# Java数据结构与算法

## 数据结构和算法的概念

### 数据结构

数据结构是指数据在计算机存储空间中的安排方式。

一般大多数数据结构都应包含以下操作:

1.怎样插入一条数据.

2.寻找某一个数据项.

3.删除某一个特定的数据项.

### 算法

算法是指程序操作存储空间中的数据的过程。

### 重要性

只要是写程序就必定会涉及数据结构及算法。当然java类库给我们提供许多现成的数据结构工具，例如:HashMap、ArrayList 等。平时只要使用即可。但这些仅限于初级程序员，要想成为高级程序要,数据结构和算法是必会的。

## 数组

数组是应用最广泛的数据结构。它被植入到答大部分编程语言中。

### 大O表示法

在计算机中，我们有一种粗略度量算法的效率的表示法，大O表示法。这种表示法倾向于算法效率与数据项个数的联系。在大O表示法中 N、N/2、2N其实都一样统称为O(N)。

### 数学基础

在复习数组之前，我们先温习下需要用到的数学基础知识。

#### 对数

就像乘法的逆运算是除法一样，对数是求幂的逆运算，是指数函数的反函数。

定义:

如果https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D134/sign=2a228f561b950a7b71354ac73ed1625c/8644ebf81a4c510fbfd3367a6659252dd42aa5bf.jpg，即a的x次方等于N，那么数x就叫做以a为底N的对数，记做https://gss2.bdstatic.com/-fo3dSag_xI4khGkpoWK1HF6hhy/baike/s%3D72/sign=374d40bc0efa513d55aa6edc3c6d9940/e61190ef76c6a7ef68d97d46fbfaaf51f2de66d7.jpg。其中，a叫做对数的[底数](https://baike.baidu.com/item/%E5%BA%95%E6%95%B0" \t "_blank)，N叫做[真数](https://baike.baidu.com/item/%E7%9C%9F%E6%95%B0" \t "_blank)，x叫做“以a为底N的对数”。

#### 二分法

二分法（Bisection method） 即一分为二的方法. 设[a，b]为R的闭区间. 逐次二分法就是造出如下的区间序列([an，bn])：a0=a，b0=b，且对任一自然数n，[an+1，bn+1]或者等于[an，cn]，或者等于[cn，bn]，其中cn表示[an，bn]的中点.

典型算法:

1.算法：当数据量很大适宜采用该方法。采用二分法查找时，数据需是排好序的。

2.基本思想：假设数据是按升序排序的，对于给定值key，从序列的中间位置k开始比较，

3.如果当前位置arr[k]值等于key，则查找成功；

4.若key小于当前位置值arr[k]，则在数列的前半段中查找,arr[low,mid-1]；

5.若key大于当前位置值arr[k]，则在数列的后半段中继续查找arr[mid+1,high]，

6.直到找到为止,时间复杂度:O(log(n)) [3]  。

### 无序数组

Java中的数组属于对象数据类型，需要创建方可使用.

#### Java数组创建

创建一个固定长度为10 的int类型数组。

int[] intArr = new int[10];

显示创建一个int类型数组，并已给定值。

int[] intArray = {1,2,3,4,5}

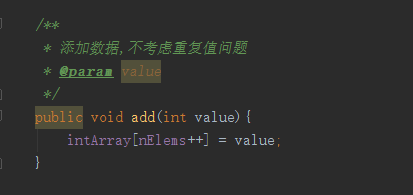
#### 封装一个无序数组



intArray用来保存数据

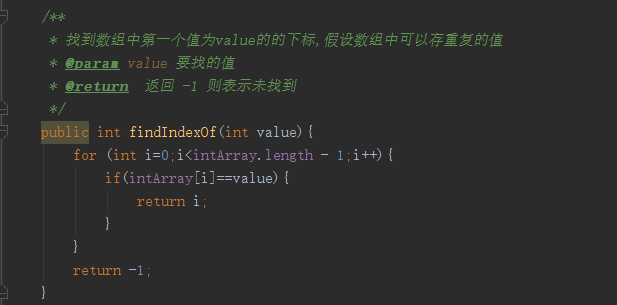
nElems属性记录下标到哪个位置,如当向该类添加一条数据时nElmes + 1

#### 无序数组的添加



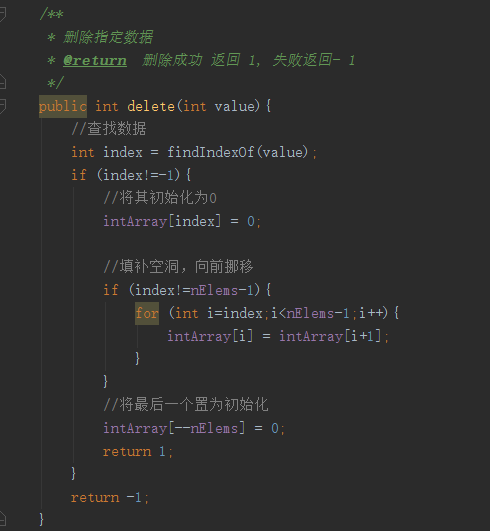
无序数组的添加非常简单，只需要向最后空位赋值即可。所以每次只需要操作一步(1)。如果数组中不允许重复的话，在添加前，还需要检测数组中的值。时间则是N/2 + 1 这里的1是常量，我们去除常量，时间则是N/2。

#### 无序数组的查找



我们进行线性查找，当匹配到值时返回该下标。分析下这种方式：运气好第一就是要找的值，不好则可能需要扫描到最后一个。假设我们每匹配一步需要1的时间，那么N步则需要N的时间，平均下来则需要 N/2的时间。

#### 无序数组的删除



删除前需要查找该数据,平均用时N/2,找到数据将其删除用时1。在数组里面有一个隐含的假设，即数组中不能含有空洞,因为这样会使其算法变得更加复杂，且效率更低，所以数组的数据必须连续存储。我们填补空洞时间为N/2,所以删除数据所用的时间最后为 N/2 + N/2 = N ，丢掉常数1.

#### 用大O表示法总结无序数组的效率

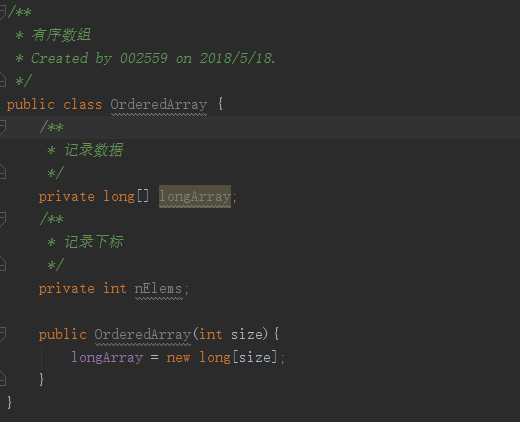
1.无序数组的插入 O(1)

2.无序数组的查找 O(N)

3.无序数组的删除 O(N)

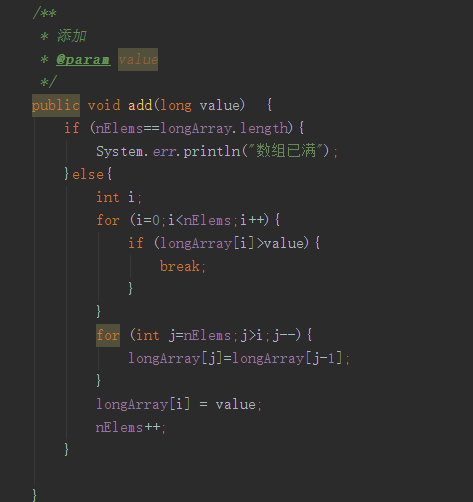
### 有序数组

#### 封装一个有序数组



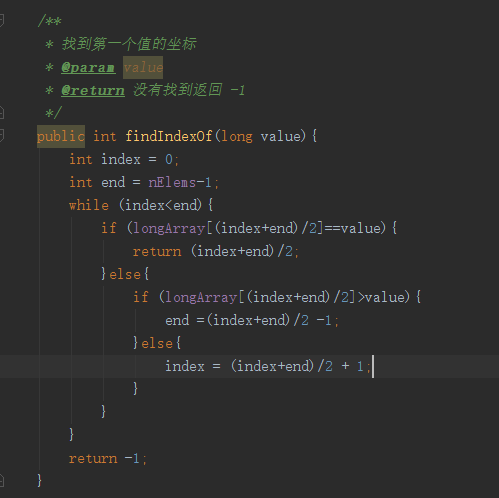
和无序数组一样

#### 有序数组的添加



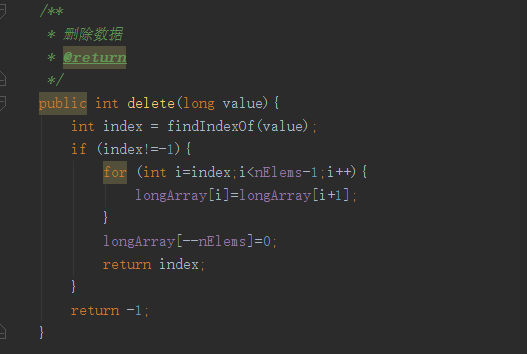
为了保持数组的有序性，添加时需要先找到最合适的插入位置，然后腾出该位置，将数据插入,时间复杂度=O(N)

#### 有序数组查找



有序数组可以使用二分法进行查找，效率大大提高了很多，特别是在数据量越大的时候 效率越明显.时间复杂度 = O(logN)

#### 有序数组删除



有序数组的删除，首先先查找到该数据，然后填补空洞

## 排序

在很多情况下我们需要对数据进行排序，例如在淘宝里面对商品的价格、信用进行排序，还比如对各省GDP总值进行排序等等。

### 简单排序

#### 冒泡排序

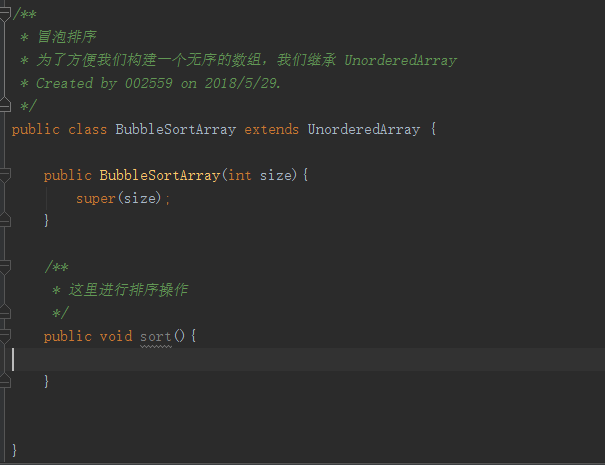
冒泡排序是一种非常简单的排序，也是非常慢排序

##### 核心思想

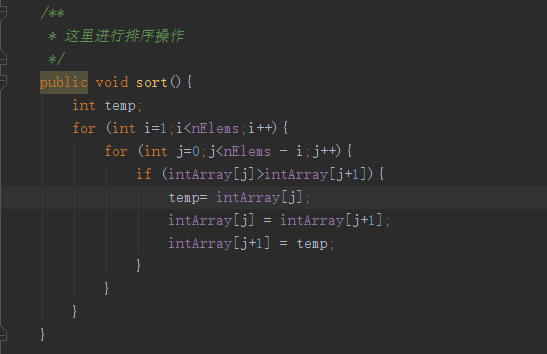
**比较两个元素的大小，如果前一个元素比后一个元素大则进行交换。每一次循环选出最大的元素，重复该操作，最后得到从小到大的数据**

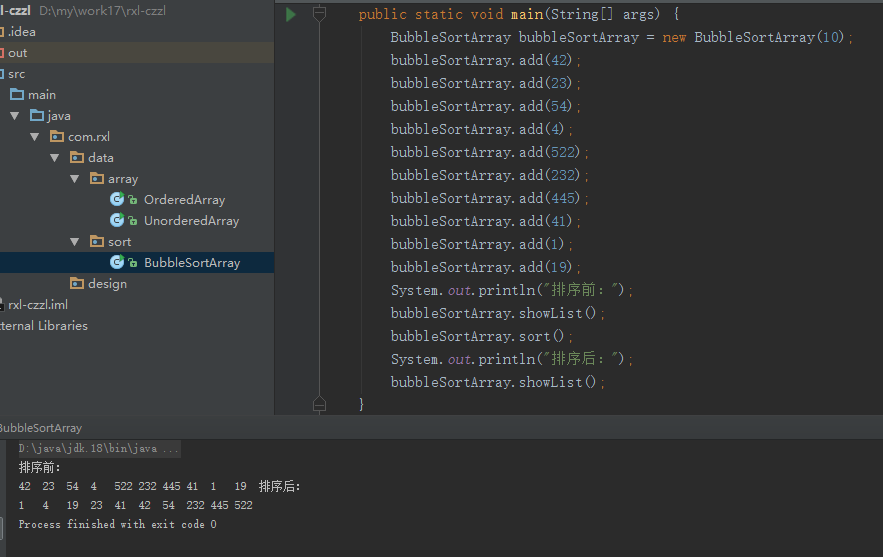
##### 代码实现

我们构建一个排序的环境



算法实现





#### 选择排序

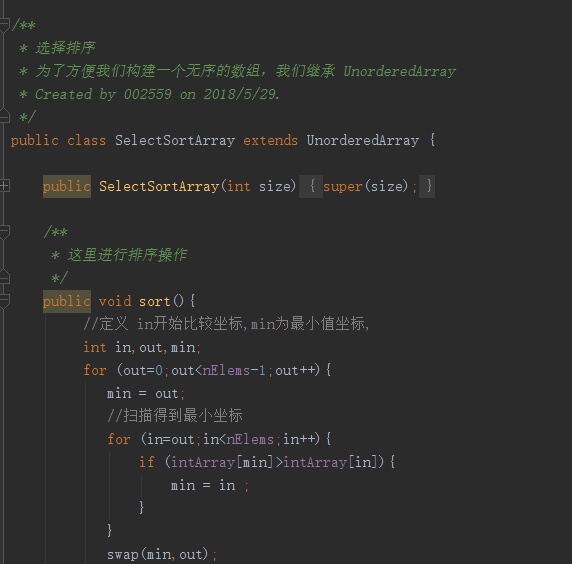
选择排序改进了冒泡排序，将必要的交换次数从O(N^2) 减少到O(N),不幸的是比较次数仍然是O(N^2)。然而在大量数据下，选择排序仍然是一个大的改进。

##### 核心思想

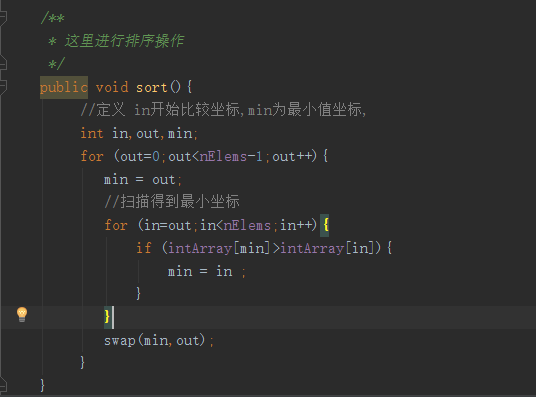
扫描所有的元素，比较得到最小的与元素，并将最小的元素与左边第一个进行交换。再扫描除第一位置的所有元素，比较得到最小元素，并将最小元素与左边第二个进行交换。以此类推。

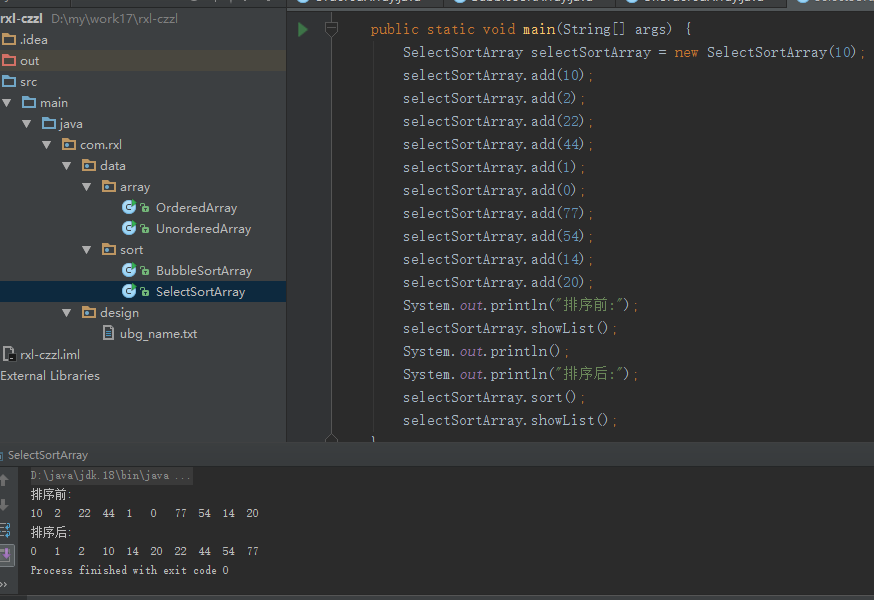
##### 代码实现

创建一个排序环境



算法实现





#### 插入排序

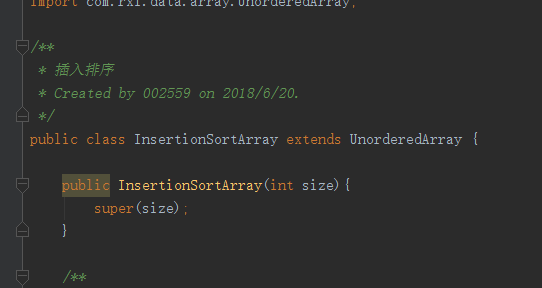
插入排序虽然时间复杂度仍然是O(N^2)，但在一般情况下比冒泡排序快一倍，比选择排序要快一些。经常被用到复杂排序算法的最后阶段。

##### 核心思想

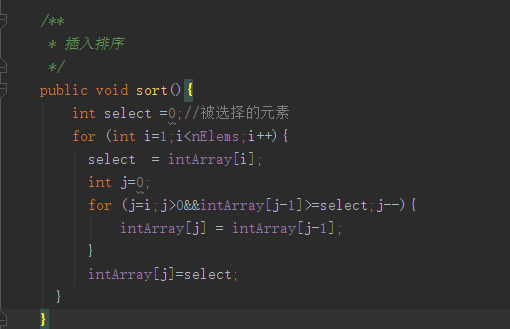
抽出一个元素,在前面的元素中找到合适的位置进行插入。

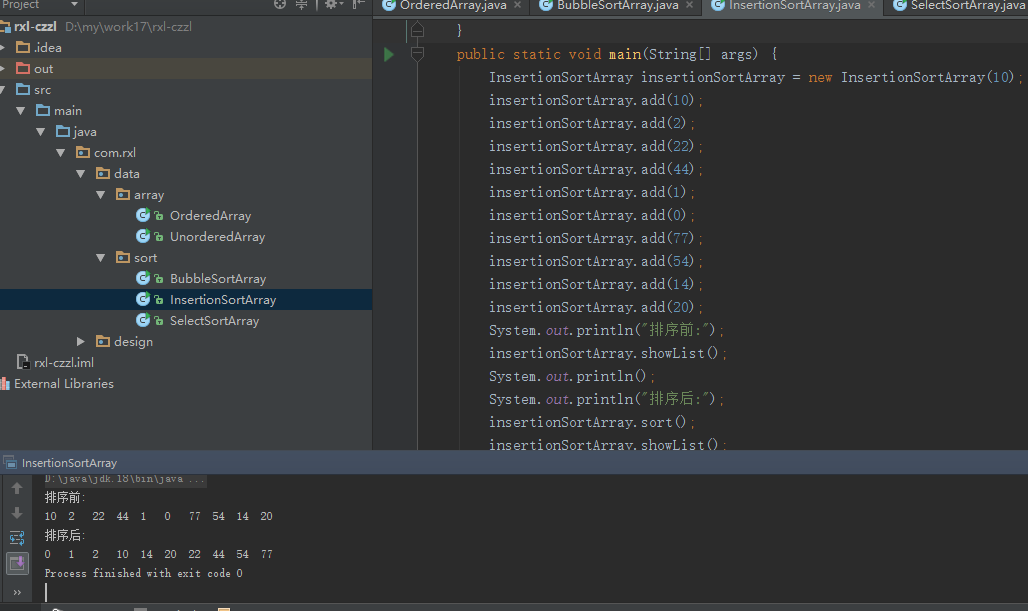
##### 代码实现

创建一个插入排序类



排序实现





## 栈和队列

## 链表

## 递归

## 二叉树

## 红-黑树

## 2-3-4树和外部存储

## 哈希表

## 堆

## 图

## 带权图

## 应用场景