原理

程序的局部性原理是指程序在执行时呈现出局部性规律，即在一段时间内，整个程序的执行仅限于程序中的某一部分。相应地，执行所访问局部性原理又表现为 时间局部性 和 空间局部性

时间局部性

程序中的某条指令一旦执行，则不久之后该指令可能再次被执行；如果某数据被访问，则不久之后该数据可能再次被访问。强调数据的重复

空间局部性

指一旦程序访问了某个存储单元，则不久之后。其附近的存储单元也将被访问。强调连续空间数据的访问，一般顺序访问每个元素（步长为大，空间局部性越差

局部性代码测试

```java
public class Hello {

    public static void main(String[] args) {
        a1();
        a2();
    }

    public static void a1() {
        long start = System.currentTimeMillis();
         int sum = 0;
         int[][] a = new int[10000][10000];
         for (int i = 0; i < 10000; i++) {
             for (int j = 0; j < 10000; j++) {
                 a[i][j] = j;
                 sum += a[i][j];
             }
         }
         long end = System.currentTimeMillis();
         System.out.println("sum : " + sum + " , time : " + (end - start));
     }

     public static void a2() {
         long start = System.currentTimeMillis();
         int sum = 0;
         int[][] a = new int[10000][10000];
         for (int i = 0; i < 10000; i++) {
             for (int j = 0; j < 10000; j++) {
                 a[j][i] = j;
                 sum += a[i][j];
             }
         }
         long end = System.currentTimeMillis();
         System.out.println("sum : " + sum + " , time : " + (end - start));
     }
}
```

```
36
37 /*输出
38 sum : 1733793664 , time : 331
39 sum : -1674119088 , time : 1449
40 */
```

输出结果：可以看到同样循环这么多次，但是二位数组的循环方式不一样产生的结果差距非常大（正常情况如a1给二维数组赋值：先行后列多）

扩展与思考

上面代码是典型的空间局部性说明，步长固定，赋值也快很多，不用再内存地址上切来切去.

其他应用 比如 数据库 自然遍历，索引查找等。

时间局部性：刚拿到缓存的数据，没有被废弃掉，马上又拿到，减少交换，速度当然更快。

下面代码 那种速度更快?

```
1  public static void a1() {
2          for (int i = 0; i < 10000; i++){
3              //doSomeThing1();
4          }
5          for (int i = 0; i < 10000; i++){
6              //doSomeThing2();
7          }
8      }
9
10   public static void a2() {
11       for (int i = 0; i < 10000; i++){
12           //doSomeThing1();
13           //doSomeThing2();
14       }
15   }
```

写法一效率高，因为循环只执行了一遍，所以效率较好

写法二效率高，因为该写法中每个循环内的局部变量大部分情况下是比写法一少，这样更容易利用CPU的寄存器以及各级缓存，这满足局部性原理，所以效率较好

简单的程序验证了一下，发现确实有时候写法一执行时间较短，有时候写法二执行之间较短，没有明显的固定规律

即使是这样的结果，我还是倾向于使用第二种写法，这不是效率的原因，而是重构中，提倡一个循环只干一件事