快速排序

它的基本思想是：通过一趟排序将要排序的数据分割成独立的两部分，其中一部分的所有数据都比另外一部分的所有数据都要小，然后再按此方法对这两部分数据分别进行快速排序，整个排序过程可以递归进行，以此达到整个数据变成有序序列。

一趟快速排序的算法是：

1）设置两个变量i、j，排序开始的时候：i=0，j=N-1；

2）以第一个数组元素作为关键数据，赋值给***key***，即***key***=A[0]；

3）从j开始向前搜索，即由后开始向前搜索(j--)，找到第一个小于***key***的值A[j]，将A[j]和A[i]互换；

4）从i开始向后搜索，即由前开始向后搜索(i++)，找到第一个大于***key***的A[i]，将A[i]和A[j]互换；

5）重复第3、4步，直到i=j；(3,4步中，没找到符合条件的值，即3中A[j]不小于***key***,4中A[i]不大于***key***的时候改变j、i的值，使得j=j-1，i=i+1，直至找到为止。找到符合条件的值，进行交换的时候i， j指针位置不变。另外，i==j这一过程一定正好是i+或j-完成的时候，此时令循环结束）。

```
    void quikSort(int[] a,int start,int end) {
        int i=start;
        int j=end;
        int index=i;
        int t=0;
        boolean flag=true;  //从尾交换
        while(i!=j) {
            if(flag) {  //尾
                if(a[j]<a[index]) {
                    //交换
                    t=a[j];
                    a[j]=a[index];
                    a[index]=t;
                    flag=!flag;
                    index=j;
                }else {
                    j--;
                }
            }else {     //从头交换
                if(a[i]>a[index]) {
                    t=a[i];
                    a[i]=a[index];
                    a[index]=t;
                    flag=!flag;
                    index=i;
                }else {
                    i++;
                }
            }
            System.out.println(Arrays.toString(a));
        }
        if(start<i-1) {
            quikSort(a,start,i-1);
        }
        if(end>i+1) {
            quikSort(a,i+1,end);
```

```
        }

    }
```

单例

```
public class Singleton {
    public volatile static Singleton instance;  //volatile防止指令重排序，确保可见性；

    private Singleton(){};

    public static Singleton getInstance(){
        if(instance==null){    //check1
            synchronized (Singleton.class) {
                if(instance==null){   //check2
                    instance=new Singleton();
                }
            }
        }
        return instance;
    }
}
```

其中的关键在于，初始化和赋值操作是分开的。在多线程情况下，由于CPU考虑到提高自身利用率，会进行指令的重排。导致的结果是，可能会出现下面这种危险的情况： 第一步：现在堆上开辟空间

第二步：把刚刚开辟的地址空间赋值给instance栈变量

第三步：初始化对象。

假如在第二步的时候，另外的线程判断instance不为null，直接调用就会出现错误。