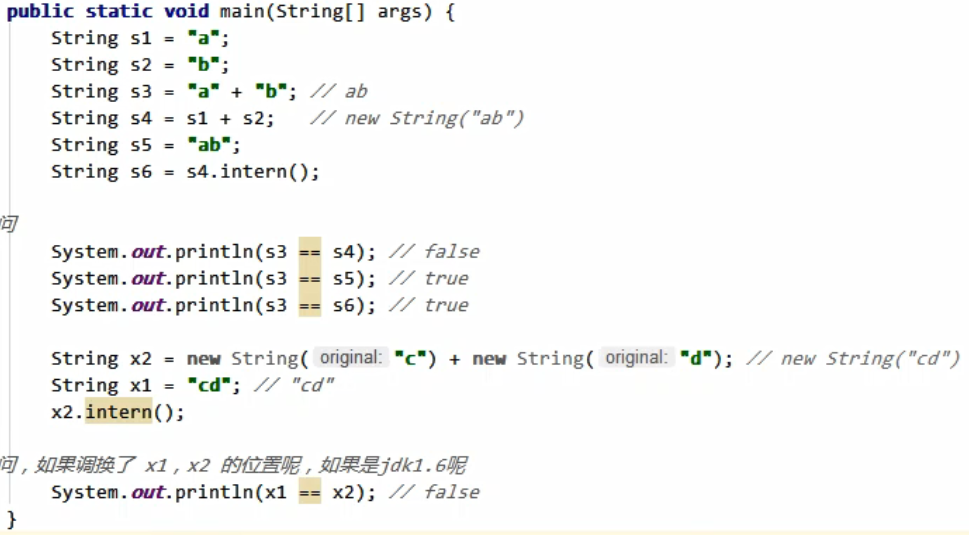


# 1.程序计数器（线程私有）

是当前线程所执行字节码的行号显示器，不会发生OOM·

# 5.字符串池（StringTable）



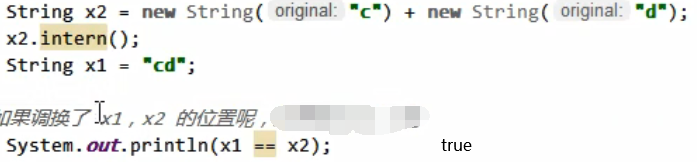
拼接两个字符串变量时，由于编译器默认用StringBuilder优化，所以return new String(…)，不放到常量池。

拼接两个字符串常量时，编译器默认将两个常量优化拼接

Tips：

String s1=new String(“a”)+ new String(“b”); //拼接仅在堆中创建一个”ab”的变量，不放入常量池。

String s2=new String(“a”); //这个会在堆和StringTable各放一份。



上面代码jdk1.8返回true，jdk1.6返回false

因为在jdk1.8中若常量池中不存在指定字符串，intern()时就将该字符串放入池子中

而jdk1.6中intern()是将该字符串复制一份后放入常量池。

### （1）StringTable的性能调优

#### a.若将很多个字符串放入jvm内存（将单词读入内存）

-XX:StringTableSize=1009 调整StringTable桶的个数（将桶的个数尽可能放大一点，避免hash冲突）

#### b.使用intern()入池

若项目有大量的字符串，而且字符串有一些会有重复，建议使用intern入池，可以节省堆内存使用（若是通过FileReader.readLine()进行读取的话，获得的字符串不会入池）

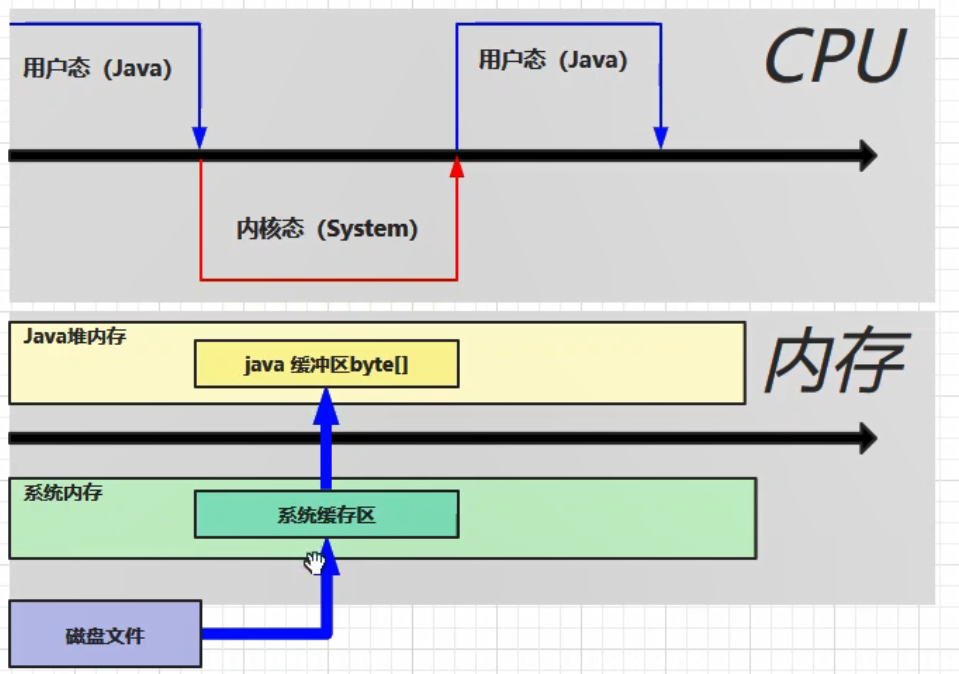
# 6.直接内存（p.42-43）

不属于java虚拟机的内存管理，属于操作系统内存

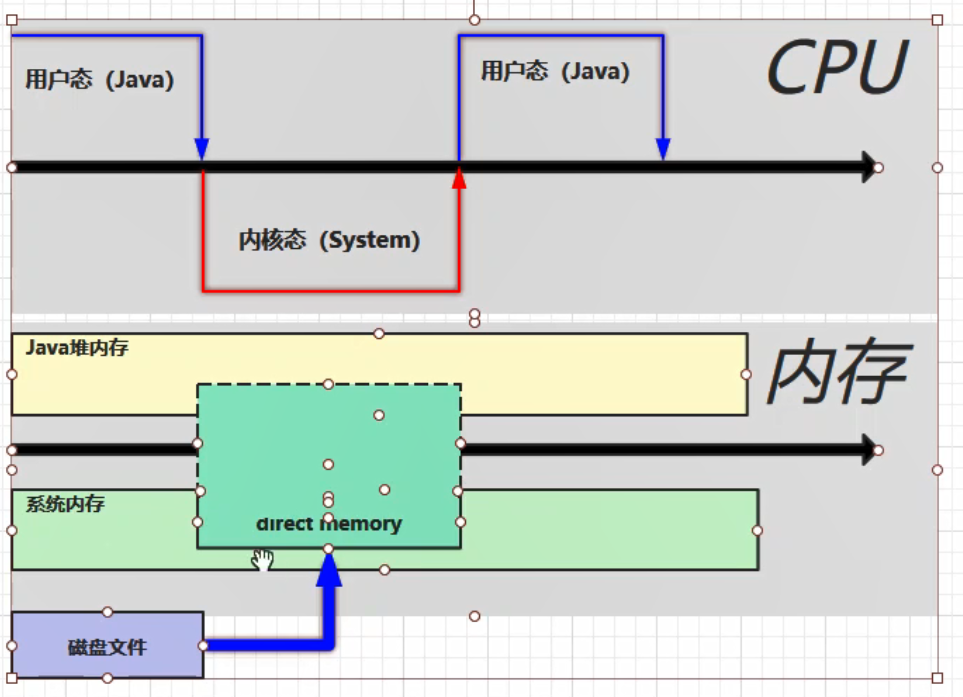
常见于NIO操作，用于数据缓冲区，分配和回收成本高，但读写性能高.

不受jvm内存回收管理

### （1）NIO比IO快的原因



io时，因为java代码不能读取系统缓冲区的数据，所以需要将磁盘文件读入两次缓冲区，所以效率不高。



用直接内存时，操作系统和java代码都可以读取直接内存中的数据，所以只要经过一次缓存。

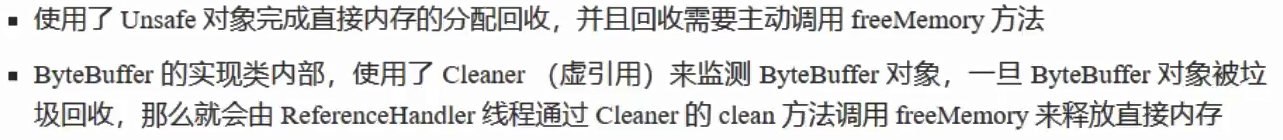
### （2）NIO内存如何查看占用与释放（p.45）

查看在java自带的工具是不能看的，需要通过任务管理器查看。

释放内存是使用了Unsafe类进行释放的



Cleaner是一个虚引用类。Cleaner类当关联的对象this被回收时，Cleaner会触发clean方法。（NIO释放使用了虚引用来释放）



### （3）禁用显示垃圾回收对NIO回收的影响

若禁止System.gc()会导致对ByteBuffer回收时，令ByteBuffer=null并System.gc()时导致不能及时回收直接内存

解决方法：

Unsafe方式手动释放

# 虚拟机参数

-XX:+PrintStringTableStatistics 打印StringTable中的个数和字符串大小

-XX:+PrintGCDetails -verbose:gc 打印GC详细信息

-XX:StringTableSize=1009 调整StringTable桶的个数

-XX:+DisableExplicitGC 禁用显示地垃圾回收（System.GC）因为它是Full GC