Program for Insertion sort is

LP_CODE2.java

```
Enter the size of array
5
Enter the elements
10 40 30 20 50
Array after sorting
10 20 30 40 50
```

Time Complexity:
The time complexity of the following algorithm will be $O(n^2)$ as for every particular element we find its correct position and insert it at that particular position. This requires $O(n)$ time complexity for one element. We are doing this for every element so $O(n^2)$ time complexity in total

Space Complexity:
It does not require any extra space so space complexity is $O(1)$.

Is insertion sort stable?
Here we just pick an element and place it in its correct place and in the logic we are only swapping the elements if the element is larger than the key, i.e. we are not swapping the element with the key when it holds equality condition, so insertion sort is stable sort.

Selection Sort
This algorithm sorts an array by repeatedly finding the minimum element (considering ascending order) from the unsorted part and putting it at the beginning.

The algorithm maintains two subarrays in a given array.

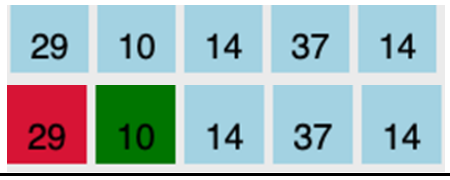
- The subarray which is already sorted.
- The remaining subarray was unsorted

Let's understand the selection sort for sorting the array in non decreasing order.

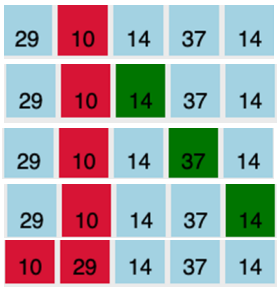
As the name suggests, in the Selection Sort algorithm we select the index having minimum value by traversing through the array and swap it with the current index, then increment the current index.
This way we keep on traversing the rest of the array and finding the minimum index and swap it with the current index. This will result in forming a sorted array on the left side.

- Iterate through the array and consider the current element index.
- Now traverse the rest of the array elements and find the minimum element index.
- Swap it with the current element index and increment the current element index by one.
- Repeat until we reach the end of the array.

Now, let's take an example to understand the working of selection sort with sorting the array in increasing order.



The current index is 0. We start from next index and find minimum element as compared to 29 if any.



Now 10 is the minimum element.
We start moving ahead to find element smaller than 10

No Element is smaller than 10, so swapping current index and index 1 (Index of 10)

Second Pass

Now the current index is 1 and the value is 29. We will find elements smaller than 29 if any.

14 is smaller than 29.

As 14 is the smallest element we swap current index and 2nd index (index of 14)

Third Pass

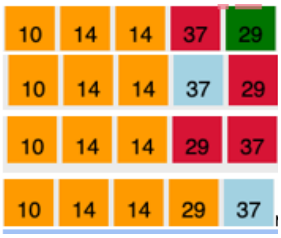
Now the current index is 2 and the value is 29, we will find values smaller than 29 if any

14 is the smallest value and the index is 4th

We swap current index 2 and index 4(Index of 14)

Fourth Pass

Now the current index is 3 and the value is 37, we will find elements smaller than 37 is any.



29 is the smallest element and the index is 4

We swap current index and index 4 (index of 29).

Now the current index is last, so our algorithm stops and we got the sorted array.

The array is now sorted in non decreasing order.

Program for Selection Sort.

LP_CODE3.java

```
Enter the size of array
5
Enter the elements
10 40 30 50 20
Array after sorting
10 20 30 40 50
```

Time Complexity:
The time complexity of the following algorithm will be $O(n^2)$ as at each particular index from left to right as we are iterating in an array, we are finding the minimum value index by traversing through the rest of the array. Thus we require $O(n)$ to find the minimum value index and we are doing this process n time so final time complexity will be $O(n^2)$.

Space Complexity:
It does not require any extra space so space complexity is $O(1)$. Thus it will be an in-place sorting algorithm.

Is selection sort stable?
As we know(Explained above) A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output as they appeared in the input unsorted array.

Selection sort works by finding the minimum element in the array and then inserting it in its correct position by swapping with the element which has minimum value. This is what makes it unstable.

Let's see an example :
arr[] ={2,3,2,1}
In the first iteration of the outer for-loop, the algorithm determines that "1"is the minimal element and exchanges it with the blue "2" :
arr[] ={1,3,2,2}
Then, it finds that the red 2 is the minimal item in the rest of the array and swaps it with "3":
arr[] ={1,2,3,2}
Finally, since 2 < 3 Selection Sort swaps the blue Deer with Monkey:
arr[] ={1,2,2,3}

As you can see, arr doesn't maintain the relative order of the two numbers. Since they are equal, the one that was initially before the other should come first in the output array. But, Selection Sort places the red 2 before the blue one even though their initial relative order was opposite.

So, we can conclude that **Selection Sort isn't stable.**

Next class Teaser

- Number theory
- Bits data structure
- Operators on bits