SELECTION SORT

1) Theory:

Let us assume the elements that needs to be sorted are

20	35	-15	7	55	1	-22
0	1	2	3	4	5	6

→ LastUnsortedIndex is a variable used to track till where the array has been sorted

→ i is a traversal variable from left to right

→ largest is a variable that will consist of largest element, which is initially assumed to be at index 0

At this point,

LastUnsortedIndex = 6, i = 0, largest = 20

Now, 20 < 35, largest = 35, i = 1; i++

Now, 35 > -15, largest = 35, i = 2; i++

Now, 35 > 7, largest = 35, i = 3; i++

Now, 35 < 55, largest = 55, i = 4; i++

Now, 55 > 1, largest = 55, i = 5; i++

Now, 55 > -22, largest = 55, i = 6; i++

↑ i has reached LastUnsortedIndex

∴ Swap LastUnsortedIndex with largest and LastUnsortedIndex--

Now, list is as follows

20	35	-15	7	-22	1	55
0	1	2	3	4	5	6

LastUnsortedIndex = 5, i is reinitialized to 0, largest is reassigned to element in index 0

i = 0; largest = 20

Elements: 20 (0) 35 (1) -15 (2) 7 (3) -22 (4) 1 (5) | 55 (6)

Now, 20 < 35; largest = 35, i = 1; i++
Now, 35 > -15; largest = 35, i = 2; i++
Now, 35 > 7; largest = 35, i = 3; i++
Now, 35 > -22; largest = 35, i = 4; i++
Now, 35 > 1; largest = 35, i = 5; i++

↑ i has reached to lastUnsortedIndex

∴ Swap largest with lastUnsortedIndex and lastUnsortedIndex--

Now, list is as follows: 20 (0) 1 (1) -15 (2) 7 (3) -22 (4) | 35 (5) | 55 (6)

lastUnsortedIndex = 4, i is reinitialized to 0, largest is reassigned to the element at index 0
i = 0; largest = 20

Elements: 20 (0) 1 (1) -15 (2) 7 (3) -22 (4) | 35 (5) | 55 (6)

Now, 20 > 1; largest = 20; i = 1; i++
Now, 20 > -15; largest = 20; i = 2; i++
Now, 20 > 7; largest = 20; i = 3; i++
Now, 20 > -22; largest = 20; i = 4; i++

↑ i has reached to lastUnsortedIndex

∴ Swap largest with LastUnsortedIndex and LastUnsortedIndex--

Now, list is as follows: -22 (0) 1 (1) -15 (2) 7 (3) | 20 (4) | 35 (5) | 55 (6)

LastUnsortedIndex = 3, i is reinitialized to 0, largest is reinitialized to the element at index 0

∴ largest = -99, i = 0

Elements:

-99	1	-15	7	20	35	55
0	1	2	3	4	5	6

Now, -99 < 1, largest = 1, i = 1; i++

Now, 1 > -15, largest = 1, i = 2; i++

Now, 1 < 7, largest = 7, i = 3; i++

↑ i has reached to LastUnsortedIndex

∴ Swap largest with LastUnsortedIndex and LastUnsortedIndex - - ; LastUnsortedIndex = 2

Now, list is as follows

-99	1	-15	7	20	35	55
0	1	2	3	4	5	6

Now, -99 < 1, largest = 1, i = 1; i++

Now, 1 > -15, largest = 1, i = 2; i++

↑ i has reached to LastUnsortedIndex

∴ Swap largest with LastUnsortedIndex and LastUnsortedIndex --

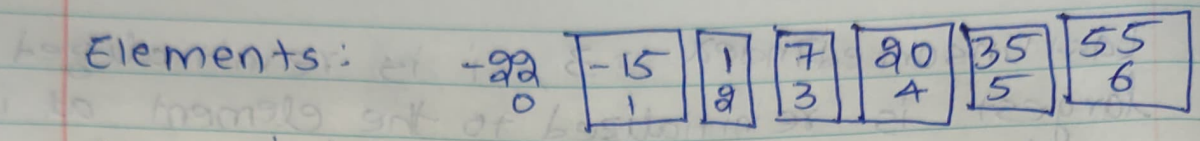
Now, list is as follows

-99	-15	1	7	20	35	55
0	1	2	3	4	5	6

LastUnsortedIndex = 1, i is reinitialized to 0 largest is reassigned to the element at index 0;

largest = -99, i = 0

Elements:



Now, i has reached to 0, the condition is as follows if lastUnsortedIndex reaches 0, control breaks out of the loop.

→ In place algorithm
→ Time complexity ← $O(n^2)$ ← Quadratic
→ It will take 100 steps to sort 10 items, 10,000 steps to sort 100 items
→ Does not require as much swapping as bubble sort
→ unstable Algorithm

Implementation of selection sort

```
import java.util.Scanner;
class SelectionSort
{
  Public static void main(String[] args)
   {
      // Collection of data
      /* To collection the data as an input from the
         user, remove these comments
      int array[] = new int[10];
      Scanner in = new Scanner(System.in);
      System.Out.println("Enter the size of array:");
      int n = in.nextInt();
      System.out.println("Enter Elements");
```

```
forlint i=0; i<n; i++)
{
   array[i] = in.nextInt();
}
*/
//Hard-code
int array[] = {20, 35, -15, 7, 55, -1, -22};
// For SelectionSort logic refer to notes
forlint lastUnsortedIndex = array.length-1; lastUnsortedIndex>0;
lastUnsortedIndex--){
int largest = 0;
forlint i=1; i<=lastUnsortedIndex; i++)
{
  if(array[i] > array[largest])
  {
     largest = i;
  }
}
  swap(array, largest, lastUnsortedIndex);
}
//Printing elements using for-each loop
for(int element : array)
{
  System.out.println(element);
}
}
Public static void swap(int[] array, int i, int)
{
```

```
if(i == j) { return; }
int temp = array[i];
array[i] = array[j];
array[j] = temp;
} //Swap
} // class SelectionSort
```