基于Jdk1.7

## String[final]

### 成员变量

|  |
| --- |
| private final char value[];//使用一个final的char[]保存数据  private int hash;//此字符串的hashcode  //String底层基于char[],不可变 |

### 构造方法

|  |
| --- |
| public String(String original) {  this.value = original.value;  this.hash = original.hash;  }  //其中的一个构造方法。可以看出就是对char[]和hash的操作过程 |

### 重要方法：

#### equals (Object anObject);

//在尾部添加一个元素。

|  |
| --- |
| public boolean equals(Object anObject) {  if (this == anObject) {//同一个对象  return true;  }  if (anObject instanceof String) {//String的实例  String anotherString = (String) anObject;  int n = value.length;  if (n == anotherString.value.length) {//先比较长度  char v1[] = value;  char v2[] = anotherString.value;  int i = 0;  while (n-- != 0) {//依次比较每个字符  if (v1[i] != v2[i])  return false;  i++;  }  return true;  }  }  return false;  } |

#### Intern()

|  |
| --- |
| public native String intern();  //如果常量池中存在当前字符串, 就会直接返回当前字符串. 如果常量池中没有此字符串, 会将此字符串放入常量池中后, 再返回  详细解释：http://www.wtoutiao.com/a/1023451.html |

### String整体总结：

|  |
| --- |
| 底层基于char[]；  在早期的JVM实现版本中，被final修饰的方法会被转为内嵌调用以提升执行效率。而从Java SE5/6开始，就渐渐摈弃这种方式了。因此在现在的Java SE版本中，不需要考虑用final去提升方法调用效率。只有在确定不想让该方法被覆盖时，才将方法设置为final。  对于”i”+”love”+”you”,在编译期间，jvm对其进行了优化，此时就可以确定他的值为iloveyou,效率很高；  对于变量相加如i+j,jvm没有对其进行优化；  String\StringBuilder\StringBuffer之间的区别？  StringBuilder是线程不安全的；StringBuffer线程安全； |