# Selection Sort in Java

Selection Sort is a technique where a array is sequentially sorted by placing the smallest or the largest element from the array one after the other in multiple iterations. The time complexity of selection sort is more then the time complexity of insertion sort. Algorithm is not suitable for large data sets.

| | |
|---|---|
| Time Complexity (Best) | Ω(n2) |
| Time Complexity (Avg) | Θ(n2) |
| Time Complexity (Worst) | O(n2) |
| Space Complexity | O(1) |

## Implementation

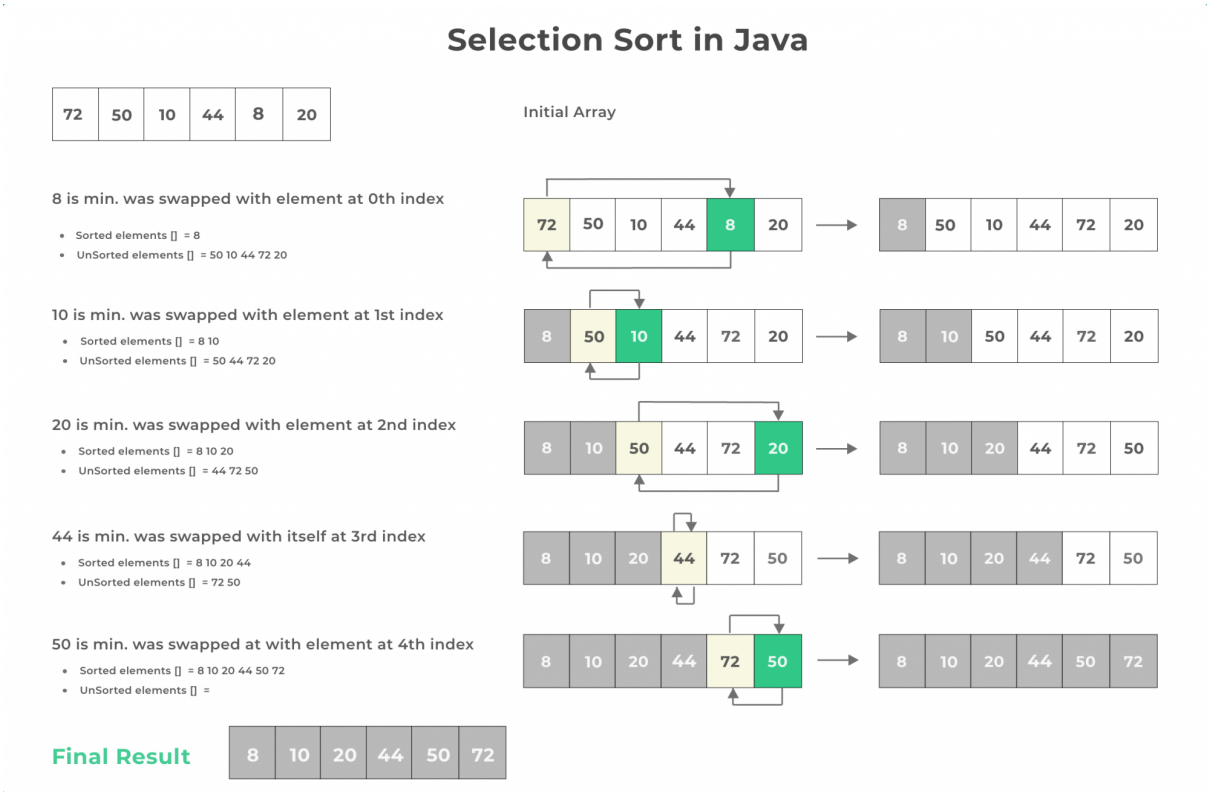
At any point during the selection sort implementation, we will have the array divided into a sorted array part on the left and an unsorted array part on the right.

- We find the smallest item in the unsorted side of the array
- We replace that with the first item in the unsorted array part
- This way we will have one more item added to the sorted part of the array on left
- One lesser item in the unsorted part of the array on the right

## Algorithm for selection sort

- sort(Arr[]):
- l= Arr.length
- FOR i from 1 to l-1:
- min = i
- FOR j = i+1 to l
- if Arr[j]<Arr[min]
- min = swap(Arr[j], Arr[min])
- END of FOR
- END of FOR

## Selection Sort in Java




```
package Sort;

public class Selection {
  static void selectionSort(int a[]) {
    int len = a.length;

    // One by one move boundary of unsorted sub-array
    for (int i = 0; i < len - 1; i++) {
      // Find the minimum element in unsorted array
      int min = i;
      for (int j = i + 1; j < len; j++)
        if (a[j] < a[min])
          min = j;

      // Swap the found minimum element with the
      // first element in unsorted part of the array
      int temp = a[min];
      a[min] = a[i];
      a[i] = temp;
    }
  }

  // Prints the sorted array
  static void printArray(int a[]) {
    int len = a.length;
    for (int i = 0; i < len; ++i)
      System.out.print(a[i] + " ");
```

```
    System.out.println();
  }

  // Main method, responsible for the execution of the code
    public static void main(String args[]) {
      int arr[] = { 72, 50, 10, 44, 20 };

      System.out.println("Array Before Sort:");
      printArray(arr);

      selectionSort(arr);

      System.out.println("Array After Sort:");
      printArray(arr);
    }
}
```

- Selection sort gives a time complexity of O(n2) in all cases regardless of arrangement of data
- It could be useful in cases when swapping operation are very costly to do this would help as number of swaps in selection sort are : size – 1