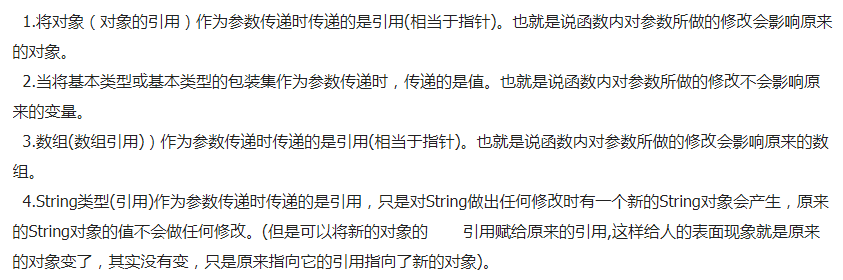
1. 基本数据类型是属于传值调用，当我们要实现交换时，方法内可以完成交换，只是实参进行了交换，方法结束，栈空间中就会销毁swap()方法内的局部变量，所以当调用swap()时，最终值还是实参的初始化值，并没有被修改。值改变作用到了形参，而不会到实参
2. 传引用调用：传引用调用中，如果传递的参数是引用数据类型，参数视为实参。在调用的过程中，将实参的地址传递给了形参，形参上的改变都发生在实参上
3. 基本类型包括： byte, short, int, long, char, float, double, boolean 这八大基本数据类型； 引用类型包括： 类类型， 接口类型和 数组。
4. （必须要new）定义数组 1e40098a23343d48ecef39b01daa489
5. 一维数组获取长度int len = nums.length;（没有括号，字符串是有的）
6. Vector

<https://www.runoob.com/java/java-vector-class.html>

1. 函数传值的情况



1. <https://www.runoob.com/java/java-string.html>
2. String的用法
   1. Int len = a.length();
   2. String类是final类，也即意味着String类不能被继承，并且它的成员方法都默认为final方法。在Java中，被final修饰的类是不允许被继承的，并且该类中的成员方法都默认为final方法。
   3. “对String对象的任何改变都不影响到原对象，相关的任何change操作都会生成新的对象”
   4. 在前面一篇讲解关于JVM内存机制的一篇博文中提到 ，在class文件中有一部分 来存储编译期间生成的 字面常量以及符号引用，这部分叫做class文件常量池，在运行期间对应着方法区的运行时常量池。
   5. 2是否多线程安全

String中的对象是不可变的，是常量，因此线程安全。

AbstractStringBuilder是StringBuilder与StringBuffer的公共父类，定义了一些字符串的基本操作，如expandCapacity、append、insert、indexOf等公共方法。StringBuffer对方法加了同步锁或者对调用的方法加了同步锁，因此它是线程安全的。StringBuilder并没有对方法进行加同步锁，因此它是线程不安全的。

StringBuilder叫**字符串缓存区**百**对象**,使用时度**可以不断添加字符、字符串等已有内容**。

需要使用时调用**toString()**获得对象里的所有字符串。

**作用：**

简化字符串常量池的个数，节省内存。

StringBuilder相当于知一个池子，可以存放很多已有的字符串，使用时可以以此获得很长的字符串，从而优化程序。

JDK底层使用StringBuilder实现字道符串拼接。

* 1. 那么有人会问既然有了StringBuilder类，为什么还需要StringBuffer类？查看源代码便一目了然，事实上，StringBuilder和StringBuffer类拥有的成员属性以及成员方法基本相同，区别是StringBuffer类的成员方法前面多了一个关键字：synchronized，不用多说，这个关键字是在多线程访问时起到安全保护作用的,也就是说StringBuffer是线程安全的

1. 关于List（List是一个接口，不能直接实例化.ArrayList是实现了List的一个类）
   1. List<String> res = new ArrayList();
   2. Res.add(int index , Object obj);
   3. Res.add(Object obj);

1.对于数组，想要知道长度，直接.length（是个属性）；数组长度是不可变的；

2.对于list map等容器，想要知道大小，.size() （是个方法）；长度是可变的；

总结：size大小大小听起来像是一个动态的词语，对应长度可变的list map;而length长度听起来则是一个固定的形容，对应长度不可变的数组。

1. HashSet
2. HashSet<Integer> s = new HashSet<>();
3. s.contains(nums[i])
4. s.add(nums[i]);
5. s.size();
6. S.remove(obj)
7. 常量
   1. Integer.MAX\_VALUE \\
   2. 注意溢出问题：
      1. 如果直接(long)(Integer.MAX + 1)的话，结果溢出为负数，因为Integer.MAX仍然是int类型，在int内+1就溢出为负数
      2. 要想保留而非溢出：(long)(Integer.MAX) + 1 ，这样先把int转为long，那么long类型+1后就看的是long的范围了。
      3. 溢出问题------记得联想到那一部分溢出，对小部分进行强转扩大范围
8. TreeSet
   1. 也是实现了abstractSet接口，也是set。可以用set的所有方法：contains(),add(),remove()
   2. 也是一棵二叉排序树
      1. Treeset.ceiling(a);找到在这棵树中，大于a的数中最小的一个数。如果没有，则返回一个空对象
      2. Treeset.floor(a)找到树中比a小的所有数中最大的一个数
9. 泛型不能是基本数据类型

