选择排序是一种简单的排序算法。此排序算法是一种基于就地比较的算法，其中，列表分为两部分，左端为已排序部分，右端为未排序部分。最初，已排序部分为空，未排序部分为整个列表。

从未排序的数组中选择最小的元素，并与最左边的元素交换，该元素成为排序数组的一部分。此过程继续将未排序的数组边界向右移动一个元素。

该算法不适用于大型数据集，因为其平均和最坏情况下的复杂度均为Ο(n^2)，其中n是项数。

## 核心思想

以下面描述的数组为例。



对于排序列表中的第一个位置，将顺序扫描整个列表。当前存储14的第一个位置，我们搜索整个列表，发现10是最低值。



因此，我们将10替换为14。一次迭代10（恰好是列表中的最小值）出现在已排序列表的第一位置。



对于第二个位置（33个位置），我们开始以线性方式扫描列表的其余部分。



我们发现14是列表中第二低的值，它应该出现在第二位。我们交换这些值。



经过两次迭代后，两个最小值以排序的方式位于开头。



以下是整个分类过程的图示-



## 代码开发

### 实现思路

Step 1 − 设置最小值到位置0  
Step 2 − 搜索列表中的最小元素  
Step 3 − 在MIN位置交换值  
Step 4 − 递增MIN以指向下一个元素  
Step 5 − 重复直到列表排序

### 伪代码

package com.paal.demo.c01SortingBasic;  
  
import com.paal.demo.Sort;  
import com.paal.demo.utils.SortTestHelper;  
  
/\*\*  
 \* <p/>  
 \* <li>title: 基础排序-选择排序</li>  
 \* <li>@author: li.pan</li>  
 \* <li>Date: 2019/11/4 12:43 下午</li>  
 \* <li>Version: V1.0</li>  
 \* <li>Description: </li>  
 \*/  
public class SelectionSort implements Sort {  
  
  
 @Override  
 public void sort(Integer[] arr) {  
  
 for (int i = 0; i < arr.length; i++) {  
 // 保存最小元素所在位置  
 int minIndex = i;  
 for (int j = i + 1; j < arr.length; j++) {  
 if (arr[j] < arr[minIndex]) {  
 minIndex = j;  
 }  
 }  
 // 自定义的用于数组元素交换的函数  
 SortTestHelper.swap(arr, i, minIndex);  
 }  
  
 }  
}  
  
 /\*\*  
 \* 选择排序测试  
 \*/  
 @Test  
 public void selectionSortTest() {  
 Integer[] integers0 = SortTestHelper.generateRandomArray(100, 0, 1000000);  
 Integer[] integers1 = SortTestHelper.generateRandomArray(10000, 0, 1000000);  
 Integer[] integers2 = SortTestHelper.generateRandomArray(100000, 0, 1000000);  
 System.out.println("------------------------------随机数组--------------------------------");  
 System.out.println("选择排序测试1"+SortTestHelper.testSort(integers0, new SelectionSort()));  
 System.out.println("选择排序测试2"+SortTestHelper.testSort(integers1, new SelectionSort()));  
 System.out.println("选择排序测试3"+SortTestHelper.testSort(integers2, new SelectionSort()));  
  
 }