Java语言与系统设计

第10讲 常用类

- String类
- String相关方法
- System类
- Math类

1. String类

- String类：代表字符串。Java 程序中的所有字符串字面值（如 "abc"）都作为此类的实例实现。
- String是一个final类。 String对象的字符内容是存储在一个final的字符数组value[] 。
- 字符串是常量，用双引号引起来表示。它们的值在创建之后不能更改。实现了序列化、可比较等接口。

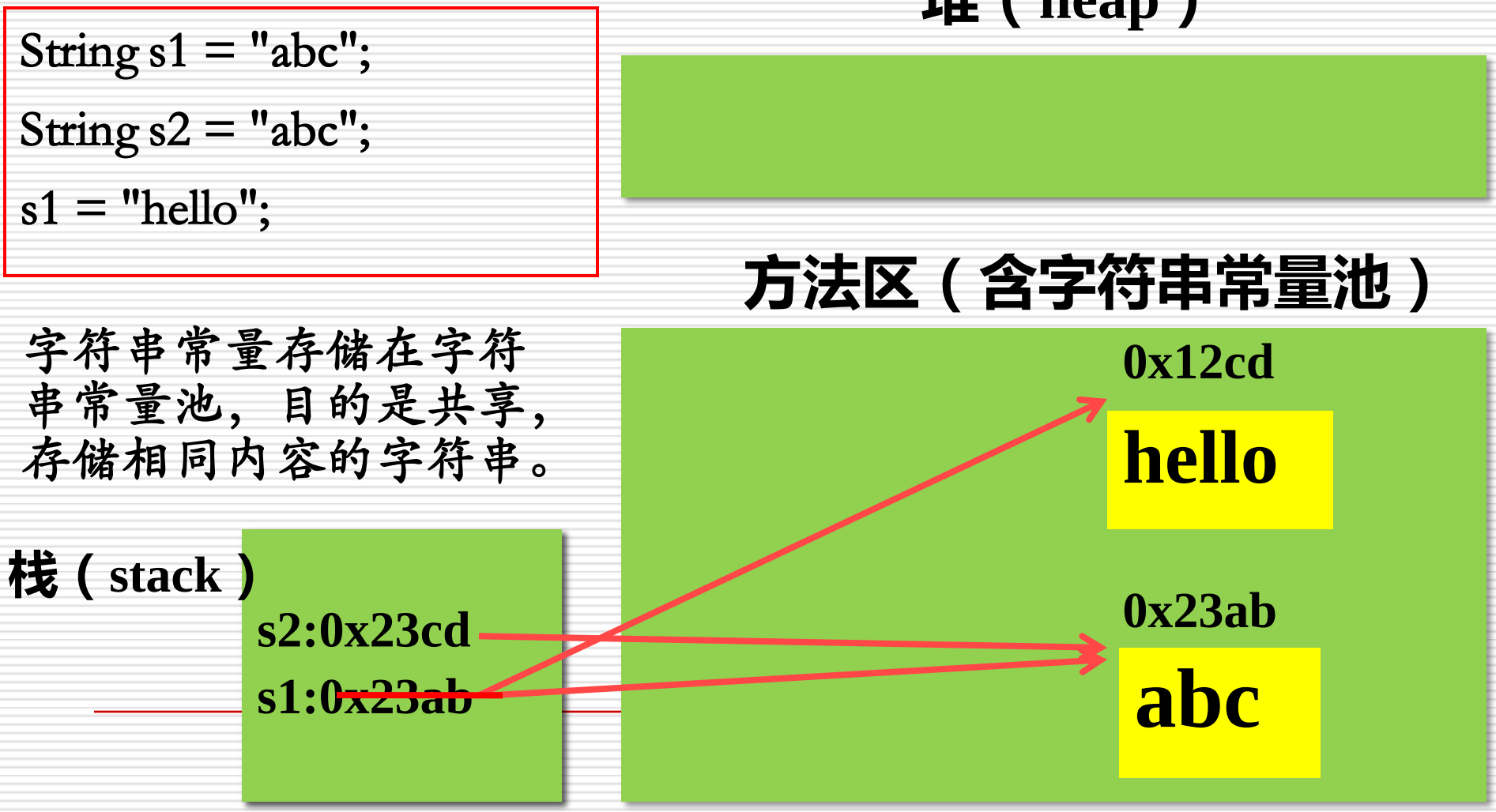
```
public final class String
    implements java.io.Serializable, Comparable<String>, CharSequence {
    /** The value is used for character storage. */
    private final char value[];

    /** Cache the hash code for the string */
    private int hash; // Default to 0
```

```
public class StringTest{
    public static void main(String[] args){
        String s1="abc";//字符串对象可以不用new
        String s2="abc";
        s1="hello";
        //System.out.println(s1==s2);
        System.out.println(s1);
        System.out.println(s2);
        //System.out.println(s1==s2);
        String s3="abc";
        s3+="def";
        System.out.println(s3);
        System.out.println(s2);
        String s4="abc";
        String s5=s4.replace('a','A');
        System.out.println(s4);
        System.out.println(s5);}}
```

字符串的内存存储实现

```
String s1 = "abc";
String s2 = "abc";
s1 = "hello";
```

字符串常量存储在字符串常量池，目的是共享，存储相同内容的字符串。



- 注意：String是不可变的序列
  - 字符串重新赋值时，是重新在新内存区赋值，而不是对原有存储区的value数组赋值。
  - 当对现有字符串进行连接操作时，也是重新指定新内存区赋值。
  - 当调用replace方法修改字符串时，也是重新指定新内存区赋值。

创建String对象的方式

```
String str = "hello";

//本质上this.value = new char[0];
String  s1 = new String();

//this.value = original.value;
String  s2 = new String(String original);

//this.value = Arrays.copyOf(value, value.length);
String  s3 = new String(char[] a);

String  s4 = new String(char[] a,int startIndex,int count);
```

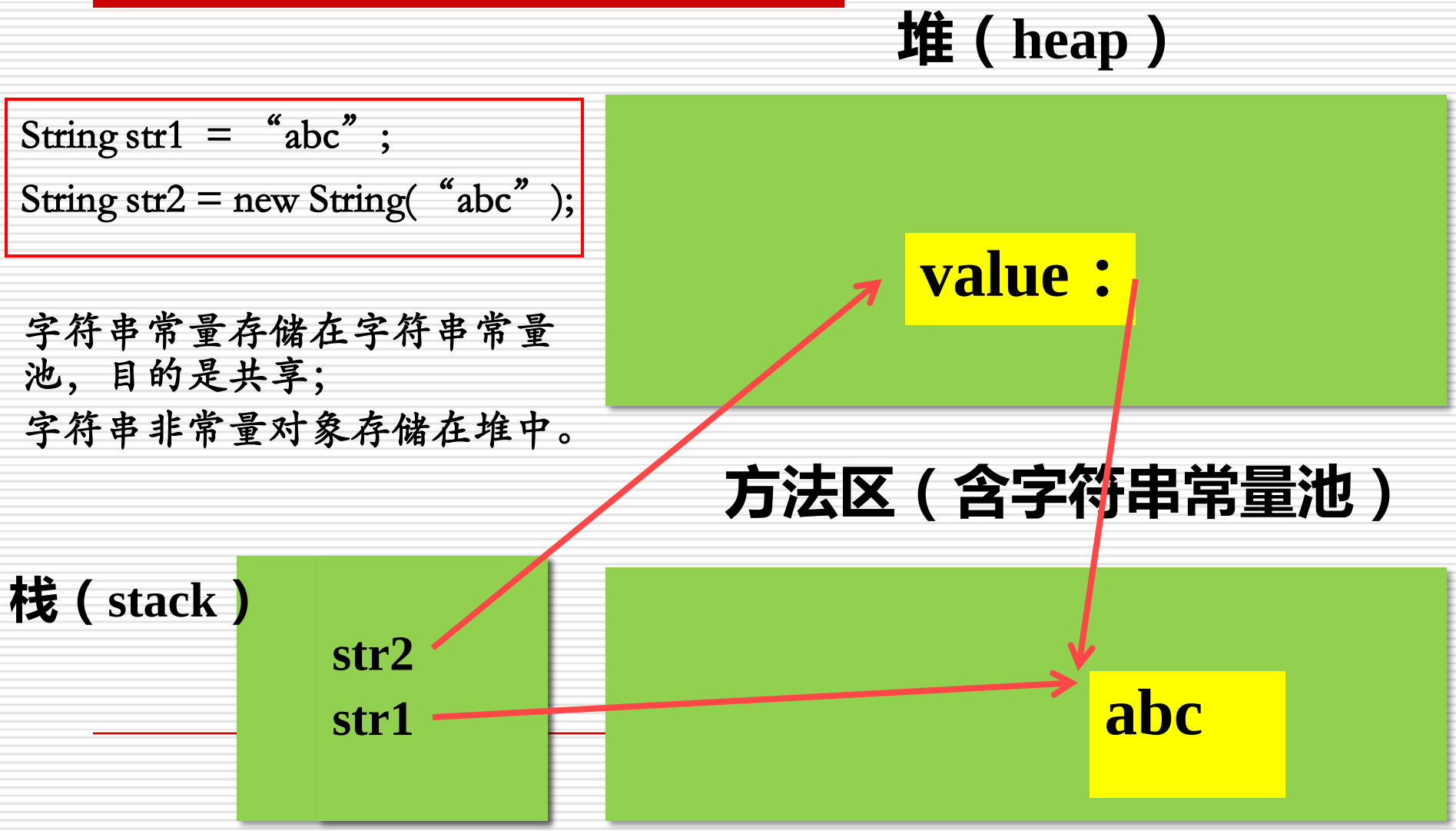
字符串的内存存储实现

String str1 = “abc”;

String str2 = new String(“abc”);

字符串常量存储在字符串常量池，目的是共享；

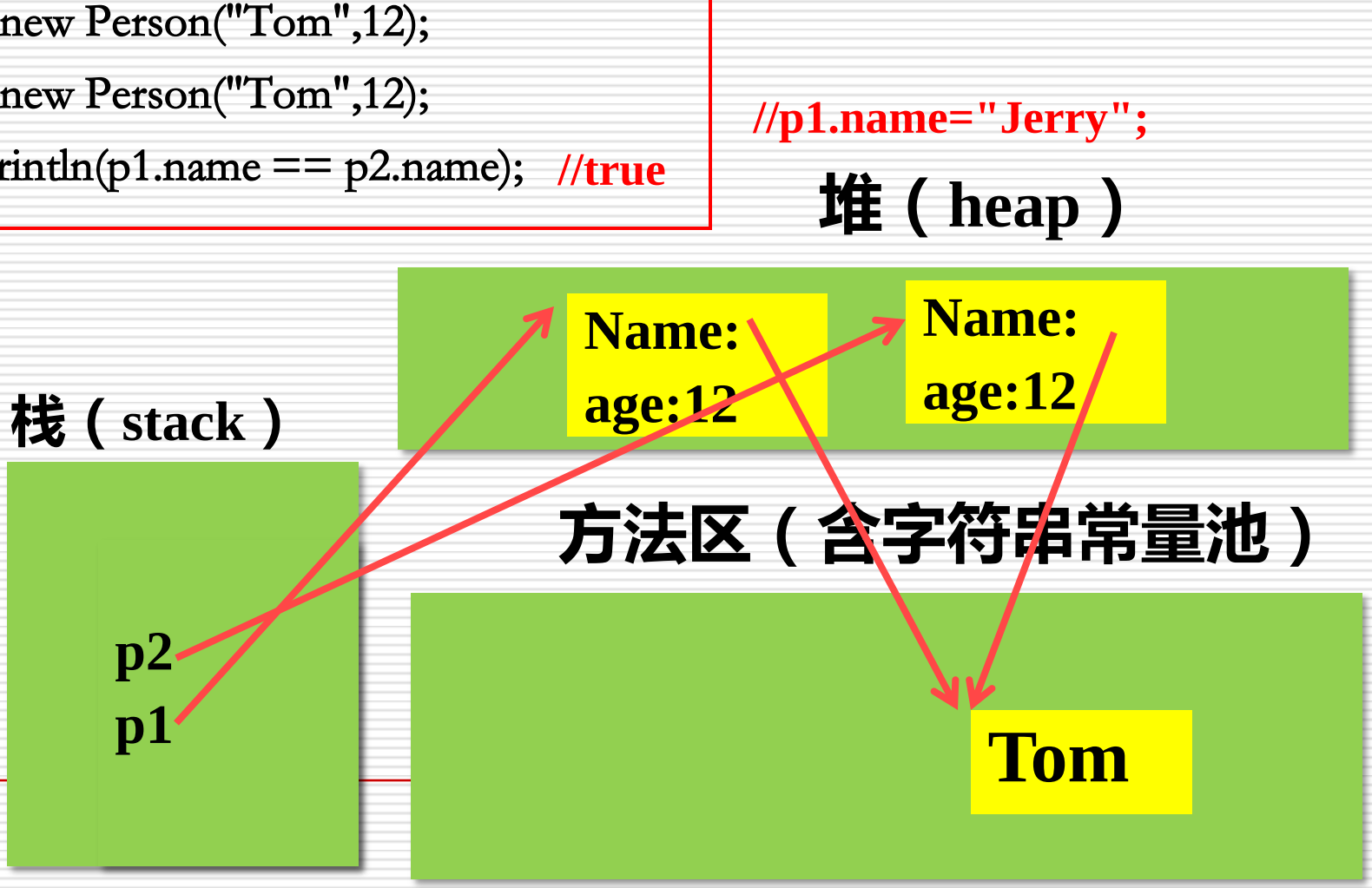
字符串非常量对象存储在堆中。



字符串的内存存储实现

```
Person p1 = new Person("Tom",12);
Person p2 = new Person("Tom",12);
System.out.println(p1.name == p2.name);   //true
```

//p1.name="Jerry";



练习

```
String s1 = "javaEE";
String s2 = "javaEE";
String s3 = new String("javaEE");
String s4 = new String("javaEE");

System.out.println(s1 == s2);
System.out.println(s1 == s3);
System.out.println(s1 == s4);
System.out.println(s3 == s4);
```

练习

```
Person p1 = new Person();
p1.name = "Tom";
Person p2 = new Person();
p2.name = "Tom";
System.out.println(p1.name.equals( p2.name));
System.out.println(p1.name == p2.name);
System.out.println(p1.name == "Tom");
String s1 = new String("bcde");
String s2 = new String("bcde");
System.out.println(s1==s2);
```

true

true

true

false

练习

String s1= new String(“abc”)在内存创建了几个对象?

答：2个。其中一个是堆空间中new创建的对象，另一个是char数组对应于常量池中的数据。如果常量池中已有“abc”，则使用已经存在的“abc”

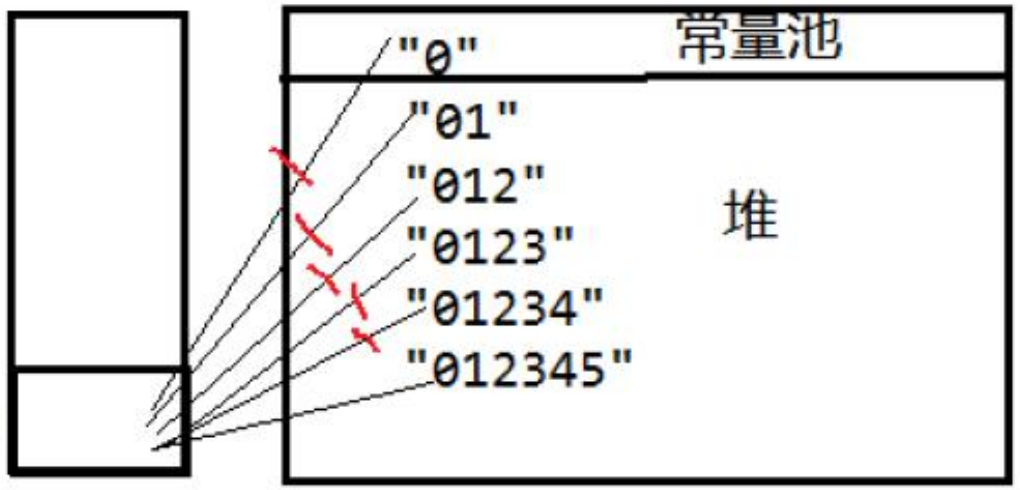
- String的连接操作
  - String s1 = "a";

说明：字符串常量池中创建一个字面量为"a"的字符串。

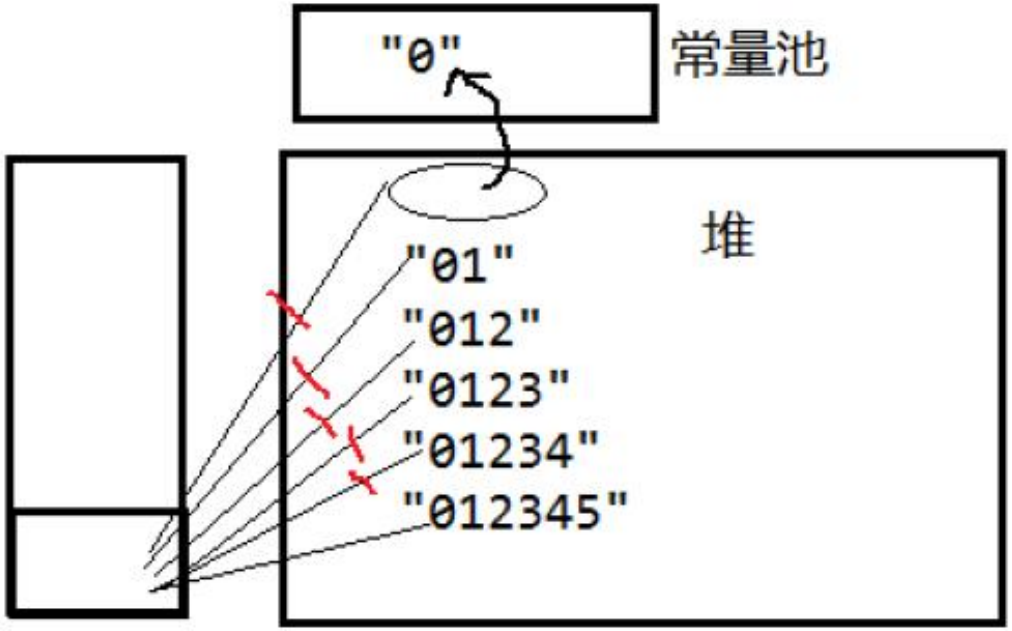
  - s1 = s1 + "b";

说明：实际上原来的“a”字符串对象已经丢弃了，现在在堆空间中产生了一个字符串s1+"b"（也就是"ab"）。如果多次执行这些改变串内容的操作，会导致大量副本字符串对象存留在内存中，降低效率。如果这样的操作放到循环中，会极大影响程序的性能。

```
String s = "0";
for(int i=1;i<=5;i++){
     s += i;
}
System.out.println(s);
```




```
String s = new String("0");
for(int i=1;i<=5;i++){
     s += i;
}
System.out.println(s);
```



- String s2 = "ab";

说明：直接在字符串常量池中创建一个字面量为"ab"的字符串。

- String s3 = "a" + "b";

说明：s3指向字符串常量池中已经创建的"ab"的字符串。

- String s4 = s1.intern();

说明：堆空间的s1对象在调用intern()之后，会将常量池中已经存在的"ab"字符串赋值给s4。

- String连接操作
  - 常量与常量的拼接结果在常量池。且常量池中不会存在相同内容的常量。
  - 只要其中有一个是变量，结果就在堆中。
  - 如果拼接的结果调用intern()方法，返回值就在常量池中。

练习

```
String s1 = "hello";

String s2 = "world";

String s3 = "hello" + "world";

String s4 = s1 + "world";

String s5 = s1 + s2;

String s6 = (s1 + s2).intern();
System.out.println(s3==s4);
System.out.println(s3==s5);
System.out.println(s4==s5);
System.out.println(s3==s6);
```

练习

引用数据类型传地址，实参的值会改值，但String是不可变的

```
public class StringTest {

    String str = new String("good");
    char[] ch = { 't', 'e', 's', 't' };

    public void change(String str, char ch[]) {
        str = "test ok";
        ch[0] = 'b';
    }
    public static void main(String[] args) {
        StringTest ex = new StringTest();
        ex.change(ex.str, ex.ch);
        System.out.print(ex.str + " and ");//
        System.out.println(ex.ch);
    }
}
```

good and best

String相关方法

- int length()：返回字符串的长度：return value.length
- char charAt(int index)：返回某索引处的字符return value[index]
- Boolean isEmpty()：判断是否是空字符串：return value.length==0
- String toLowerCase()：使用默认语言环境，将String中的所有字符转换为小写
- String toUpperCase()：使用默认语言环境，将String中的所有字符转换为大写

String相关方法

- String trim()：返回字符串的副本，忽略前导空白和尾部空白
- boolean equals(Object obj)：比较字符串的内容是否相同
- boolean equalsIgnoreCase(String anotherString)：与equals方法类似，忽略大小写
- String concat(String str)：将指定字符串连接到此字符串的结尾。等价于用“+”
- int compareTo(String anotherString)：比较两个字符串的大小

String相关方法

- String substring(int beginIndex)：返回一个新的字符串,它是此字符串的从beginIndex开始截取到最后的一个子字符串。
- String substring(int beginIndex, int endIndex)：返回一个新字符串，它是此字符串从beginIndex开始截取到endIndex(不包含)的一个子字符串。
- boolean endsWith(String suffix)：测试此字符串是否以指定的后缀结束
- boolean startsWith(String prefix)：测试此字符串是否以指定的前缀开始
- boolean startsWith(String prefix, int toffset)：测试此字符串从指定索引开始的字符串是否以指定前缀开始

String相关方法

- boolean contains(CharSequence s)：当且仅当此字符串包含指定的char 值序列时，返回true
- int indexOf(String str)：返回指定子字符串在此字符串中第一次出现处的索引
- int indexOf(String str, int fromIndex)：返回指定子字符串在此字符串中第一次出现处的索引，从指定的索引开始
- int lastIndexOf(String str)：返回指定子字符串在此字符串中最右边出现处的索引
- int lastIndexOf(String str, int fromIndex)：返回指定子字符串在此字符串中最后一次出现处的索引，从指定的索引开始反向搜索

String相关方法

- String replace(char oldChar, char newChar)：返回一个新字符串，它是通过用newChar替换此字符串中出现的所有oldChar得到。
- String replaceAll(String regex, String replacement): 使用replacement替换此字符串所有匹配给定的正则表达式的子字符串。
- String replaceFirst(String regex, String replacement): 使用给定的replacement替换此字符串匹配给定的正则表达式的第一个子字符串。

String相关方法

- boolean matches(String regex)：告知此字符串是否匹配给定的正则表达式。
- String[] split(String regex)：根据给定正则表达式的匹配拆分此字符串。
- String[] split(String regex, intl imit)：根据匹配给定的正则表达式来拆分此字符串，最多不超过limit个，如果超过了，剩下的全部都放到最后一个元素中.

StringBuffer类

- java.lang.StringBuffer代表可变的字符序列，JDK1.0中声明，可以对字符串内容进行增删，此时不会产生新的对象。
- 很多方法与String相同。
- 作为参数传递时，方法内部可以改变值。

```
abstract class AbstractStringBuilder implements Appendable, CharSequence {
    /**
     * The value is used for character storage.
     */
    char[] value;

    /**
     * The count is the number of characters used.
     */
    int count;
```

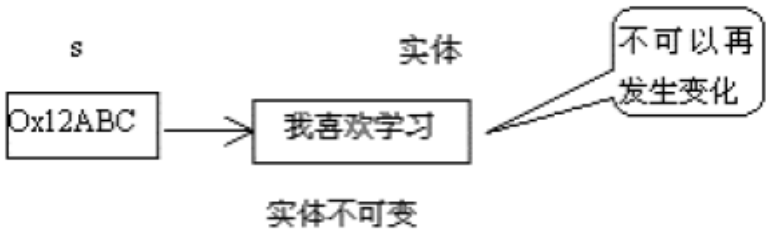
value没有final声明,value可以不断扩容。

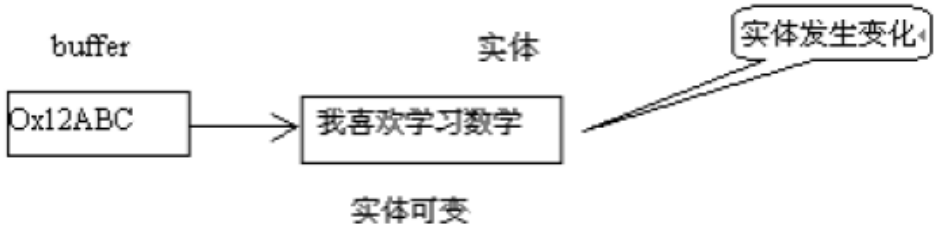
count记录有效字符的个数。

StringBuffer类

- StringBuffer类不同于String，其对象必须使用构造器生成。有三个构造器：
- StringBuffer()：初始容量为16的字符串缓冲区
- StringBuffer(int size)：构造指定容量的字符串缓冲区
- StringBuffer(String str)：将内容初始化为指定字符串内容

```
String s = new String("我喜欢学习");
StringBuffer buffer = new StringBuffer("我喜欢学习");
buffer.append("数学");
```





StringBuffer类

- StringBuffer append(xxx)：提供了很多的append()方法，用于进行字符串拼接
- StringBuffer delete(int start, int end)：删除指定位置的内容
- StringBuffer replace(int start, int end, String str)：把[start,end)位置替换为str
- StringBuffer insert(int offset, xxx)：在指定位置插入xxx
- StringBuffer reverse()：把当前字符序列逆转
- 当append和insert时，如果原来value数组长度不够，可扩容。

StringBuffer类

- StringBuilder和StringBuffer 非常类似，均代表可变的字符序列，而且提供相关功能的方法也一样。
- String(JDK1.0)：不可变字符序列
- StringBuffer(JDK1.0)：可变字符序列、效率低
- StringBuilder(JDK 5.0)：可变字符序列、效率高
- 注意：作为参数传递的话，方法内部String不会改变其值，StringBuffer和StringBuilder会改变其值。

3. System类

- System类代表系统，系统级的很多属性和控制方法都放置在该类的内部。该类位于java.lang包。
  - 由于该类的构造器是private的，所以无法创建该类的对象，也就是无法实例化该类。其内部的成员变量和成员方法都是static的，所以也可以很方便的进行调用。
- 成员变量
  - System类内部包含in、out和err三个成员变量，分别代表标准输入流(键盘输入)，标准输出流(显示器)和标准错误输出流(显示器)。

3. System类

- 成员方法
- native long currentTimeMillis()：该方法的作用是返回当前的计算机时间，时间的表达格式为当前计算机时间和GMT时间(格林威治时间)1970年1月1号0时0分0秒所差的毫秒数。
- void exit(int status)：该方法的作用是退出程序。其中status的值为0代表正常退出
- void gc()：该方法的作用是请求系统进行垃圾回收。至于系统是否立刻回收，则取决于系统中垃圾回收算法的实现以及系统执行时的情况。

3. System类

- 成员方法
  - String getProperty(String key)：该方法的作用是获得系统中属性名为key的属性对应的值。系统中常见的属性名以及属性：

属性名	属性说明
java.version	Java 运行时环境版本
java.home	Java 安装目录
os.name	操作系统的名称
os.version	操作系统的版本
user.name	用户的账户名称
user.home	用户的主目录
user.dir	用户的当前工作目录

4. Math类

- java.lang.Math提供了一系列静态方法用于科学计算。其方法的参数和返回值类型一般为double型。

abs　绝对值
acos,asin,atan,cos,sin,tan　三角函数
sqrt　平方根
pow(double a,doble b)　a的b次幂
log　自然对数
exp　e为底指数
max(double a,double b)
min(double a,double b)
random()　返回0.0到1.0的随机数
long round(double a)　double型数据a转换为long型（四舍五入）
toDegrees(double angrad)　弧度—>角度
toRadians(double angdeg)　角度—>弧度

1．用运算符“==”比较字符串对象时，只要两个字符串包含的是同一个值，结果便为 true。（） 错

2．String 类字符串在创建后可以被修改。（） 错

3．方法 replace (String srt1, String srt2)将当前字符串中所有 srt1 子串换成 srt2子串。 对

4．方法compareTo在所比较的字符串相等时返回 0。（） 对

5．方法IndexOf((char ch,-1)返回字符ch在字符串中最后一次出现的位置。（） 错

6．方法startsWith()判断当前字符串的前缀是否和指定的字符串一致。（） 对

◆ String s = “good”；下面选项正确的是：

A. s += “student”； **A**

B. char c = s[1]; **D**

C. int len = s .length;

D. String t = s.toLowerCase( );

- String s = “people”; String t = “people”;
  String c[] = {“p”,”e”,”o”,”p”,”l”,”e”};
  下面哪一选项的语句返回值为真:

A. s .equals(t);

B. t .equals(c);

C. s==t;

D. t .equals(new String(“people”));

E. t==c;

A

C

D