Task01: selection sort

Selection sort Algorithm:

Step1: create array and add elements into it and find size of array n .

Step2: finding the largest number using n -1 and decreasing n value by n--.

step3: swap the number , move the largest element found to end of the unsorted part

step4: Sorting done:

Task02:

pseudo code:

For i = 0 to length(arr) - 1

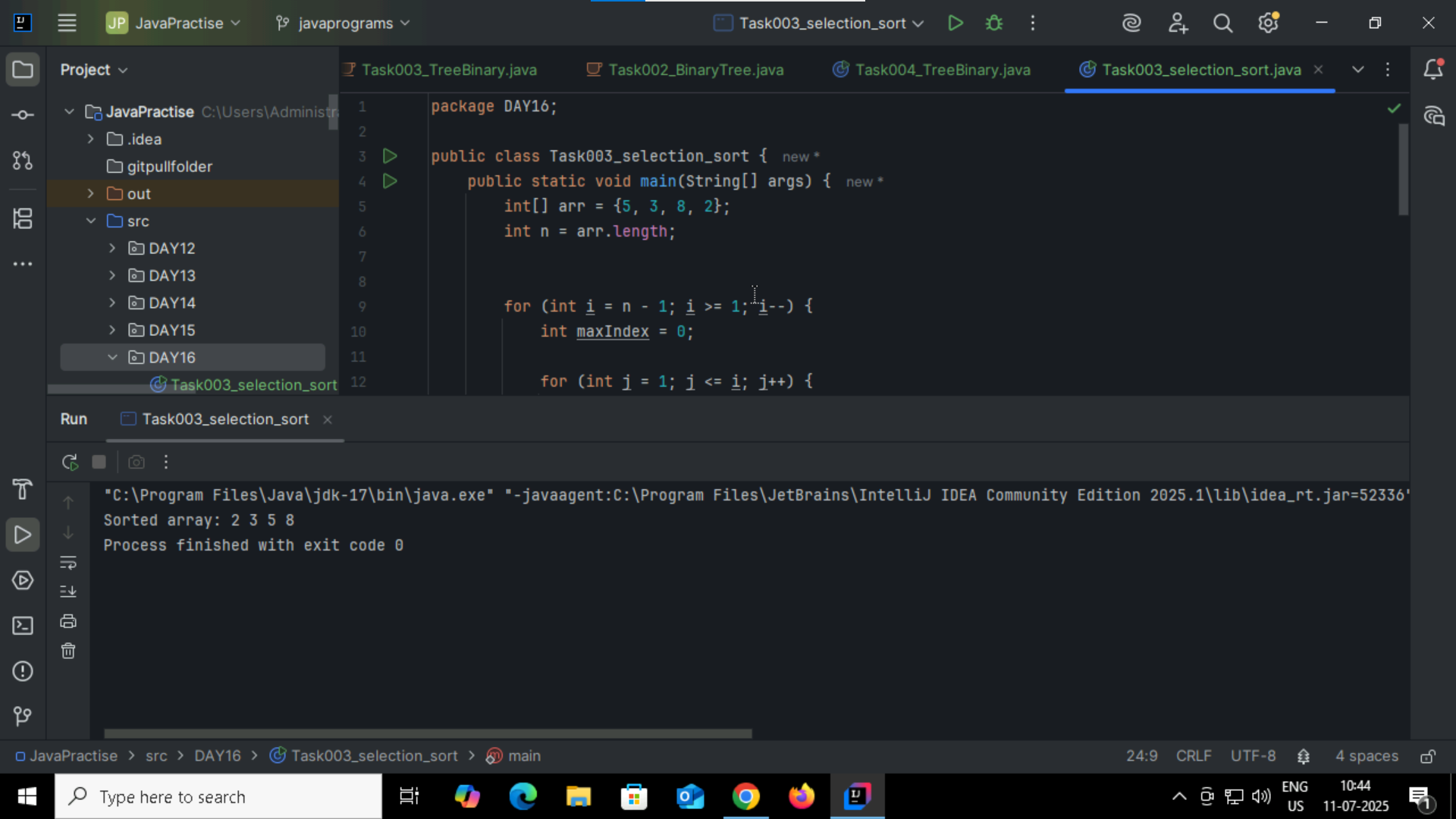
minIndex = i

For j = i + 1 to length(arr) - 1

If arr[j] < arr[minIndex]

minIndex = j

Swap arr[i] and arr[minIndex]

Task03:  


Task04: bubble sort  
  
Algorithm:  
step1: Check if the first element in the input array is greater than the next element in the array.  
step2: If the first element is greater than the second (in ascending order), swap their positions.

Step3:  Move to the second and third elements, compare, and swap if necessary

Step4:  Continue this process for each pair of adjacent elements until the end of the list is reached.

Step5: upto list sort it will repeat loop.  
  
Task05: pseudo code  
  
inti, j, temp;

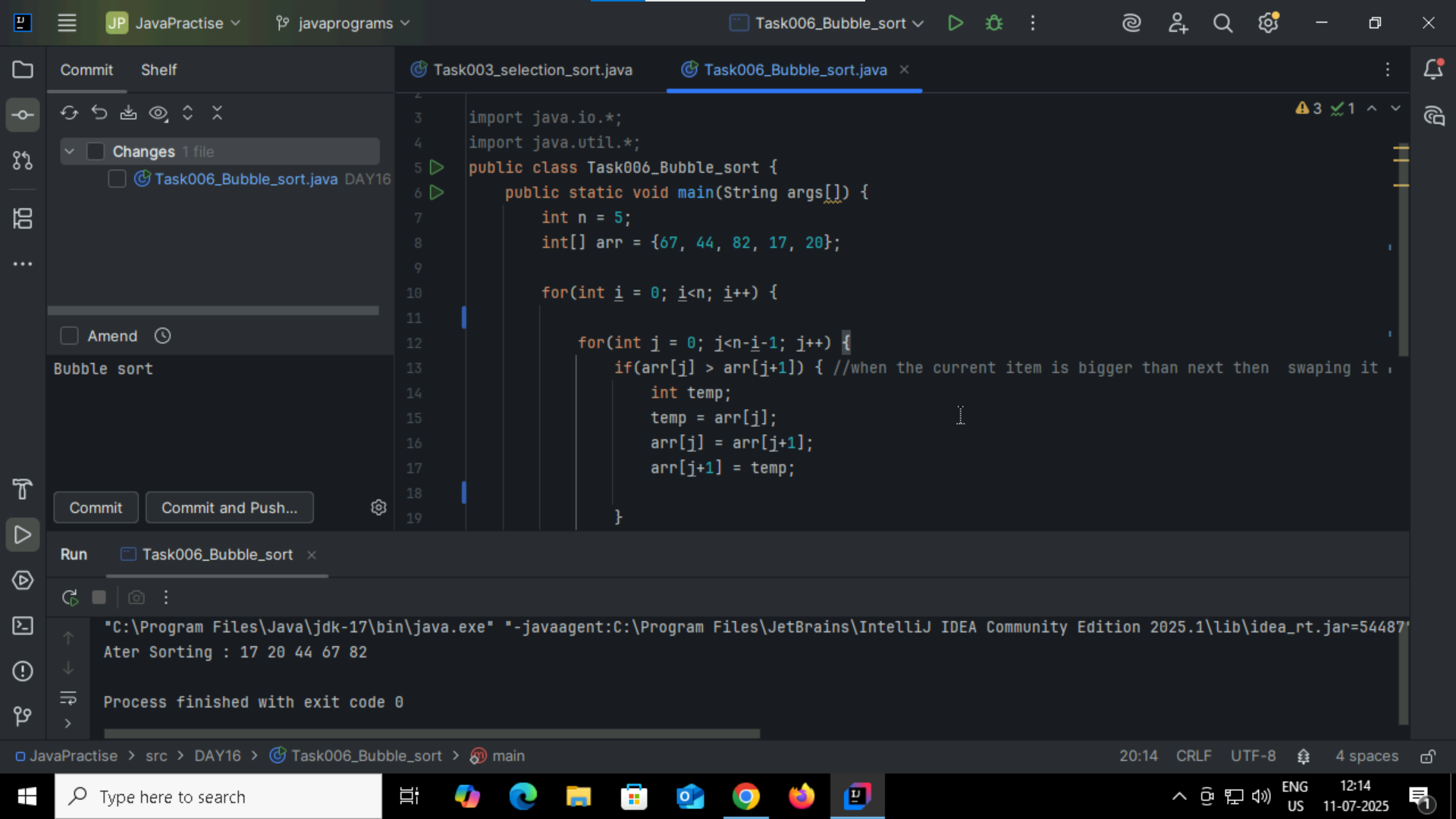
for (i = (array\_size - 1); i>= 0; i--)

for (j = 1; j <= i; j++)

if (numbers[j-1] > numbers[j]){

temp = numbers[j-1];

numbers[j-1] = numbers[j];

numbers[j] = temp;  
  
Task06:  


Task007: Insertion Sort

Algorithm:

Step1 : take first element as sorted because we assume.  
step2: iterates through the remaining elements, one at a time.  
step3: Repeat from i = 1 to n – 1 and compare with other element.  
step4: The current element is then placed in its correct sorted position.  
step4: repeats until all elements have been inserted into the sorted portion.  
  
Task008:  
pseudo code  
for j = 2 to A.length

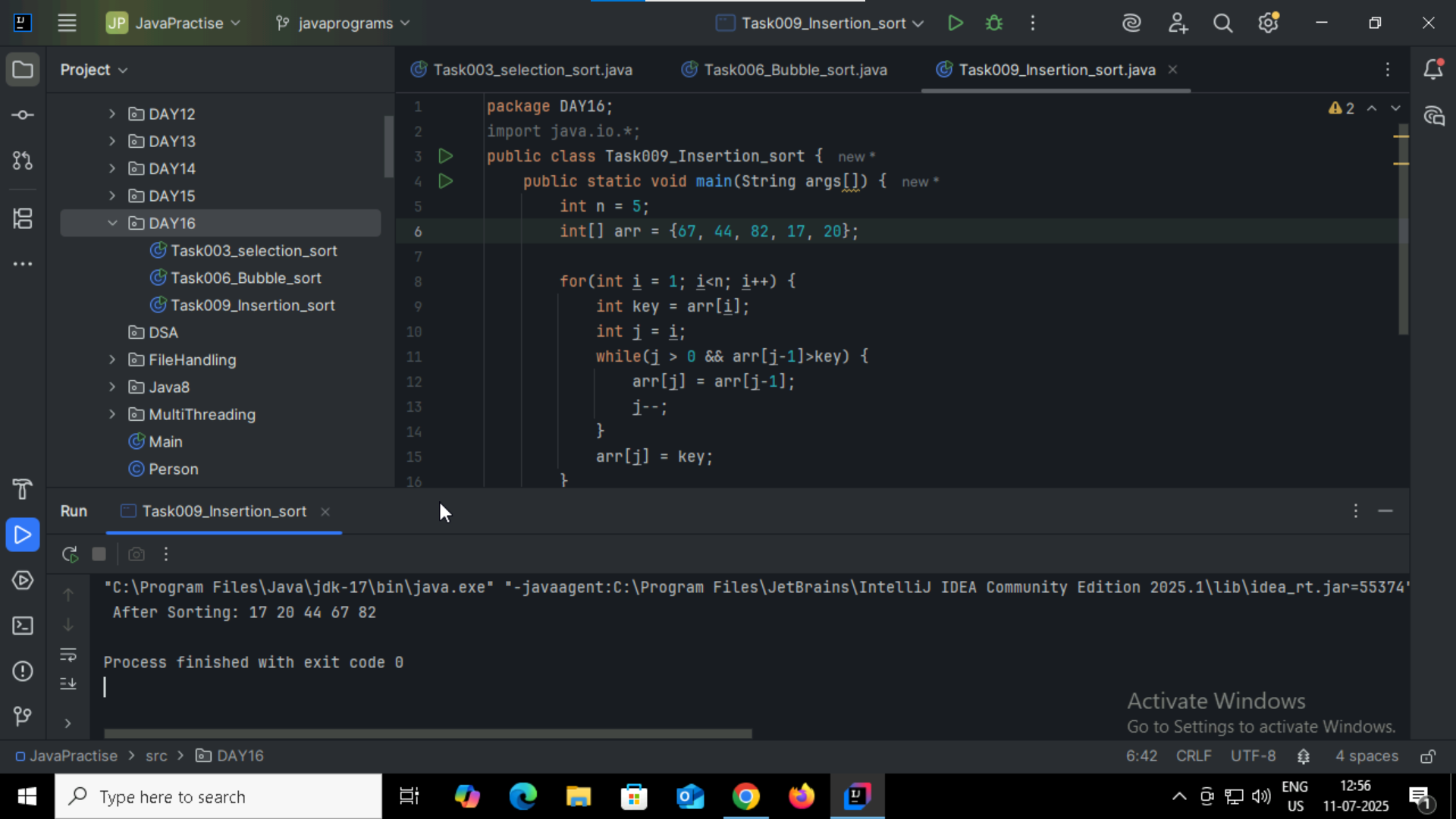
key = A[j]

i = j 1

while i > 0 and A[i] > key

A[i + 1] = A[i]

i = i -1

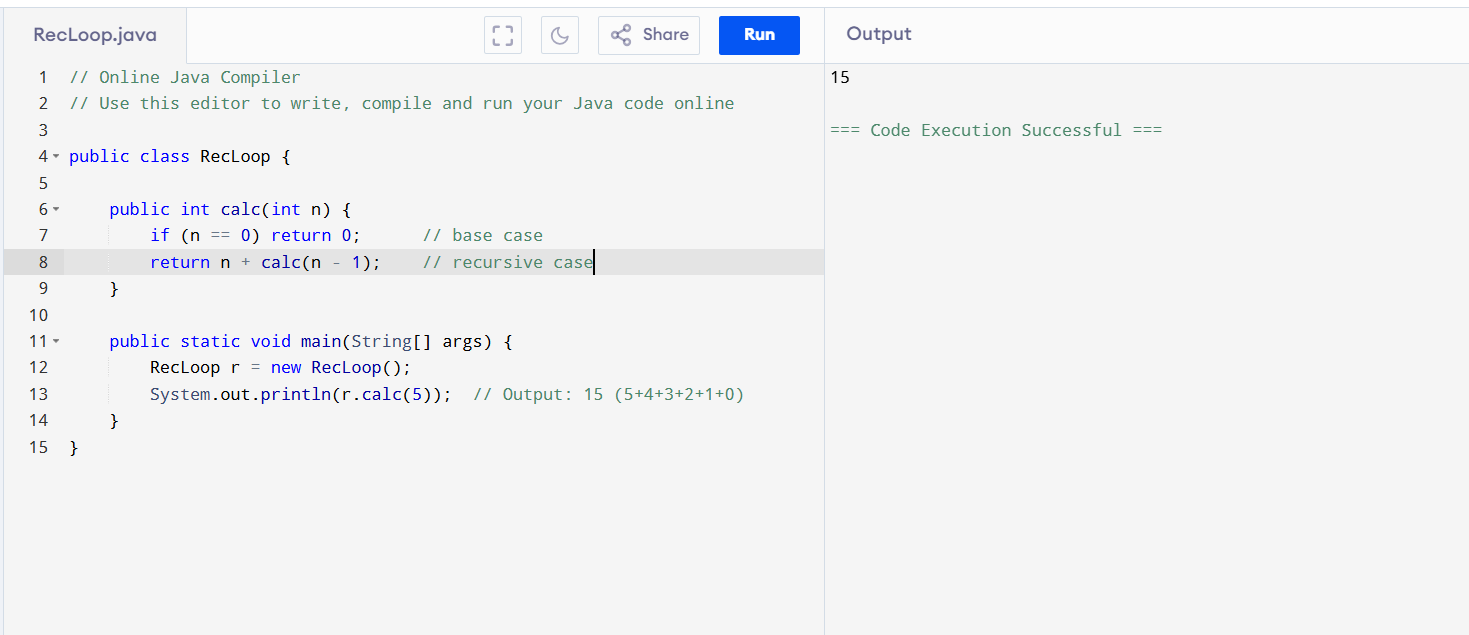
A[i + 1] = key  
  
Task009:  


Task010:

What are the advantages and disadvantages of Bubble sort Algo?

Advantages:  
-beginners friendly  
-wont consume much memory directly work on the input array.  
- if the elements already sorted, it can finish fastly.  
Disadvantages:  
- time complexity is more O(n)2  
- comparatively very slow   
- swapping also takes time.

Task011:



Task012: Merge Sort  
Algorithm:

Step1: If it is only one element in the list, consider it already sorted, so return.  
step2: Divide the array into two parts , left part and right right.  
step3: Recursively apply merge sort to both parts.  
step4:merge the two parts into single array.

Task013:  
pseudo code:  
procedure mergeSort(arr):

if length of arr <= 1:  
 return arr  
mid = length of arr / 2  
left = arr[0 ... mid - 1]  
right = arr[mid ... end]

left = mergeSort(left)  
 right = mergeSort(right)

return merge(left, right)  
  
procedure merge(a, b):

result = empty array

while a and b are not empty:

if a[0] <= b[0]:

append a[0] to result

remove a[0] from a

else:

append b[0] to result

remove b[0] from b

while a is not empty:

append a[0] to result

remove a[0] from a

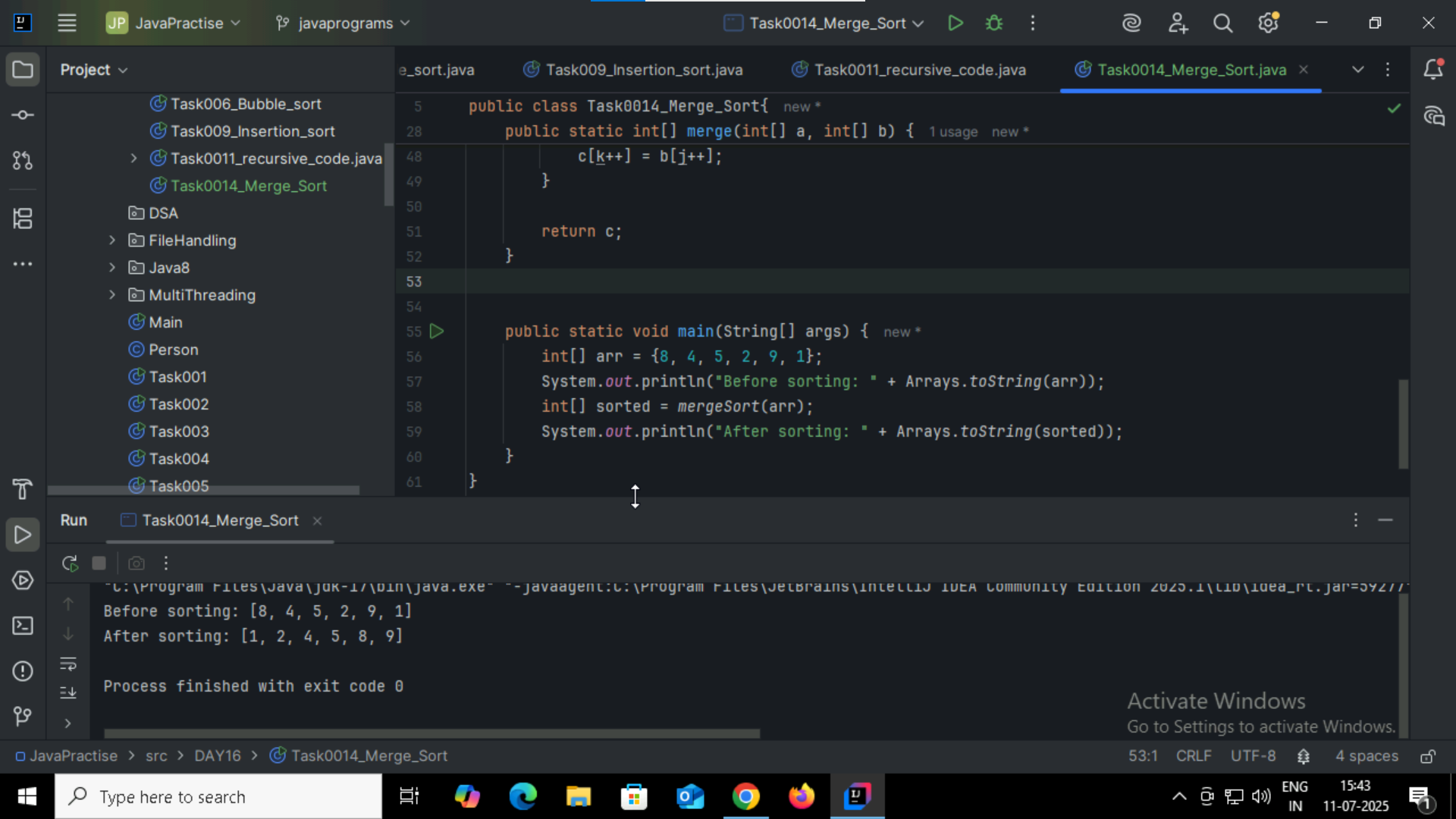
while b is not empty:

append b[0] to result

remove b[0] from b

return result

Task014:



Task015: Quick Sort   
Algorithm  
step1: Choose any index value has pivot  
Step2:Take two variables to point left and right of the list excluding pivot  
step3: While value at left is less than pivot move right While value at right is greater than pivot move left.  
Step4:If both step 5 and step 6 does not match swap left and right  
  
Task016:

procedure quickSort(arr, low, high):

if low < high:

pivotIndex = partition(arr, low, high)

quickSort(arr, low, pivotIndex - 1)

quickSort(arr, pivotIndex + 1, high)

procedure partition(arr, low, high):

pivot = arr[high]

i = low - 1

for j = low to high - 1:

if arr[j] < pivot:

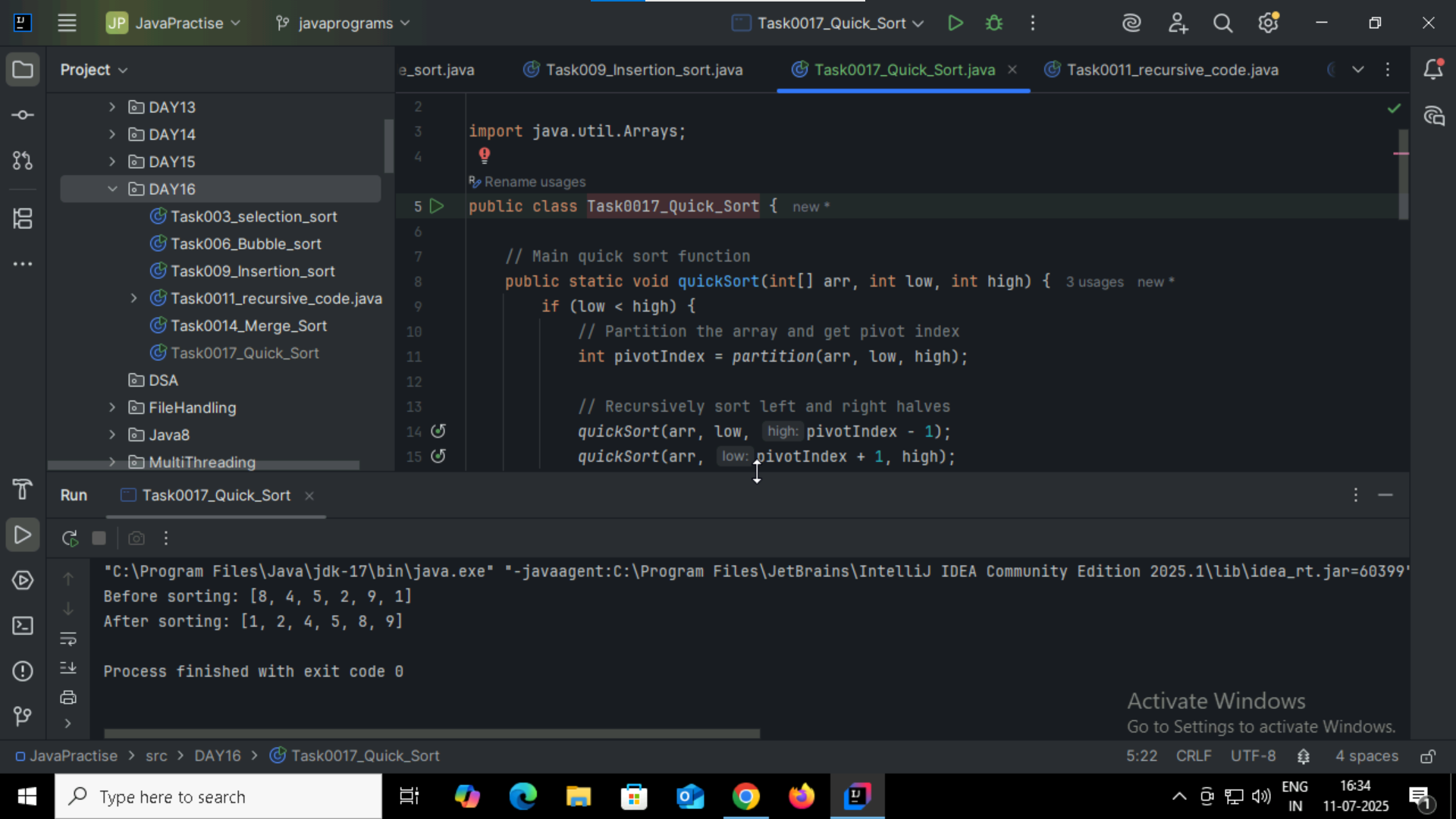
i = i + 1

swap arr[i] and arr[j]

swap arr[i + 1] and arr[high]

return i + 1

Task017:



Add ons:

1.

What is the difference between binary tree and binary search tree (bst)

Can you explain diff between structure and operation of Binary tre and BST.  
\* binary tree \* binary search tree  
  
a each tree has less than 2 childrens A special binary tree where left < root < right  
No specific order between nodes Left child < parent < right child  
Can contain duplicates freely Usually does not allow duplicates (or stored carefully)  
  
2.In sorted array why do you think binary search tree is best than linear search.. Can you ecplain pl  
--  
In a sorted array, binary search (or BST) is much faster than linear search.  
Linear search is simpler but slower for large data.  
Binary search saves time by eliminating half the search space at each step.

3.

Difference between static and dynamic arrays.. Plz list them   
static array dynamic array  
size id fixed can grow size  
compile-time memory allocation run time memory allocation  
faster slower