1.java是值传递还是引用传递

答：全部均为值传递!!!!

以前我认为基本类型是值传递,而对象是引用传递,这是错误的!!!!

有时候我会觉得传对象时里面的属性值变化了,

(瞎猜:)其实是属性值的地址变化了,值也就改变了,而原来的值还在,指向原来的地址,只是没人用而已(等待被回收)

来自〈https://blog.csdn.net/bjweimengshu/article/details/79799485〉

2.尾递归

有一种特殊的递归方式叫尾递归。如果函数中的递归调用都是尾调用,则该函数是尾递归函数。尾递归的特性使得递归调用不需要额外的空间 . 不能有任何操作, 如果有f(n)+1这类的操作,都不算尾递归(因为还是保存了函数的返回值)

> 来自:https://www.ibm.com/developerworks/cn/java/j-understanding-functional-programming-2/index.html?ca=drs-#icomments
> https://www.cnblogs.com/bellkosmos/p/5280619.html

```
// 实现斐波那契数列
// 普通递归
 int FibonacciRecursive(int n) {
    if( n < 2)
       return n;
    return FibonacciRecursive(n-1)+FibonacciRecursive(n-2);
 }

 // 尾递归
  int FibonacciTailRecursive(int n,int ret1,int ret2) {
    if(n==0)
```

```
        return ret1;
    return FibonacciTailRecursive(n-1,ret2,ret1+ret2);
}
```
来自: https://blog.csdn.net/mengxiangjia_linxi/article/details/78158819

尾递归效率高的原理：

尾递归就是从最后开始计算，每递归一次就算出相应的结果，也就是说，函数调用出现在调用者函数的尾部，因为是尾部，所以根本没有必要去保存任何局部变量. 直接让被调用的函数返回时越过调用者，返回到调用者的调用者去。**精髓：尾递归就是把当前的运算结果（或路径）放在参数里传给下层函数，也不用开辟新的栈空间,直接用上一个栈**

尾递归优化得益于编译器的支持,恰巧java**不支持尾递归优化**,但是上面那种省栈省开销还是有的(应该吧).一般函数式语言都是支持的,比如scala

但是可以利用lambda的懒加载来实现尾递归优化,
详情：https://www.cnblogs.com/invoker-/p/7723420.html#autoid-3-0-0

3. java中的CAS和AQS

CAS ： Conmpare And Swap (比较和交换) 是用于实现**多线程同步的原子指令。** 它将内存位置的内容与给(期望)定值进行比较，只有在相同的情况下，将该内存位置的内容修改为新的给定值。 这是作为单个原子操作完成的。
AQS： 抽象队列同步器(AbstractQueuedSynchronizer) ，是用来**构建锁或者其他同步组件的基础框架**，它使用一个int成员表示同步状态，通过内部的FIFO队列来完成资源获取线程的排序工作。ReentrantLock、Semaphore、CountDownLatch、CyclicBarrier等并发类均是基于AQS来实现的，

> https://blog.csdn.net/yanghan1222/article/details/80247844
> https://www.jianshu.com/p/0f876ead2846
> https://www.cnblogs.com/fsmly/p/11274572.html