String

1.1String实例化

1.直接赋值方式

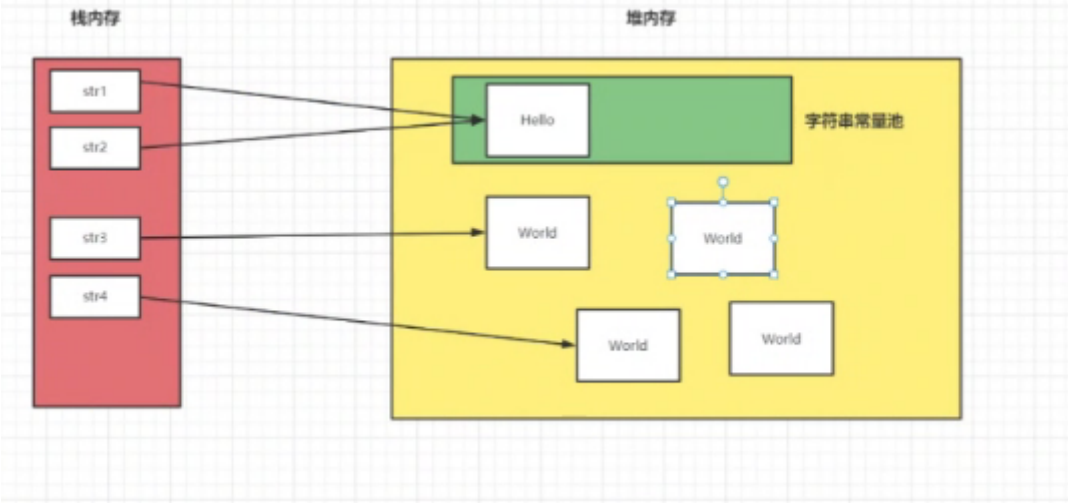
```
String str="hello";
```

2.通过构造器创造的方式

```
String str2 = new String("hello")
```

```
public class Test {
    public static void main(String[] args) {
        String str1="hello";
        String str2="hello";
        System.out.println(str1==str2);
        String str3= new String("world");
        String str4= new String("world");
        System.out.println(str3==str4);
    }
}
```

判断两个对象的内存地址是否一致，判断两个对象是不是同一块内存区域



字符串常量池是java在堆内存中开辟的一块空间，专门用来存储String对象，使用字符串常量池，可以节约内存空间，只要是直接赋值的方式，数据都会存储到字符串常量池中。

通过直接赋值的方式创建一个字符串对象”hello“的时候，首先检查字符传常量池，”hello“是否已经存在于常量池中，如果存在则直接引用，如果不存在，先创建再引用

以上的创建机制做到了资源的重复利用，需要创建很多重复的字符串对象，只需要在常量池中进行多次引用即可，而不需要重复创建相同的数据

而通过构造器创建的对象，字符串不在存储到字符串常量池中，而是直接存储到堆内存中，此时不会去进行资源的重复利用，只要创建一个对象，就会独享一块内存区域，哪怕对象的值全部是一样的，也不会进行共享，都是独立的

```
public class Test {
```

```
    public static void main(String[] args) {
        String str1="hello";
        String str2="hello";
        System.out.println(str1==str2);
        String str3= new String("world");
        String str4= new String("world");
        System.out.println(str3==str4);

//          包装类常量池是有范围的值必须在：-128-127:
        Integer n1=1;
        Integer n2=1;
        System.out.println(n1==n2);

        Integer n5=-129;
        Integer n6=-129;
        System.out.println(n5==n6);//false

        Integer n3= new Integer(1);
        Integer n4= new Integer(1);
        System.out.println(n3==n4);
    }
}
```

```
Integer i1 = 40;

Integer i2 = 40;

Integer i3 = 0;

Integer i4 = new Integer(40);

Integer i5 = new Integer(40);

Integer i6 = new Integer(0);

System.out.println("i1=i2 " + (i1 == i2)); //输出 i1=i2 true

System.out.println("i1=i2+i3 " + (i1 == i2 + i3)); //输出 i1=i2+i3 true

//i2+i3得到40,比较的是数值

System.out.println("i1=i4 " + (i1 == i4)); //输出 i1=i4 false

System.out.println("i4=i5 " + (i4 == i5)); //输出 i4=i5 false

//i5+i6得到40，比较的是数值

System.out.println("i4=i5+i6 " + (i4 == i5 + i6)); //输出 i4=i5+i6 true

System.out.println("40=i5+i6 " + (40 == i5 + i6)); //输出 40=i5+i6 true
```

1.2 判断两个字符串是否相等

不能使用==去判断，==判断的是内存地址，字符串相等是指字符串的值是否一致，而不是她们是否是同一块内存

判断字符串是否相等，一般使用equals方法来完成，并不是String类中定义的方法，而是从Object中继承过来的方法

Object中原生的方法定义：

```
public boolean equals(Object anObject){
    return this ==anObject;
}
```

Stirng类中对方法的重写

```
public boolean equals(Object anObject) {
    if (this == anObject) {
        return true;
    }
    if (anObject instanceof String) {
        String anotherString = (String)anObject;
        int n = value.length;
        if (n == anotherString.value.length) {
            char v1[] = value;
            char v2[] = anotherString.value;
            int i = 0;
            while (n-- != 0) {
                if (v1[i] != v2[i])
                    return false;
                i++;
            }
            return true;
        }
    }
    return false;
}
```

1.1.3String常用方法

方法	描述
public String()	创建一个空的字符串对象
public String(String o)	创建一个有值的字符串对象
public String(char value[])	将一个char数组转为字符串对象
public String(char value[],int offset,int count)	将一个指定范围的 char 数组转成字符串对象
public String(byte[] bytes)	将一个byte数组转为字符串对象
public String(byte[] bytes,int offset,int count)	将一个指定范围的byte数组转为字符串对象
public int length()	返回字符串的长度
public boolean isEmpty()	判断字符串是否为空
public char charAt(int index)	返回字符串中指定位置的字符
public byte[] getBytets()	将字符串转为byte数组
public boolean equals(Object anObject)	判断两个字符串是否相等
public boolean equalsIgnoreCase(String str)	判断两个字符串是否相等并且忽略大小写
public int compareTo(String str)	对两个字符串进行排序
public boolean startsWith(String str)	判断是否以指定值开头
public boolean endsWith(String str)	判断是否以指定值结尾
public int hashCode()	获取字符串的散列码
public int indexOf(String str)	从头开始查找指定字符串的位置
public int indexOf(String str,int index)	从 index 开始查找指定字符串的位置
public String substring(int index)	截取字符串从指定位置到结尾
public String substring(int index1,int index2)	截取字符串在指定区间的值
public String concat(String str)	追加字符串
public String replaceAll(String str1,String str2)	替换字符串
public String[] split(String str)	根据指定字符串进行截取
public String toLowerCase()	全部转小写
public String toUpperCase()	全部转大写
public char[] toCharArray()	将字符串转为char数组

```
byte[] bytes = {65,66,67,68,69,70};
String str = new String(bytes);
System.out.println(str.charAt(1));
str = "Hello World";
byte[] bytes1 = str.getBytes();
System.out.println(Arrays.toString(bytes1));
String str1 = "B";
```

```
String str2 = "B";
System.out.println(str1.compareTo(str2));
System.out.println(str.startsWith("He"));
System.out.println(str.endsWith("ab"));
System.out.println(str.hashCode());
System.out.println(str1.hashCode());
System.out.println(str2.hashCode());
System.out.println(str.indexOf("z"));
System.out.println(str.indexOf("z", 5));
String substring = str.substring(3);
System.out.println(substring);
String substring1 = str.substring(3, 5);
System.out.println(substring1);
String concat = str.concat("666");
System.out.println(concat);
String replace = str.replace("l", "a");
System.out.println(replace);
String str3 = "Tom,Cat,Jack,ABC,123";
String[] split = str3.split(",");
for (int i = 0; i < split.length; i++) {
    System.out.println(split[i]);
}
String str4 = "AbCdEf";
String str5 = str4.toLowerCase();
System.out.println(str5);
String str6 = str4.toUpperCase();
System.out.println(str6);
char[] chars = str.toCharArray();
for (int i = 0; i < chars.length; i++) {
    System.out.println(chars[i]);
}
```

```
B
[72, 101, 108, 108, 111, 32, 87, 111, 114, 108, 100]
0
true
false
-862545276
66
66
-1
-1
lo World
lo
Hello World666
Heaao Worad
Tom
Cat
Jack
ABC
123
abcdef
ABCDEF
H
e
l
l
o

W
o
r
l
d
```