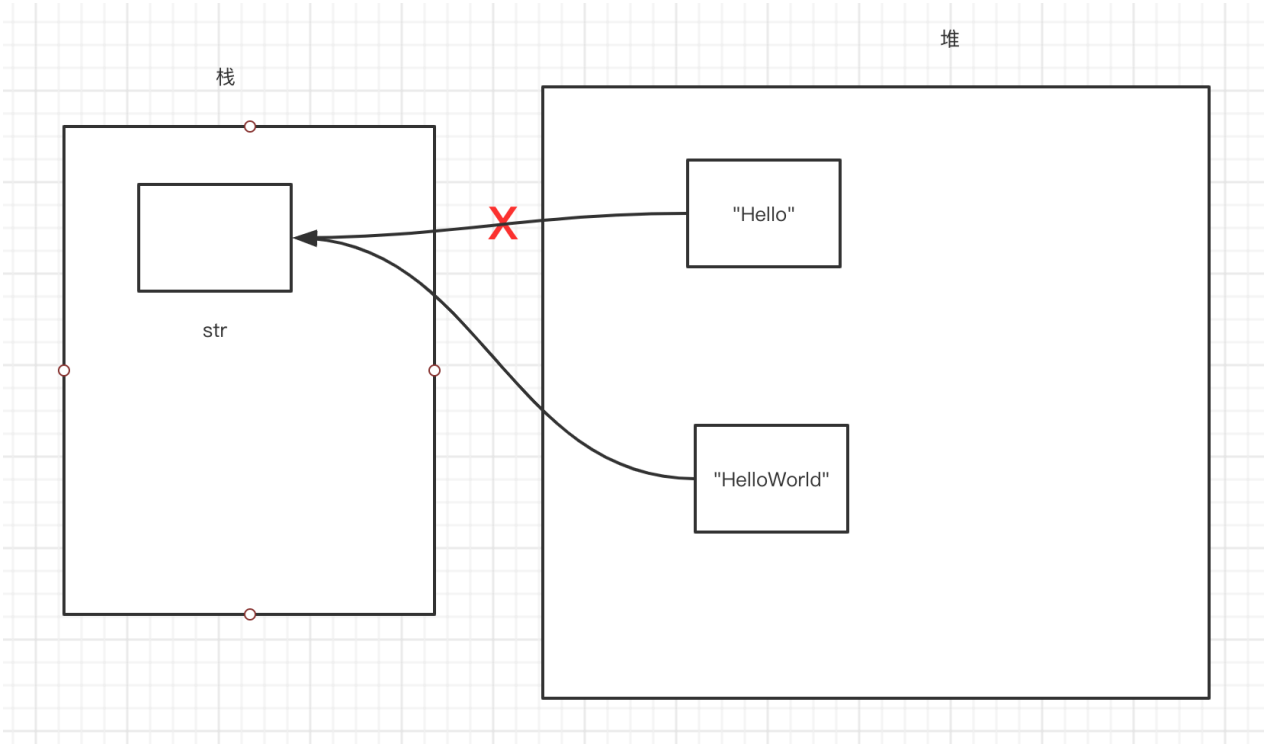
StringBuffer

String 对象一旦创建，值不能修改（原来的值不能修改，一旦修改就是一个新的对象，只要一改动，就会创建一个新的对象）

修改之后会重新开辟内存空间来存储新的对象，会修改 String 的引用。



String 的值为什么不能修改？修改之后会创建一个新的对象？而不是在原有对象的基础上进行修改？

因为 String 底层是用数组来存值的，数组长度一旦创建就不可修改，所以导致上述问题。

StringBuffer 可以解决 String 频繁修改造成的空间资源浪费的问题。

StringBuffer 底层也是使用数组来存值。

- StringBuffer 数组的默认长度为 16，使用无参构造函数来创建对象。

```
*/
@HotSpotIntrinsicCandidate
public StringBuffer() {
    super( capacity: 16);
}
```

- 使用有参构造创建对象，数组长度=值的长度+16。

```
@HotSpotIntrinsicCandidate
public StringBuffer(String str) {
    super( capacity: str.length() + 16);
    append(str);
}
```

```
package com.southwind.demo;

public class Test {
    public static void main(String[] args) {
        StringBuffer stringBuffer = new
StringBuffer("Hello");
        StringBuffer stringBuffer1 = new
StringBuffer();
        //stringBuffer 底层数组的长度是 21
        //stringBuffer1 底层数组的长度是 16
        stringBuffer1.append("Hello");

 System.out.println(stringBuffer.toString().equals(s
tringBuffer1.toString()));
        System.out.println(stringBuffer.length());
        System.out.println(stringBuffer1.length());
```

```
    }
}
```

length 方法返回的并不是底层数组的长度，而是它的有效长度（值的长度）

StringBuffer 一旦创建，默认会有 16 个字节的空间去修改，但是一旦追加的字符串长度超过 16，如何处理?

StringBuffer 不会重新开辟一块新的内存区域，而是在原有的基础上进行扩容，通过调用父类 ensureCapacityInternal() 方法对底层数组进行扩容，保持引用不变。

```
private void ensureCapacityInternal(int minimumCapacity) {
    // overflow-conscious code
    int oldCapacity = value.length >> coder;
    if (minimumCapacity - oldCapacity > 0) {
        value = Arrays.copyOf(value,
                newLength: newCapacity(minimumCapacity) << coder);
    }
}
```

StringBuffer 的常用方法，StringBuffer 是线程安全的，但是效率较低，StringBuilder 是线程不安全的，但是效率较高。

HashMap：线程不安全，效率高

Hashtable：线程安全，效率低

方法	描述
public StringBuffer()	创建一个空的 StringBuffer对象
public StringBuffer(String str)	创建一个值为 str 的 StringBuffer 对象

public synchronized int length()	返回 StringBuffer 的长度
public synchronized char charAt(int index)	返回指定位置的字符
public synchronized StringBuffer append(String str)	追加内容
public synchronized StringBuffer delete(int start,int end)	删除指定区间的值
public synchronized StringBuffer deleteCharAt(int index)	删除指定位置的字符
public synchronized StringBuffer replace(int start,int end,String str)	将指定区间的值替换成 str
public synchronized String substring(int start)	截取字符串从指定位置到结尾
public synchronized String substring(int start,int end)	截取字符串从start开始，到end结束
public synchronized StringBuffer insert(int offset,String str)	在指定位置插入 str
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fromIndex开始查找指定字符的位置
public synchronized StringBuffer reverse()	进行反转

public synchronized String toString()	转为 String

读取数据不需要考虑线程安全问题，因为这种操作不存在安全隐患。