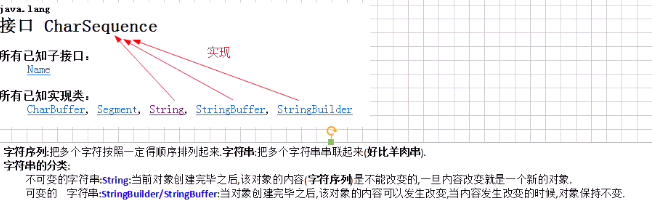
# Day 20

## 01字符串相关的类

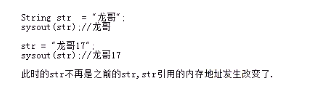


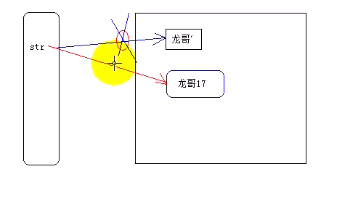
char表示一个字符，一个字符可能由多个字节编码成。

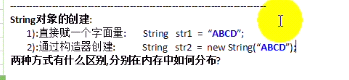
String其实就是对char数组的封装

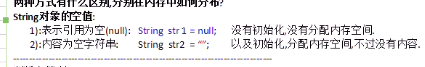
## 02深入string类分析

String对象：不可变的对象，一旦值改变就是一个新的对象，即引用的对象发生了改变

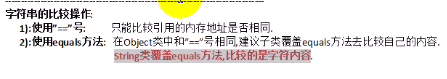


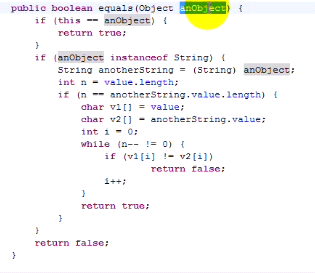










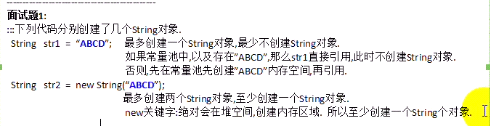


## 03string面试题分析

常量池：编译常量池和运行常量池，都存在于方法区中。

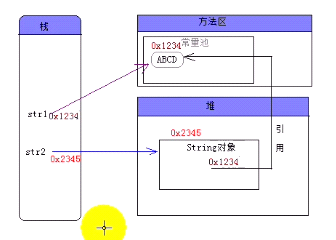
编译常量池：把字节码加载进jvm的时候，存储的是字节码的相关信息（一般不做研究）

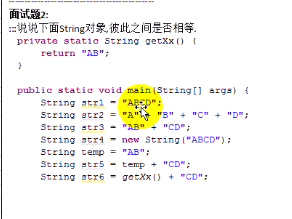
运行常量池：存储常量数据（研究）



String str=new String(“ABCD”);//现在内存堆中创建一个对象，并将该对象的引用赋值给str；

如果常量池中已经存在了字符串ABCD则内存堆直接存储该常量池的引用，否则先在常量池中创建字符串ABCD,然后将引用存储到内存堆中。

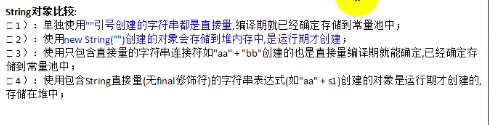




编译器优化

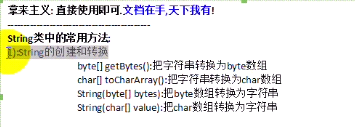
编译器优化会将str=”A”+”B”+”C”编译为就是str=”ABC”,可以查看反编译结果

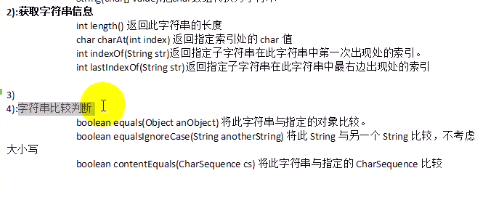
**编译时期不能确定变量值为多少所以不能优化，只有在运行时期才能确定；所以编译只会检查语法**





## 04string中的常用方法





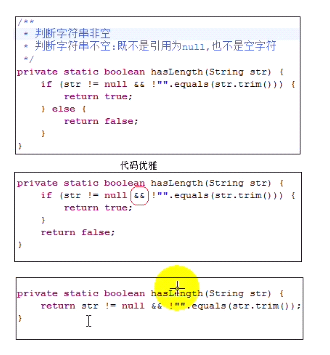


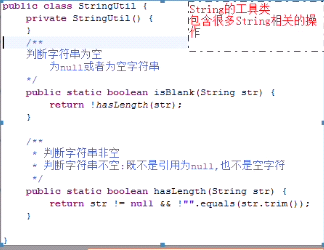
## 05常用类string的操作练习

## 06string的工具类封装



将常量放在前面可以防止str为null，可以不用判断str是否为null了



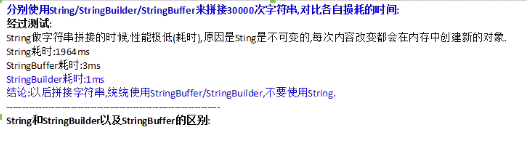


工具类：静态调用就私有化构造器；不希望被实例化就将构造器私有化。

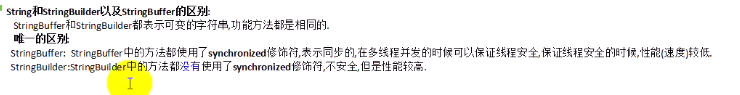
不希望被继承的类就使用final修饰

由于空间局部性的存在，所以对于二维数组来说，先对行遍历比先对列遍历更快。

## 07string和stringBuilder以及stringBuffer的区别



String连接字符串耗时又耗资源

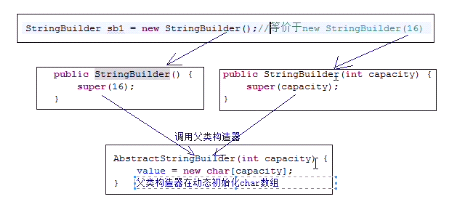


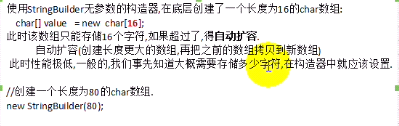
StringBuffer的方法保证是线程安全的，但是这样的话相对有StringBuilder来说性能更低，如果没有线程并发的场景的话建议使用StringBuilder；否则使用StringBuffer

## 08stringbuilder的常用操作



查看源代码







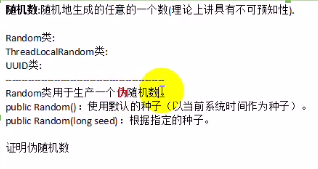
如果超过的话就自定扩容，类似Redis的重哈希。

因为StringBuilder对字符串的底层操作使用的是字符数组，因此如果调用无参构造器的话就会先初始化数组的长度为16，之后如果长度不够的话就会自动将容量翻倍，将旧的内容拷贝到新的数组，这会消耗性能。

当我们知道将要创建的数组的容量时，在初始化stringbuilder时指定容量能够提高性能。

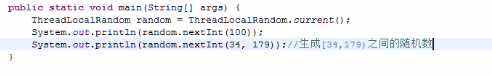
## 09随机数











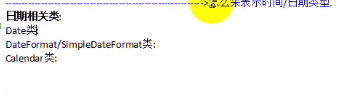




尽管UUID生产的数字保证了程序对数字唯一性的要求，但是它生成的唯一数字没有什么含义，也不能解析出含义。因此为了表示某中含义，一般我们自己来生成uuid。如Twitter的分布式数据id，mongodb的ObjectId（因为mongodb天生就是分布式的，所以它提供了一种这样的id-ObjectId）。

## 10生成验证码

## 11日期类

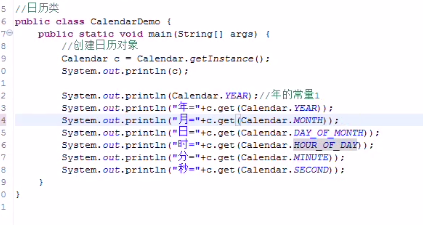


## 12日期格式化



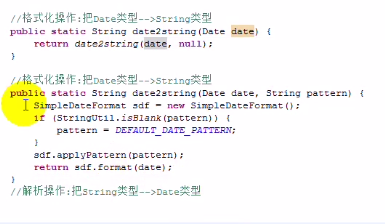


## 13日历类



抽象方法可以调用，因为运行时是子类实现的，多态

## 14实现日期的格式化和解析工作



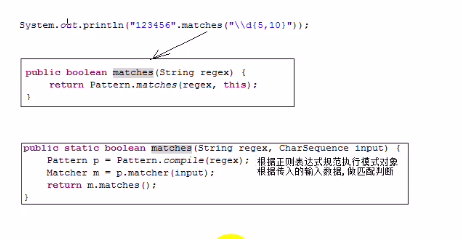
设置格式🡪返回格式化的结果

## 15日历操作练习

## 16正则表达式







## 17小结

