BUBBLE SORT ALORITHM

The bubble sort is a sorting algorithm that compares two elements that are close to each other and then swapping them in the previous element is larger than the current element   
  
 -We are going to start off at1

So our current is =1 and

Our previous element is Current -1=0

In this example we can see that 7 (prev ) is Larger than 2(Current ) therefore we swap the two

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 2 | 1 | 8 | 3 | 1 | 11 |

Now we move on to the next element so current is now c=2 and prev =1

We compare the two and find that 7 is larger than 1 and we swap them

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 7 | 1 | 8 | 3 | 1 | 11 |

We continue this until the end essentially bubbling the largest element to the end

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 8 | 3 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 8 | 3 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 8 | 3 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 3 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 3 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 3 | 1 | 8 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 3 | 1 | 8 | 11 |

From this we can clearly see that last element is our largest element in the set thus we continue to do the same thing to find the second largest and so on   
  
Again we are starting at c=1 and prev =c-1 but this time we end at Number of elements -1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 2 | 1 | 7 | 3 | 1 | 8 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 7 | 3 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 7 | 3 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 7 | 3 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 8 | 1 | 11 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 7 | 1 | 8 | 11 |

From this we see now that 8 is our second largest number and then we continue to do the same thing until we get all the numbers in order

The end result ?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 3 | 7 | 8 | 11 |

But how do we write it in code?

We are going to use nested loops

One to compare elements on the list and one to iterate to the next comparison

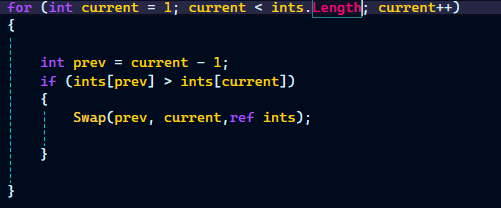
The code (Simple integers array)

In the following we have the following array of integers



Now how do we sort it

First the for loop that compares elements



And a swap method to swap 2 elements in the array

A screen shot of a computer code

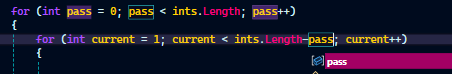
Description automatically generated

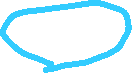
What else is missing?,The next iteration of the methos so we need another for loop

A computer screen shot of a code

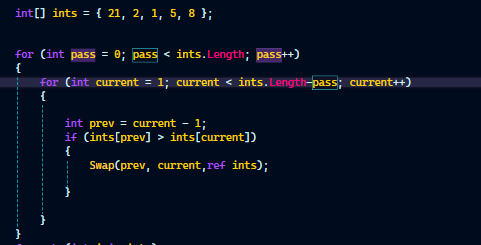
Description automatically generated

But we have to make sure we also don’t compare the last element so in our inner loop we are going to decrease the maximum limit by one with each pass we do this buy decreasing pass from the limit





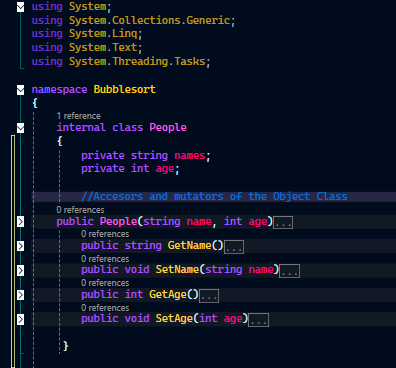
And the completed structure looks like this



I’ve placed the code in a separate file you can access and mess around

Implications of the Sorting Algorithm With objects

(Sorting Algorithm/Objects)



This is going to be our object class and we are going to work with the following textfile



Now we need to sort the Objects in ascending order of names

So What do we improve in our Algorithm

First we place all that Sorting as a method (Making it reuseable and useable only when called )