Byte
short short
int         Integer
long  Long
boolean Boolean
float Float
double      Double
char  Character

基本数据类型对象包装类的最常见作用，
就是用于基本数据类型和字符串类型之间做转换
基本数据类型转成字符串。
      基本数据类型+""
      基本数据类型.toString(基本数据类型值);
      如：  Integer.toString(34);//将 34 整数变成"34";
字符串转成基本数据类型。
      xxx a = Xxx.parseXxx(String);
      int a = Integer.parseInt("123");
      double b = Double.parseDouble("12.23");
      boolean b = Boolean.parseBoolean("true");
      Integer i = new Integer("123");
      int num = i.intValue();
十进制转成其他进制。
      toBinaryString();
      toHexString();
      toOctalString();
其他进制转成十进制。
      parseInt(string,radix);
*/
class IntegerDemo
{
      public static void sop(String str)
      {
            System.out.println(str);
      }

      public static void main(String[] args)
      {
            //整数类型的最大值。
            //sop("int max :"+Integer.MAX_VALUE);
```

```
//          将一个字符串转成整数。

            int num = Integer.parseInt("123");//必须传入数字格式的字符串。
            //long x = Long.parseLong("123");

//          sop("num="+(num+4));

//          sop(Integer.toBinaryString(-6));
//          sop(Integer.toHexString(60));
            int x = Integer.parseInt("3c",16);
            sop("x="+x);
      }
}


/*
JDK1.5 版本以后出现的新特性。

*/

class IntegerDemo1
{
      public static void main(String[] args)
      {

//          Integer x = new Integer(4);

            Integer x = 4;//自动装箱。//new Integer(4)

            x = x/* x.intValue() */ + 2;//x+2:x 进行自动拆箱。变成了 int 类型。和 2 进行加法运算。
                        //再将和进行装箱赋给 x。



            Integer m = 128;
            Integer n = 128;

            sop("m==n:"+(m==n));

            Integer a = 127;
            Integer b = 127;

            sop("a==b:"+(a==b));//结果为 true。因为 a 和 b 指向了同一个 Integer 对象。
                                    //因为当数值在 byte 范围内容，对于新特性，如果该数值已经存在，则不会在开辟新的
```

空间。

```
	}

	public static void method()
	{
		Integer x = new Integer("123");

		Integer y = new Integer(123);

		sop("x==y:"+(x==y));
		sop("x.equals(y):"+x.equals(y));
	}

	public static void sop(String str)
	{
		System.out.println(str);
	}

}
```