PS 2 - CMPE 160.01: Introduction to Object Oriented Programming

Arrays and Methods:

11/03/2022

1 Array Reverse:

You can pass arrays when invoking a method. A method may also return an array. For example, we can write a method that takes an array and returns another array that is the reversal of the given array.

We should be able to call the said method (call it 'reverse') like this:

```
int[] list1 = {1, 2, 3, 4, 5, 6};
int[] list2 = reverse(list1);
```

And list2 should contain the elements **6,5,4,3,2,1**.

One simple solution to create a new array at first, same size as our original array. After that, we will copy elements from the original array to new array, starting from last element and finishing with the first.

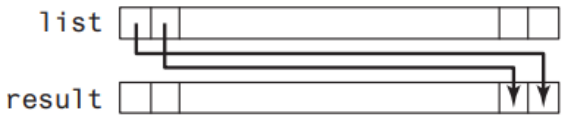

Figure 1: Copying elements from first array to second empty array

2 Selection Sort:

Suppose you want to sort a list in ascending order. Selection sort finds the smallest number in the list and swaps it with the first element. It then finds the smallest number remaining and swaps it with the second element, and so on, until only a single number remains. Suppose we have the list 2, 9, 5, 4, 8, 1, 6. You can see below how we can sort it using selection sort algorithm.

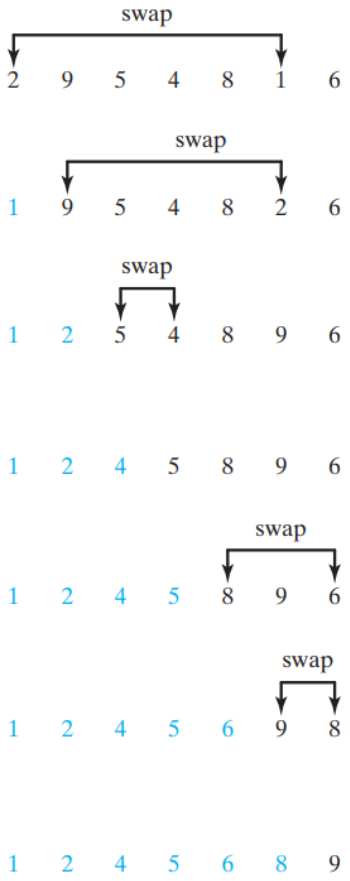

Figure 2: Selection sort algorithm

On the first line we can see that **1** is selected, being the smallest number on the array and swapped with the first element, becoming the first element. After that we continue doing that with **2** and **4**. When we look at the 4th line, we see that next smallest element is **5**, and its already in its spot (the next leftmost spot). Then with **6** and **8**, we continued to do the same and **9** is already in its place, being the last element left.

Tip: When you will implement the algorithm by yourself: Start by writing the code for the first iteration to find the smallest element in the list and swap it with the first element, then observe what would be different for the second iteration, the third, and so on. The insight this gives will enable you to write a loop that generalizes all the iterations.

Solution can be described as follows:

```
for (int i = 0; i < list.length  1; i++) {
    select the smallest element in list[i..list.length1];
    swap the smallest with list[i], if necessary;
    // list[i] is in its correct position.
    // The next iteration applies on list[i+1..list.length1]
}
```