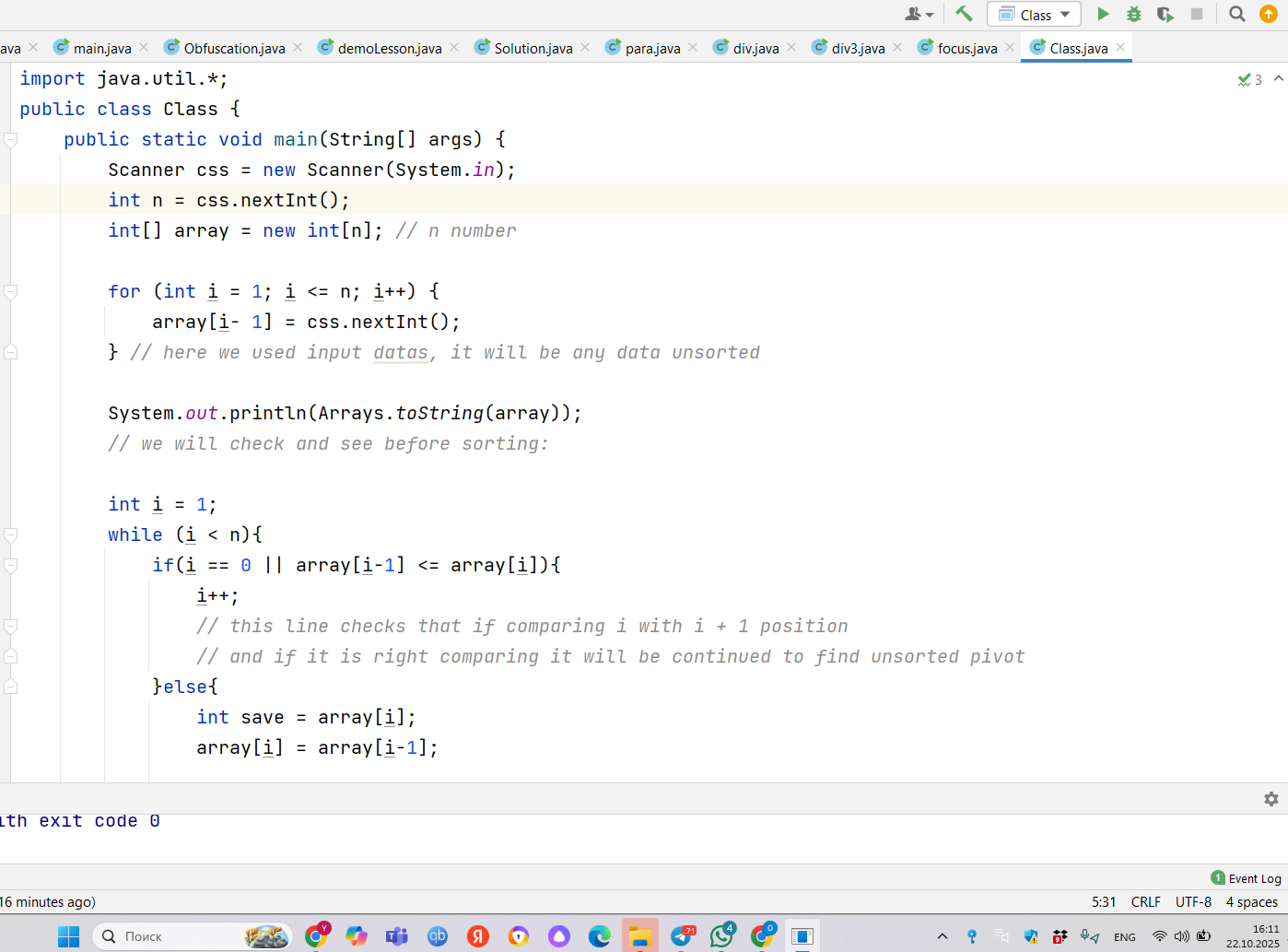
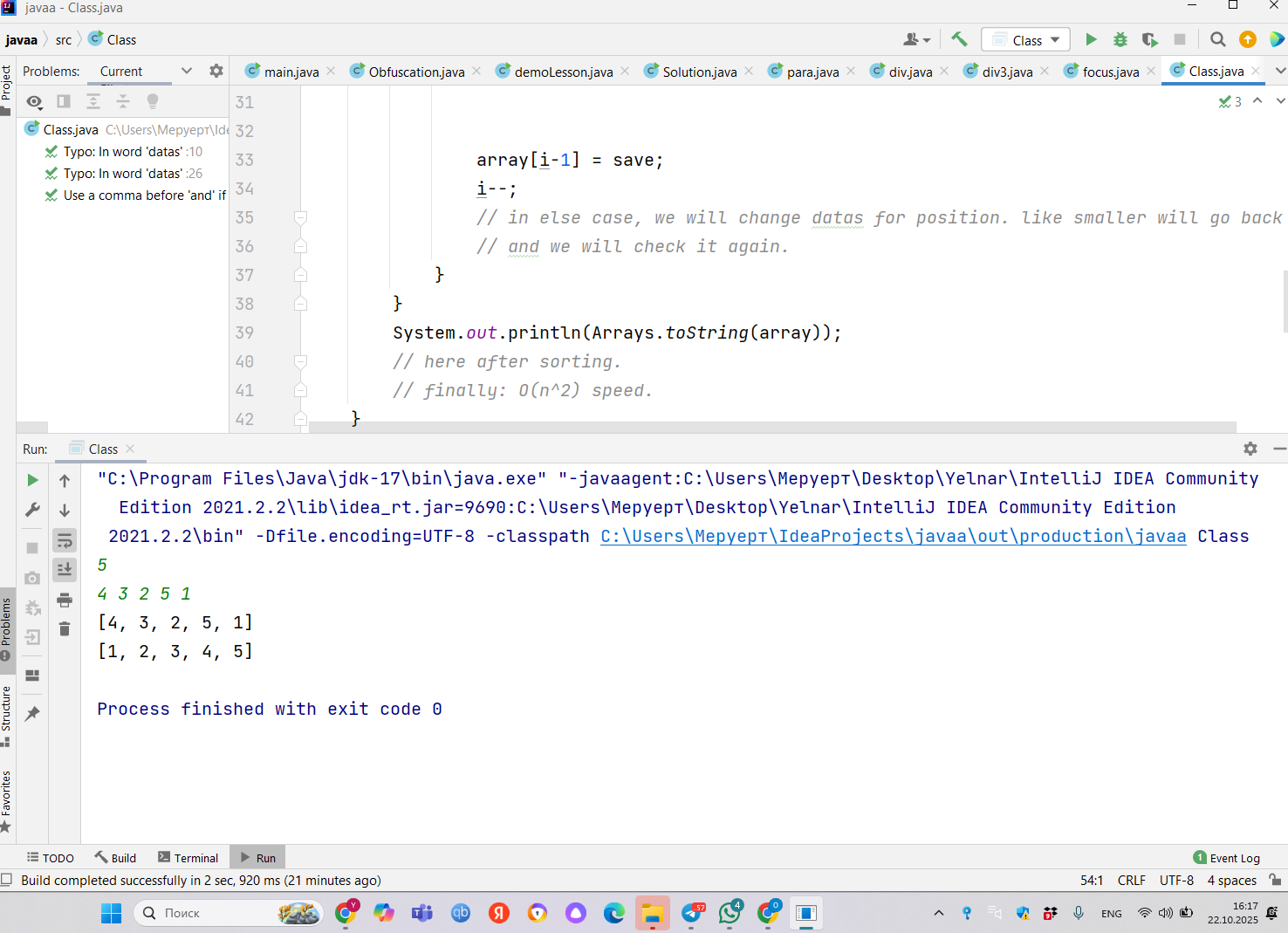
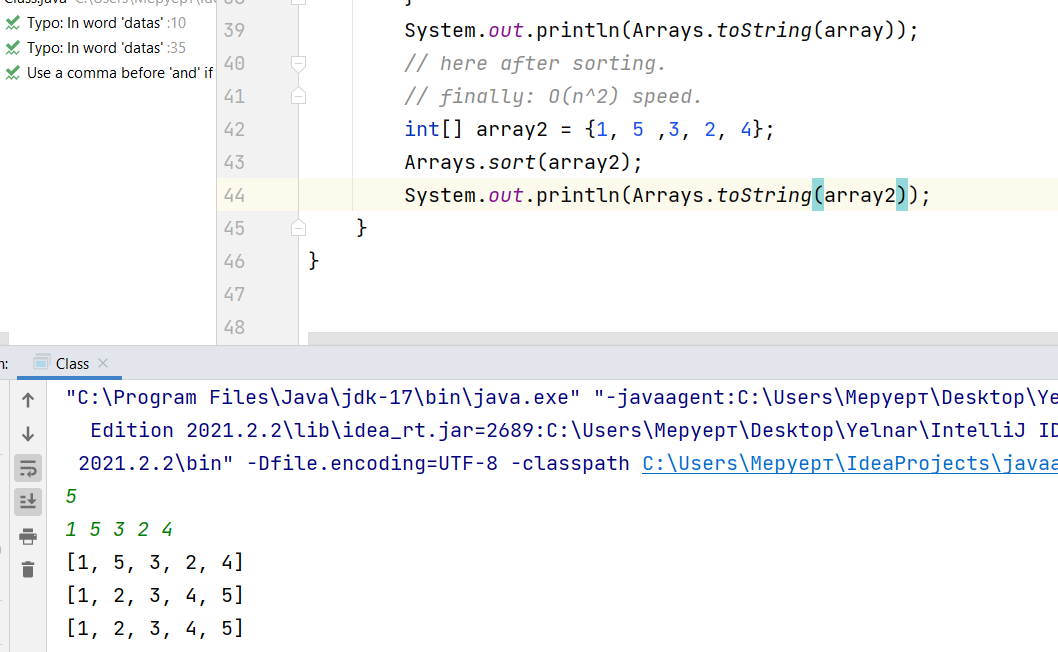
**1. write YOURSELF a basic sorting algorithm.**

**If you can add one of the merge sort, quick sort and heap sort. but rather one bad that you wrote yourself.**





**2. add few test cases: show that your algo work on few cases. For instance compare against sorting utility**

  
  
here i compared sorting utility. for utility i wrote data and gave same data for input. they are same and my code also works.

**3. analysis: write analysis in big O and explain how you apply master theorem and to which problem. You can ask a colleague because you need to be super quick.**  
About analysis, i think in best case it will be: O(n) like one going forward, like ideal case.   
for worst case: it takes O(n^2) must move back.   
  
Space complexity: O(1)  
  
I asked about master theorem from GPT, it is about takes parts and will manage this part. like quicksort. I think i cannot apply master theorem for my code. because it is iterative like goes by cycle(like loop{while, for} ) i mean not recursive. but if we compare it merge sort. mine is slower. (O(n^2)) vs O(n log n)

import java.util.\*;

public class Class {

public static void main(String[] args) {

Scanner css = new Scanner(System.*in*);

int n = css.nextInt();

int[] array = new int[n]; *// n number*

for (int i = 1; i <= n; i++) {

array[i- 1] = css.nextInt();

} *// here we used input datas, it will be any data unsorted*

System.*out*.println(Arrays.*toString*(array));

*// we will check and see before sorting:*

int i = 1;

while (i < n){

if(i == 0 || array[i-1] <= array[i]){

i++;

*// this line checks that if comparing i with i + 1 position*

*// and if it is right comparing it will be continued to find unsorted pivot*

}else{

int save = array[i];

array[i] = array[i-1];

array[i-1] = save;

i--;

*// in else case, we will change datas for position. like smaller will go back*

*// and we will check it again.*

}

}

System.*out*.println(Arrays.*toString*(array));

*// here after sorting.*

*// finally: O(n^2) speed.*

int[] array2 = {1, 5 ,3, 2, 4};

Arrays.*sort*(array2);

System.*out*.println(Arrays.*toString*(array2));

}

}

Input: 5

1 5 3 2 4  
  
Output:

[1, 5, 3, 2, 4]

[1, 2, 3, 4, 5]

[1, 2, 3, 4, 5]