* 今天学习目标

* 掌握冒泡排序

* 思路：从前到后两两比较，假如前比后大的话，交换位置

* 掌握选择排序

* 思路：从当前中选出最小的。（怎么样查找最小值的下标）

* 能够比较排序算法效率

* 冒泡排序，选择排序，系统提供排序

* 总结性能最好的是：系统提供排序>选择排序>冒泡排序

* 掌握二分查找

* 拆半查找

* 先排序好的

* from,to,mid

mid=（from+to）/2

* key

if(data[mid]==key){

return mid;

}else if(data[mid]>key){

* 左

to=mid-1;

mid=(from+to)/2;

} else if(data[mid]<key){

* from=mid+1;

mid=(from+to)/2;

}

* 找不到返回-1

* 什么情况下，才不找（终止循环）from>to

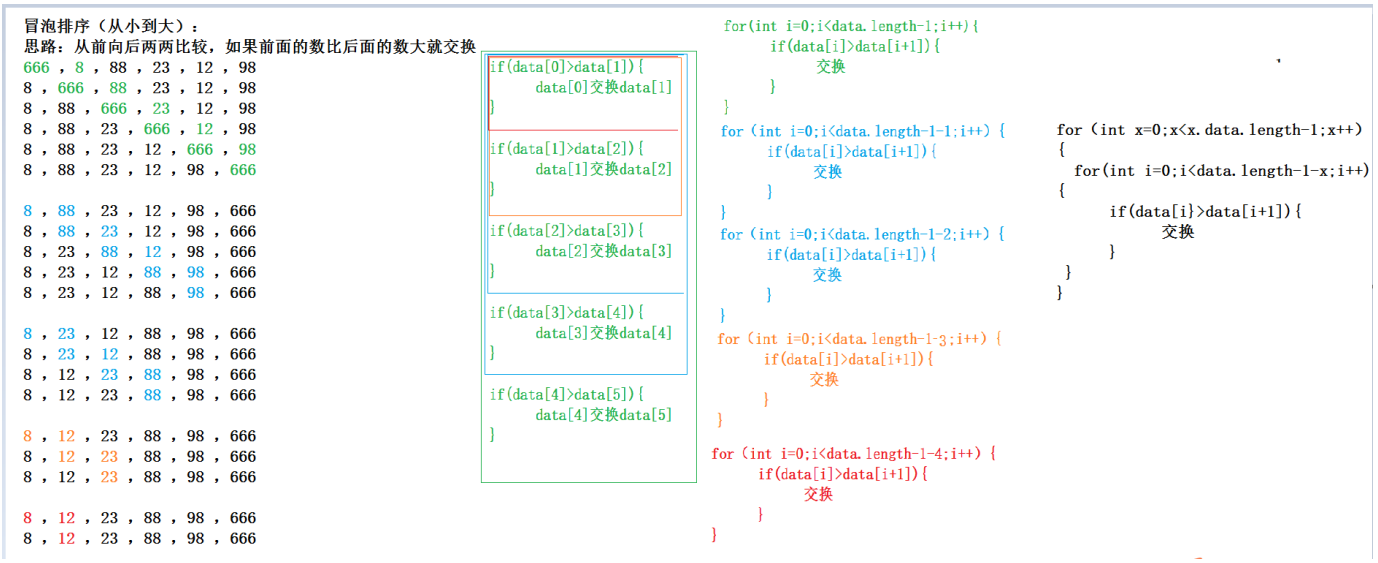
---

* 掌握冒泡排序

* 思路：从前向后两两比较，如果前面的数比后面的数大就交换

* 分析：

int [] data={666,8,88,23,12,98};



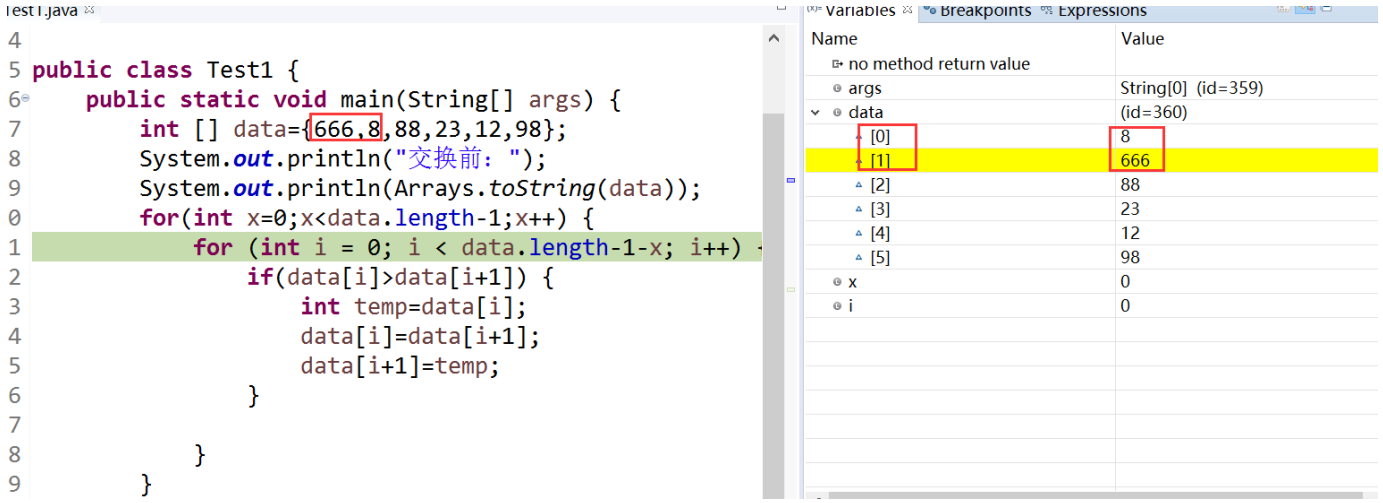
交换前:
[666, 8, 88, 23, 12, 98]
交换后:
[8, 12, 23, 88, 98, 666]

```
public static void main(String[] args) {
        int [] data={666,8,88,23,12,98};
        System.out.println("交换前: ");
        System.out.println(Arrays.toString(data));
        for(int x=0;x<data.length-1;x++) {
            for (int i = 0; i < data.length-1-x; i++) {
                if(data[i]>data[i+1]) {
```

```
                    int temp=data[i];
                    data[i]=data[i+1];
                    data[i+1]=temp;
                }

            }
        }
        System.out.println("交换后：");
        System.out.println(Arrays.toString(data));
    }
```

* 断点调试查看过程替换过程



* 掌握选择排序

  * 思路：从当前的数中找到最小的数，交换到指定位置

    * int [] data={666,8,88,23,12,98};

  * 分析：

选择排序（从小到大）
 * 思路：从当前的数中找到最小的数，交换到指定位置

666 , 8 , 88 , 23 , 12 , 98

666 , 8 , 88 , 23 , 12 , 98

8 , 666, 88 , 23 , 12 , 98

8 , 12, 88 , 23 , 666 , 98

8 , 12, 23 , 88 , 666 , 98

8 , 12, 23 , 88 , 666 , 98

8 , 12, 23 , 88 , 98 , 666

如何从当前的数中找到最小的数？下标

666 , 8 , 88 , 23 , 12 , 98

min

```
if(data[1]<data[min]){
    min=1;
}
if(data[2]<data[min]){
    min=2;
}
if(data[3]<data[min]){
    min=3;
}
if(data[4]<data[min]){
    min=4;
}
if(data[5]<data[min]){
    min=5;
}
```

把min元素换到0上

```
for (int i=min+1;i<data.length;i++) {
    if(data[i]<data[min]){
        min=i;
    }
}
```

把min元素换到0上

8 , 666, 88 , 23 , 12 , 98

min

```
if(data[2]<data[min]){
    min=2;
}
if(data[3]<data[min]){
    min=3;
}
if(data[4]<data[min]){
    min=4;
}
if(data[5]<data[min]){
    min=5;
}
```

把min元素换到1

```
for(int x=0;x<data.lengh-1;x++){
    min=x;
    for(int i=min+1;i<data.length;i++){
        if(data[i]<data[min]{
            min=i;
        }
    }
}
```

```
for (int i=min+2;i<data.length;i++) {
    if(data[i]<data[min]){
        min=i;
    }
}
```

把min元素换到1上

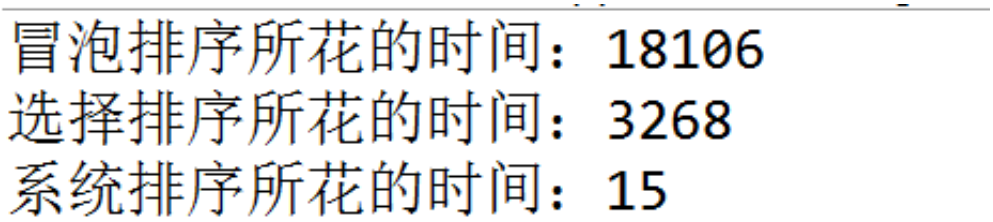
```
[666, 8, 88, 23, 12, 98]
--------------------------------
[8, 666, 88, 23, 12, 98]
--------------------------------
[8, 12, 88, 23, 666, 98]
--------------------------------
[8, 12, 23, 88, 666, 98]
--------------------------------
[8, 12, 23, 88, 98, 666]
--------------------------------
```

```
public static void main(String[] args) {
        int [] data={666,8,88,23,12,98};
        System.out.println(Arrays.toString(data));
        System.out.println("--------------------------------");
```

```
        for (int x = 0; x < data.length-1; x++) {
            int min=x;
            for (int i = min+1; i < data.length; i++) {
                if(data[i]<data[min]) {
                    min=i;
                }
            }
            int temp=data[x];
            data[x]=data[min];
            data[min]=temp;
            System.out.println(Arrays.toString(data));
            System.out.println("--------------------------------");
        }
    }
```

* 能够比较排序算法效率

    * 随记产生10万条数据，通过不同排序所花的时间




```
package com.lg.test;

import java.util.Arrays;
import java.util.Random;

public class Test3 {
    public static void main(String[] args) {
```

```
        Random r = new Random();
        int arrLength = 100000;
        int arr[] = new int[arrLength];
        for (int i = 0; i < arrLength; i++) {
            int rand = r.nextInt(arrLength);
            arr[i] = rand;
        }
        long start = System.currentTimeMillis();
        bubbleSort(arr);
        long end = System.currentTimeMillis();
        System.out.println("冒泡排序所花的时间："+(end - start));

        for (int i = 0; i < arrLength; i++) {
            int rand = r.nextInt(arrLength);
            arr[i] = rand;
        }
        start = System.currentTimeMillis();
        selectionSort(arr);
        end = System.currentTimeMillis();
        System.out.println("选择排序所花的时间："+(end - start));

        for (int i = 0; i < arrLength; i++) {
            int rand = r.nextInt(arrLength);
            arr[i] = rand;
        }
        start = System.currentTimeMillis();
        Arrays.sort(arr);
        end = System.currentTimeMillis();
        System.out.println("系统排序所花的时间："+(end - start));
    }

    public static void bubbleSort(int arr[]) {
        for (int x = 0; x < arr.length - 1; x++) {
            for (int i = 0; i < arr.length - 1 - x; i++) {
                if (arr[i] > arr[i + 1]) {
                    int temp = arr[i];
                    arr[i] = arr[i + 1];
                    arr[i + 1] = temp;
                }
            }
```

```
        }
    }

    public static void selectionSort(int arr[]) {
        for (int x = 0; x < arr.length - 1; x++) {
            int min = x;
            for (int i = min + 1; i < arr.length; i++) {
                if (arr[i] < arr[min]) {
                    min = i;
                }
            }
            int temp = arr[x];
            arr[x] = arr[min];
            arr[min] = temp;
        }
    }
}

```

* 掌握二分查找

* 前提数组中元素从小到大排序

* 例子：3 ，8 ，12 ，23 ，88 ，128 ，666

二分查找（折半查找）：

前提是数组从小到大排序

1) key=23

```
if(data[mid]==key){
     return mid;
}
```

2) key=12

```
from mid   to
 |   |    /
[3 , 8 , 12 , 23 , 88 , 128 , 666]
```

```
if(data[mid]==key){
     return mid;
}else if(data[mid]>key){
   to=mid-1;mid=(from+to)/2
} else {  from=mid+1;mid=(from+to)/2}
```

```
if(data[mid]==key){
     return mid;
}else if(data[mid]>key){
   to=mid-1;mid=(from+to)/2
} else {  from=mid+1;mid=(from+to)/2}
```

```
      mid
       |
from   /
      to
[3 , 8 , 12 , 23 , 88 , 128 , 666]
```

3)key=100

```
from        mid          to
 |           |           /
[3 , 8 , 12 , 23 , 88 , 128 , 666]
```

```
          mid
           |
          to
         /
      from
[3 , 8 , 12 , 23 , 88 , 128 , 666]
```

找不到就返回-1

```java
public static void main(String[] args) {
    int[] arr = { 3, 8, 12, 23, 88, 128, 666 };
    int index = binarySearch(arr, 23);
    System.out.println(index);
    System.out.println(binarySearch(arr, 12));
    System.out.println(binarySearch(arr, 128));
    System.out.println(binarySearch(arr, 100));
}

public static int binarySearch(int[] arr, int key) {
    int from = 0;
    int to = arr.length - 1;
    int mid = (to - from) / 2;
    while (from <= to) {
        if (arr[mid] == key) {
            return mid;
        } else if (arr[mid] > key) {
            to = mid - 1;
            mid = (from + to) / 2;
        } else {
            from = mid + 1;
            mid = (from + to) / 2;
        }
    }
    return -1;
}
```

3
2
5
-1