

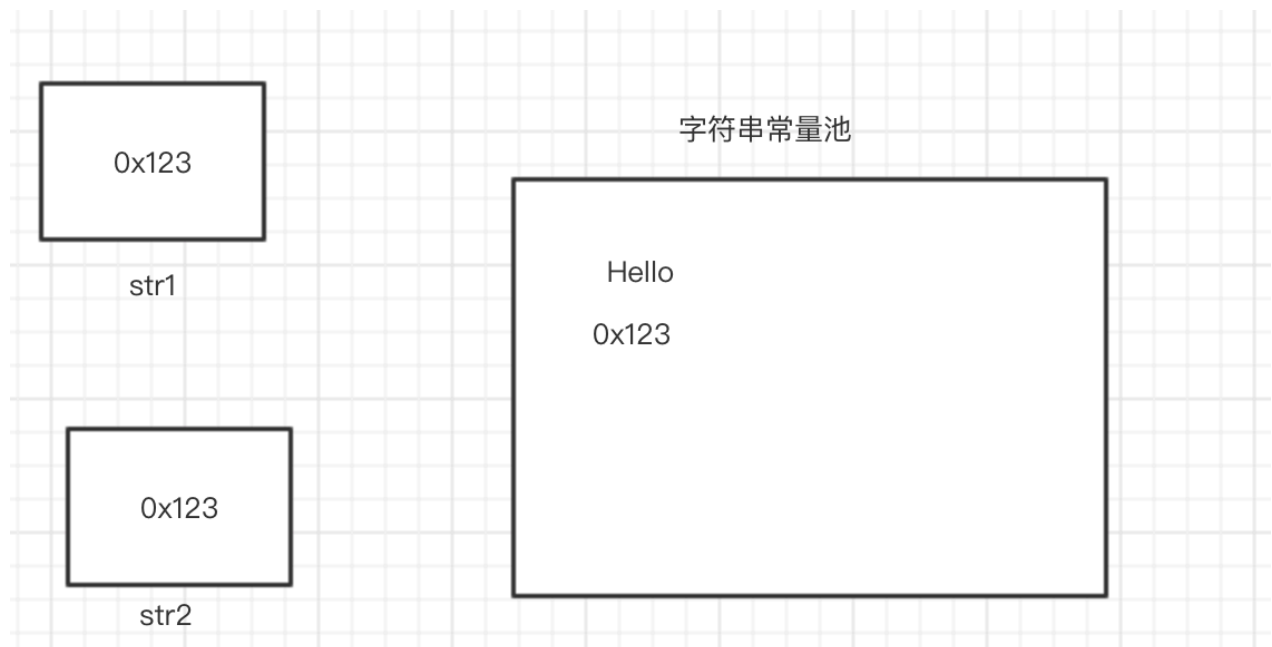
String 3 大特性

- 不变性：是一个 immutable 模式的对象，不变模式的主要作用是当一个对象需要被多线程共享并频繁访问时，可以保证数据的一致性。
- 常量池优化：String 对象创建之后，会在字符串常量池中进行缓存，下次创建同样的对象时，会直接返回缓存的引用。
- final：String 类不可被继承，提高了系统的安全性。

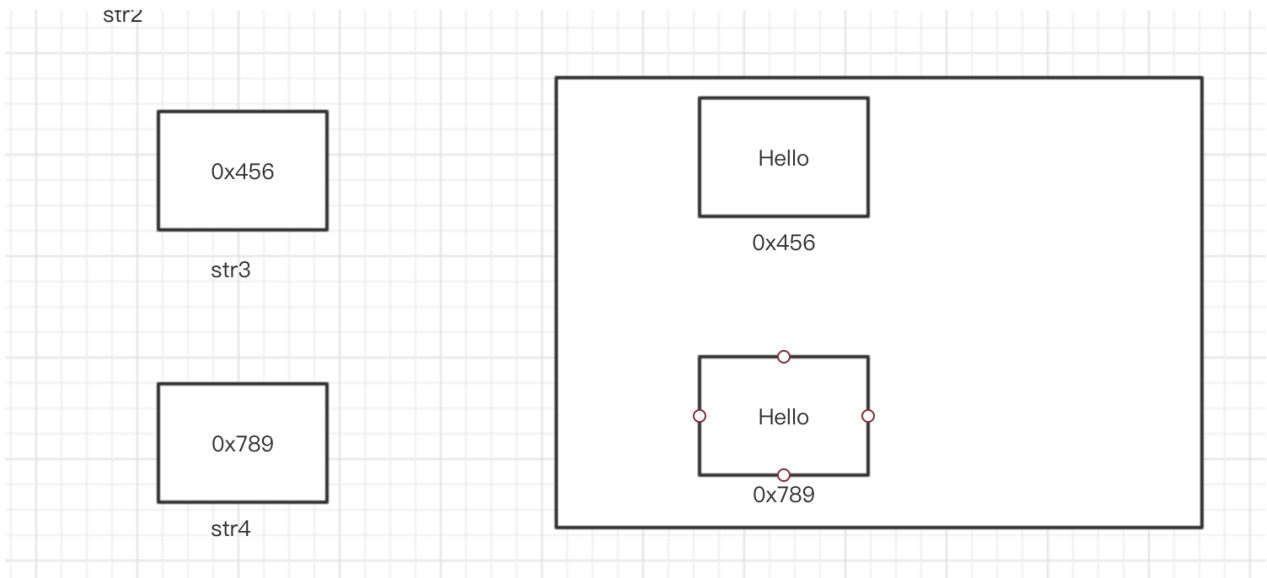
String 的实例化有两种方式：

- 直接赋值
- 通过构造函数，可以直接将字符串的值传入，也可以传入一个 char 数组。

直接赋值和通过构造函数创建主要区别在于存储的区域不同，直接赋值存储在字符串常量池中。



通过构造函数创建，存储在堆内存中。



equals 重写

```
public boolean equals(Object anObject) {
    if (this == anObject) {
        return true;
    }
    if (anObject instanceof String) {
        String aString = (String)anObject;
        if (coder() == aString.coder()) {
            return isLatin1() ? StringLatin1.equals(value, aString.value)
                : StringUTF16.equals(value, aString.value);
        }
    }
    return false;
}
```

因为 String 类对 equals 方法进行了重写，所以我们可以直接调用String 的 equals 方法来判断两个字符串的值是否相等。

intern() 方法

当调用某个字符串对象的 intern() 方式，会去字符串常量池中寻找，如果已经存在一个值相等的字符串对象的话，则直接返回该对象的引用，如果不存在，则在字符串常量池中创建该对象，并返回。

String 常用的方法

1、字符串截取

```
public String substring(int beginIndex);
public String substring(int beginIndex, int endIndex);
```

subString(int beginIndex) 从 beginIndex 位置开始截取，一直到字符串结尾。

subString(int beginIndex,int endIndex) 从 beginIndex 位置开始截取，一直到 endIndex 结束，同时不包含 endIndex。

2、字符串分割

```
public String[] split(String regex);
```

split 方法支持正则表达式，进行复杂的字符串分割。

```
String str = new String("Hello,World;Java-String");
String[] array = str.split("[,;|-]");
for(String item:array){
    System.out.println(item);
}
```

• String 的常用方法

方法	描述
public String()	创建一个值为空的对象
public String(String original)	创建一个值为original的对象
public String(char value[])	将一个char型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型数组转为字符串对象
public String(byte[] bytes)	将一个byte型数组转为字符串对象
public String(byte bytes[], int offset, int length)	将一个指定范围的byte型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[] getBytes()	将字符串转为byte型数组
public boolean equals(Object anObject)	判断两个字符串是否相等
public boolean equalsIgnoreCase(String anotherString)	判断两个字符串是否相等并且忽略大小写

public int compareTo(String anotherString)	对两个字符串进行排序
public boolean startsWith(String prefix)	判断是否以指定的值开头
public boolean endsWith(String suffix)	判断是否以指定的值结尾
public int hashCode()	获取字符串的散列值
public int indexOf(String str)	从头开始查找指定字符的位置
public int indexOf(String str, int fromIndex)	从指定的位置开始查找指定字符的位置
public String substring(int beginIndex)	截取字符串从指定位置开始到结尾
public String substring(int beginIndex, int endIndex)	截取字符串从指定位置开始到指定位置结束
public String concat(String str)	追加字符串
public String replaceAll(String regex, String replacement)	替换字符串
public String[] split(String regex)	用指定字符串对目标字符串进行分割，返回数组
public String toLowerCase()	将字符串转为小写
public String toUpperCase()	将字符串转为大写
public char[] toCharArray()	将字符串转为char型数组

```

char[] array =
{'J','a','v','a',' ',' ','H','e','l','l','o',' ',' ','W','o','r','l','d'};
String str = new String(array);
System.out.println(str);
System.out.println("str长度: "+str.length());
System.out.println("str是否为空: "+str.isEmpty());
System.out.println("下标为2的字符是: "+str.charAt(2));
System.out.println("h的下标是: "+str.indexOf('H'));
String str2 = "Hello";
System.out.println("str和str2是否相等: "+str.equals(str2));
String str3 = "HELLO";
System.out.println("str2和str3忽略大小写是否相等: "+str2.equalsIgnoreCase(str3));
System.out.println("str是否以Java开头: "+str.startsWith("Java"));
System.out.println("str是否以Java结尾: "+str.endsWith("Java"));
System.out.println("从2开始截取str: "+str.substring(2));
System.out.println("从2到6截取str: "+str.substring(2, 6));
System.out.println("将str中的World替换为Java: "+str.replaceAll("World", "Java"));
System.out.println("用逗号分割str: "+Arrays.toString(str.split(",")));
System.out.println("将str转为char类型数组: "+Arrays.toString(str.toCharArray()));
System.out.println("str3转为小写: "+str3.toLowerCase());

```

```
System.out.println("str2转为大写: "+str2.toUpperCase());
```

运行结果如下图所示。

```
/Library/Java/JavaVirtualMachines/jdk-10.0.1.jdk/Contents/Home/bin/java "
Java,Hello,World
str长度: 16
str是否为空: false
下标为2的字符是: v
H的下标是: 5
str和str2是否相等: false
str2和str3忽略大小写是否相等: true
str是否以Java开头: true
str是否以Java结尾: false
从2开始截取str: va,Hello,World
从2到6截取str: va,H
将str中的World替换为Java: Java,Hello,Java
用逗号分割str: [Java, Hello, World]
将str转为char类型数组: [J, a, v, a, ,, H, e, l, l, o, ,, W, o, r, l, d]
str3转为小写: hello
str2转为大写: HELLO
```

1、字符串截取

字符串截取是 String 最常用的操作之一，String 提供了两个截取字符串的方法。

```
public String substring(int beginIndex)
public String substring(int beginIndex, int endIndex)
```

substring(int beginIndex) 是从下标为 beginIndex 的位置开始截取，一直到字符串的结尾。

substring(int beginIndex, int endIndex) 是从下标为 beginIndex 的位置开始截取，一直到 endIndex 的位置结束，且不包括该位置的字符。

需要注意的是 substring 方法会将截取的结果以新字符串的形式返回，原字符串的值不会改变，具体操作如下所示。

```
String str = new String("Hello World");
System.out.println(str.substring(3));
System.out.println(str);
```

运行结果如下图所示。

```
/Library/Java/JavaVirtualMachines,
lo World
Hello World
```

2、字符串分割

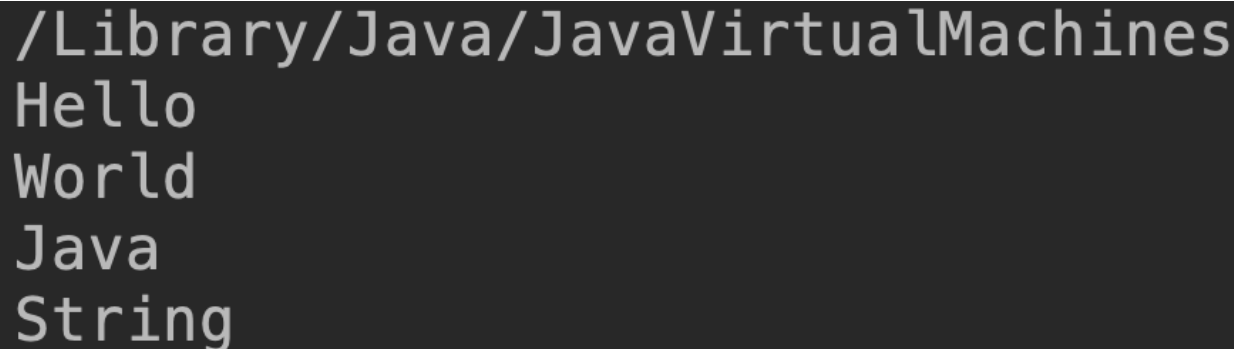
字符串分割也是 String 最常用的操作之一，它是指将目标字符串按照某个分割符，切割成一个字符串数组，String 通过 split 方法完成字符串分割。

```
public String[] split(String regex)
```

split 方法支持传入正则表达式，进行复杂的字符串分割，比如 "Hello,World;Java-String"，如果要分别将 Hello、World、Java、String 截取出来，使用统一的某个分割符肯定无法完成，这时候可以借助于正则表达式 "[,;|-]" 来实现，具体操作如下所示。

```
String str = new String("Hello,World;Java-String");
String[] array = str.split("[,;|-]");
for (String item:array){
    System.out.println(item);
}
```

运行结果如下图所示。



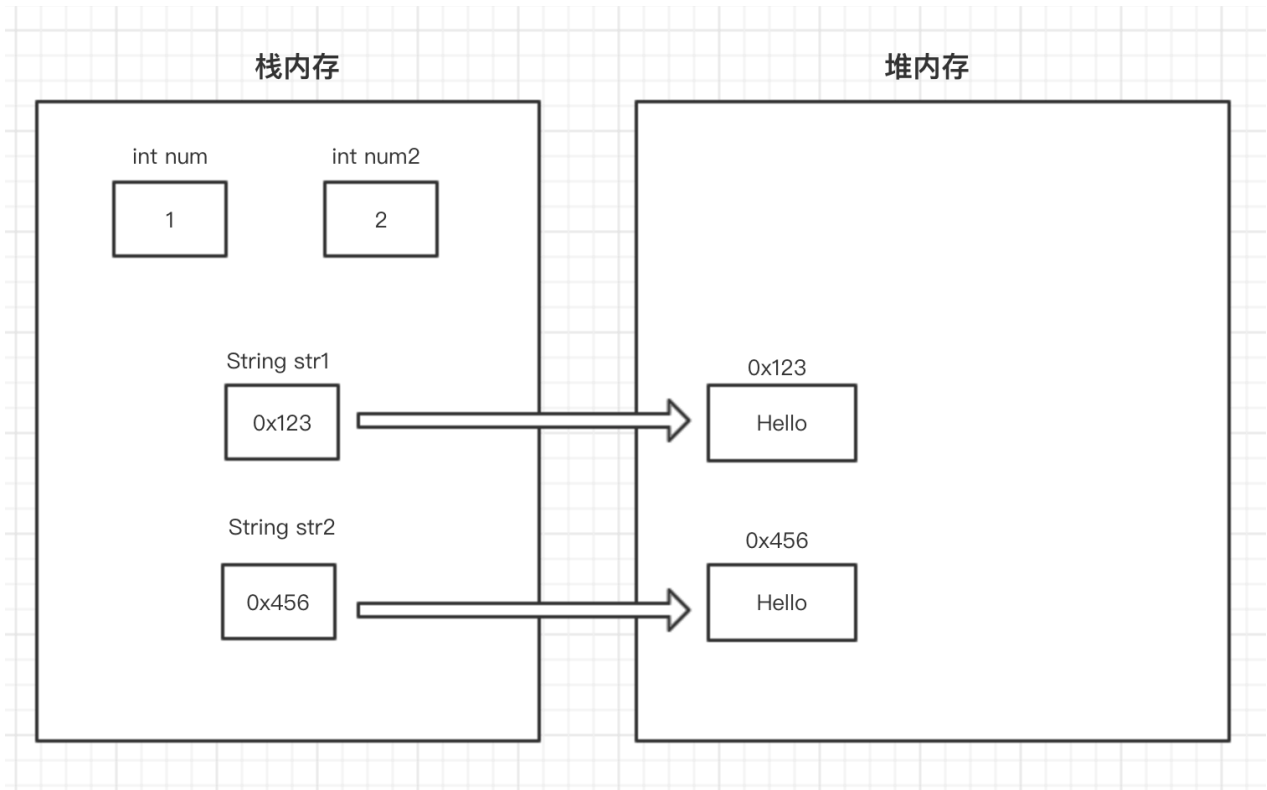
```
/Library/Java/JavaVirtualMachines
Hello
World
Java
String
```

经典面试题

1、== 和 equals 的区别？

== 可以理解为是比较栈内存中的值，如果变量是基本数据类型，则栈内存中存放的就是具体数值，如果是引用类型，则栈中存放的是引用的内存地址。

所以对于基本数据类型，== 是比较值是否相等，对于引用数据类型，比较的是引用的内存地址是否相等。



`equals` 是 `Object` 类提供的一个方法，其本质就是在用 `==` 进行判断。

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

但是 Java 中任意一个类都可以对其进行重写，根据具体需求重新定义其判断逻辑，比如我们自定义一个 `Student` 类，如下所示。

```
public class Student {  
    private Integer id;  
    private String name;  
    public Student(Integer id, String name) {  
        this.id = id;  
        this.name = name;  
    }  
}
```

创建两个成员变量值完全相等的实例化对象，并用 `equals` 方法判断是否相等。

```
Student student1 = new Student(1, "张三");  
Student student2 = new Student(1, "张三");  
System.out.println(student1.equals(student2));
```

结果为 false，因为两个实例化对象必然会在堆内存中开辟两块空间来存储，所以内存地址一定是相同的。而在现实的逻辑中，如果两个学生的 id 和 name 都一样，那么我们就认为他们是同一个学生，用程序如何实现呢？通过重写 equals 方法即可，如下所示。

```
public class Student {
    private Integer id;
    private String name;

    public Student(Integer id, String name) {
        this.id = id;
        this.name = name;
    }

    @Override
    public boolean equals(Object obj) {
        Student student = (Student) obj;
        if(id.equals(student.id) && name.equals(student.name)){
            return true;
        }
        return false;
    }
}
```

再次运行代码，返回值为 true。

2、下述代码的运行结果是？

```
String str1 = "Hello World";
String str2 = "Hello"+" World";
System.out.println(str1 == str2);
```

true, "Hello" 和 " World" 都是字符串面值，字符串面值 + 字符串字面值的结果仍然保存在字符串常量池中，所以 str1 和 str2 相同。

3、下述代码的运行结果是？

```
String str1 = "Hello World";
String str2 = "Hello";
str2 += " World";
System.out.println(str1 == str2);
```

false，这题看似与第 2 题一样，为什么结果完全不同呢？因为 str2 = "Hello"+" World" 是直接创建，str2 = "Hello"; str2 = "Hello"; 是先创建再修改，同时修改完成之后的字符串是放在堆内存中的，为什么呢？因为 str2 是一个字符串变量，" World" 是字符串字面值，当字符串字面值与 String 类型变量拼接时，得到的新字符串不再保存在常量池中，而是在堆中开辟一块新的空间来存储。

4、下述代码的运行结果是？

```
String str1 = "Hello World";
String str2 = " World";
String str3 = "Hello"+str2;
System.out.println(str1 == str3);
```

false, str2 是变量, "Hello" 是字符串面值, 字符串面值 + 变量会在堆内存中开辟新的空间来存储, 所以 str1 和 str3 不同。

5、下述代码的运行结果是？

```
String str1 = "Hello World";
final String str2 = " World";
String str3 = "Hello"+str2;
System.out.println(str1 == str3);
```

true, "Hello" 是字符串面值, str2 是常量, 字符串面值+常量的结果仍然保存在字符串常量池中, 所以 str1 和 str3 相同。

6、下述代码的运行结果是？

```
String str1 = "Hello World";
final String str2 = new String(" World");
String str3 = "Hello"+str2;
System.out.println(str1 == str3);
```

false, str2 是常量, 但是 new String(" World") 保存在堆内存中, 所以即使使用 final 进行了修饰, str2 仍然保存在堆中, 则 str3 也就保存在堆中, 所以 str1 和 str3 不同。

7、下述代码的运行结果是？

```
String str1 = "Hello World";
String str2 = "Hello";
String str3 = " World";
String str4 = str2 + str3;
System.out.println(str4 == str1);
System.out.println(str4.intern() == str1);
```

true, 当调用 str4 的 intern 方法时, 如果字符串常量池已经包含一个等于 str4 的字符串, 则返回该字符串, 否则将 str4 添加到字符串常量池中, 并返回其引用, 所以 str4.intern() 与 str1 相同。

8、什么是字符串常量池？

字符串常量池位于堆内存中，专门用来存储字符串常量，可以提高内存的使用率，避免开辟多块空间存储相同的字符串，在创建字符串时 JVM 会首先检查字符串常量池，如果该字符串已经存在池中，则返回它的引用，如果不存在，则实例化一个字符串放到池中，并返回其引用。

9、String 是线程安全的吗？

String 是不可变类，一旦创建了String对象，我们就无法改变它的值。因此，它是线程安全的，同一个字符串实例可以被多个线程共享，保证了多线程的安全性。

10、在使用 HashMap 的时候，用 String 做 key 有什么好处？

HashMap 内部实现是通过 key 的 hashCode 来确定 value 的存储位置，而因为字符串是不可变的，当创建字符串时，它的 hashCode 被缓存下来，不需要再次计算，所以相比于其他对象更快。