## final 关键字

### 1）final修饰的变量不可重新赋值

例如 下面就会报错

final String test=“abc”;

test=”BCD”;

### 2）final修饰的方法不可被重写

class PersonalLoan{

public final String getName(){

return "personal loan";

}

}

class CheapPersonalLoan extends PersonalLoan{

@Override

public final String getName(){

return "cheap personal loan"; //compilation error: overridden method is final

}

}

### 3）final修饰的类不能被继承

String 类就是fianl的，所以 String是不能被别人继承的。

## String 特性：字符串常量池

每当我们创建一个字符串对象时，首先就会检查字符串池中是否存在面值相等的字符串，如果有，则不再创建，直接放回字符串池中对该对象的引用，若没有则创建然后放入到字符串池中并且返回新建对象的引用。

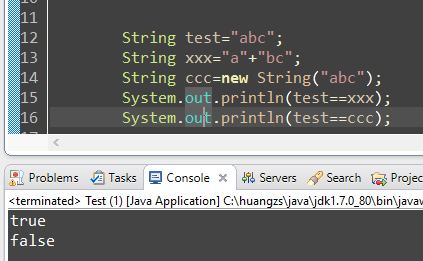
### 字符串String是特殊对象，如果new的话一定会创建一个新的对象（无论字符串池中是否有该字符串），String 对象重新指向一个字符串池中没有的字符串时，也会创建一个新的对象。如果指向一个字符串池中有的字符串时，则不会重新创建一个对象。这样在大量String对象有重复值时，可以节约内存。

指向同一个内存对象，也就是字符串池中的 abc对象

String test="abc";

String xxx="abc";

String ccc="abc";



## StringBuffer

### 一般字符串的频繁拼接就用StringBuffer。

StringBuffer和String一样都是用来存储字符串的，只不过由于他们内部的实现方式不同，导致他们所使用的范围不同，对于StringBuffer而言，他在处理字符串时，若是对其进行修改操作，它并不会产生一个新的字符串对象，所以说在内存使用方面它是优于String的。其实在使用方法，StringBuffer的许多方法和String类都差不多，所表示的功能几乎一模一样，只不过在修改时StringBuffer都是修改自身，而String类则是产生一个新的对象，这是他们之间最大的区别。

同时StringBuffer是不能使用=进行初始化的，它必须要产生StringBuffer实例，也就是说你必须通过它的构造方法进行初始化。

在StringBuffer的使用方面，它更加侧重于对字符串的变化，例如追加、修改、删除，相对应的方法：

## subString() 方法

这个方法返回的不是一个全新的对象，只是通过偏移量来对父字符串的截取引用。内存的额外消耗是40.

