Day 16 - 104608492 - Shirisha Perapagu

Sorting Techniques and other Home Tasks

**Task1**

Selection sort Algorithm:

1. Start with first element index of the array
2. Find the smallest element in the rest of array and store its index in minIndex.
3. When the smallest element is found, swap it with the first element.
4. Move to the second element and perform steps 1-3 for the second element.
5. Repeat this process until the whole array is sorted.

**Task2**

Selection sort Pseudocode:

procedure SelectionSort(array)

    n = length of array

    for i from 1 to n-1 do

        minIndex = i

        for j from i+1 to n do

            if array[j] < array[minIndex] then

                minIndex = j

            end if

        end for

        temp = array[minIndex]

        array[minIndex] = array[i]

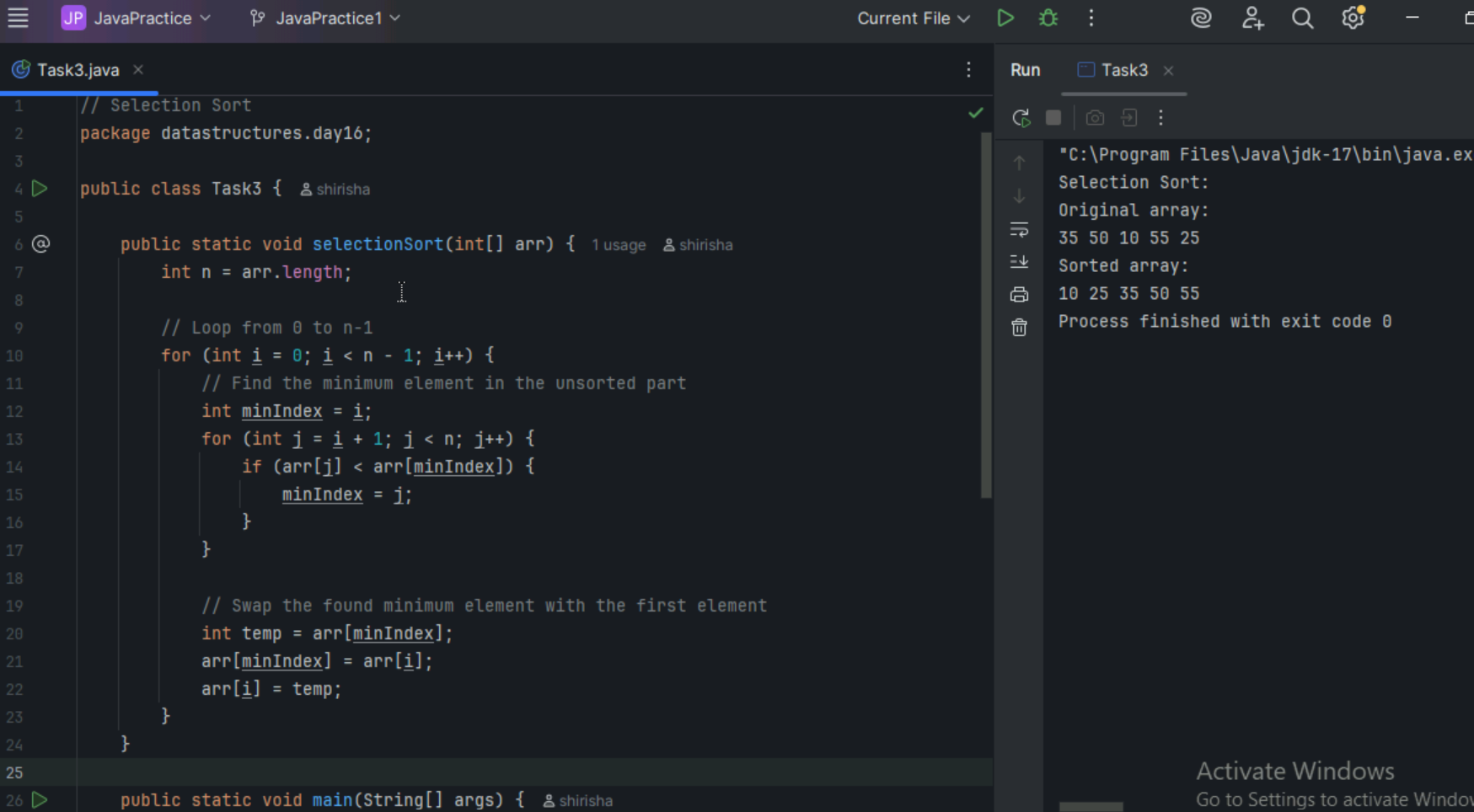
        array[i] = temp

    end for

end procedure

**Task3**

Selection sort Code:



**Task 4**

Bubble Sort Algorithm

1. Start at beginning of the array.
2. Compare 2 adjacent elements.
3. Swap if first number is greater than second.
4. Move to the next pair and repeat.
5. Do this until the end of array.
6. Repeat the whole process for the remaining elements.
7. Stop when no swaps are needed.

**Task 5**

Bubble Sort Pseudocode

procedure bubbleSort(array)

n= length of array

for i=0 to n-1

swapped = false

for j=0 to n-1-i

if array[j]>array[j+1]

swap array[j] and array[j+1]

swapped = true

end if

end for

if not swapped

break

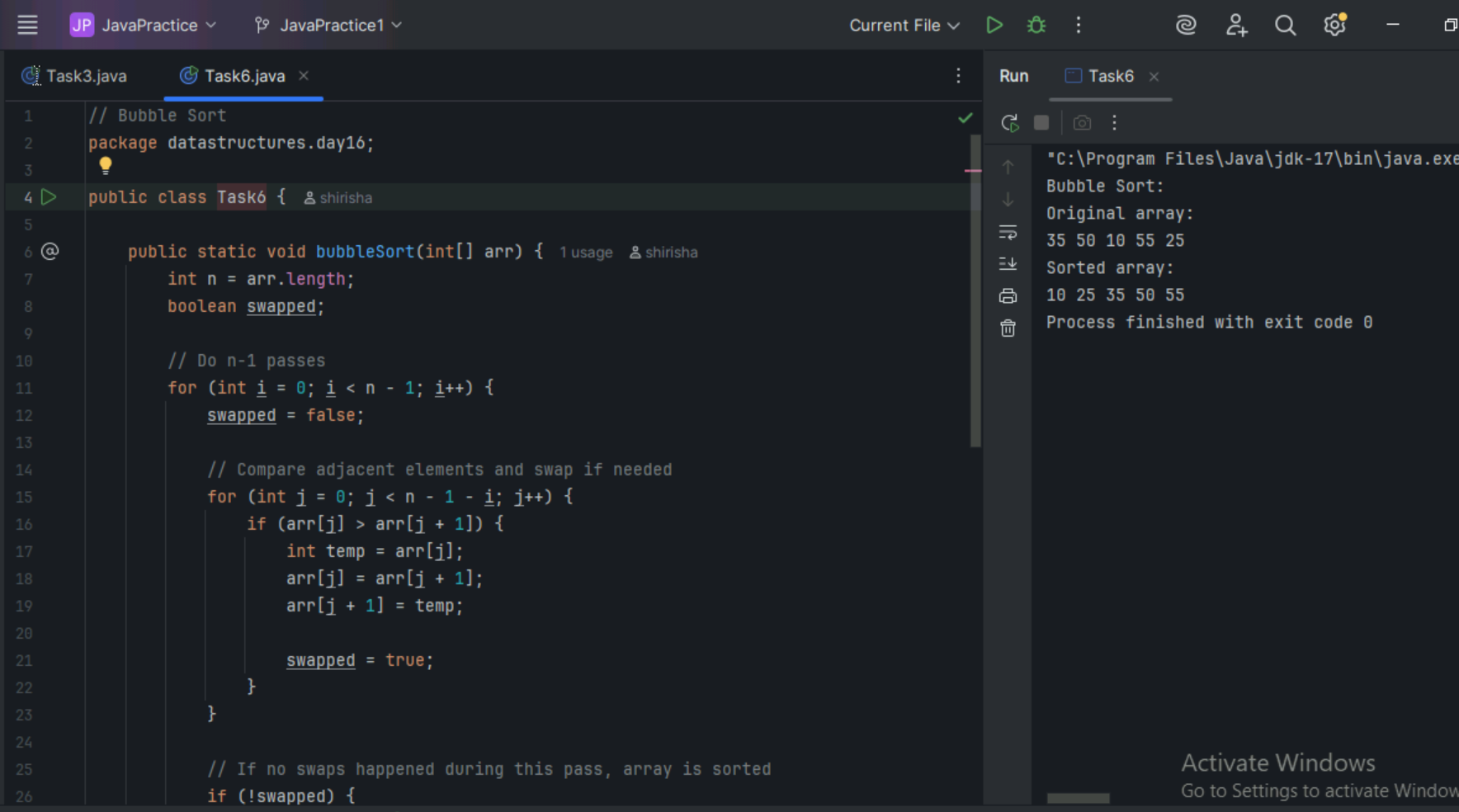
end if

end for

end procedure

**Task 6**

Bubble Sort Code



**Task 7**

Insertion Sort Algorithm

1. Start from second element i.e. index 1
2. Store it in a variable named next.
3. Compare next with elements before it.
4. When the element is greater than next, shift that element one position to right.
5. Insert the next in its correct position.
6. Repeat 2-5 steps for all the remaining elements.

**Task 8**

Insertion Sort Pseudocode

procedure insertionSort(array)

n = length of array

for i = 1 to n – 1

next = array[i]

        j = i – 1

        while j >= 0 and array[j] > next

array[j + 1] = array[j]

j = j - 1

        end while

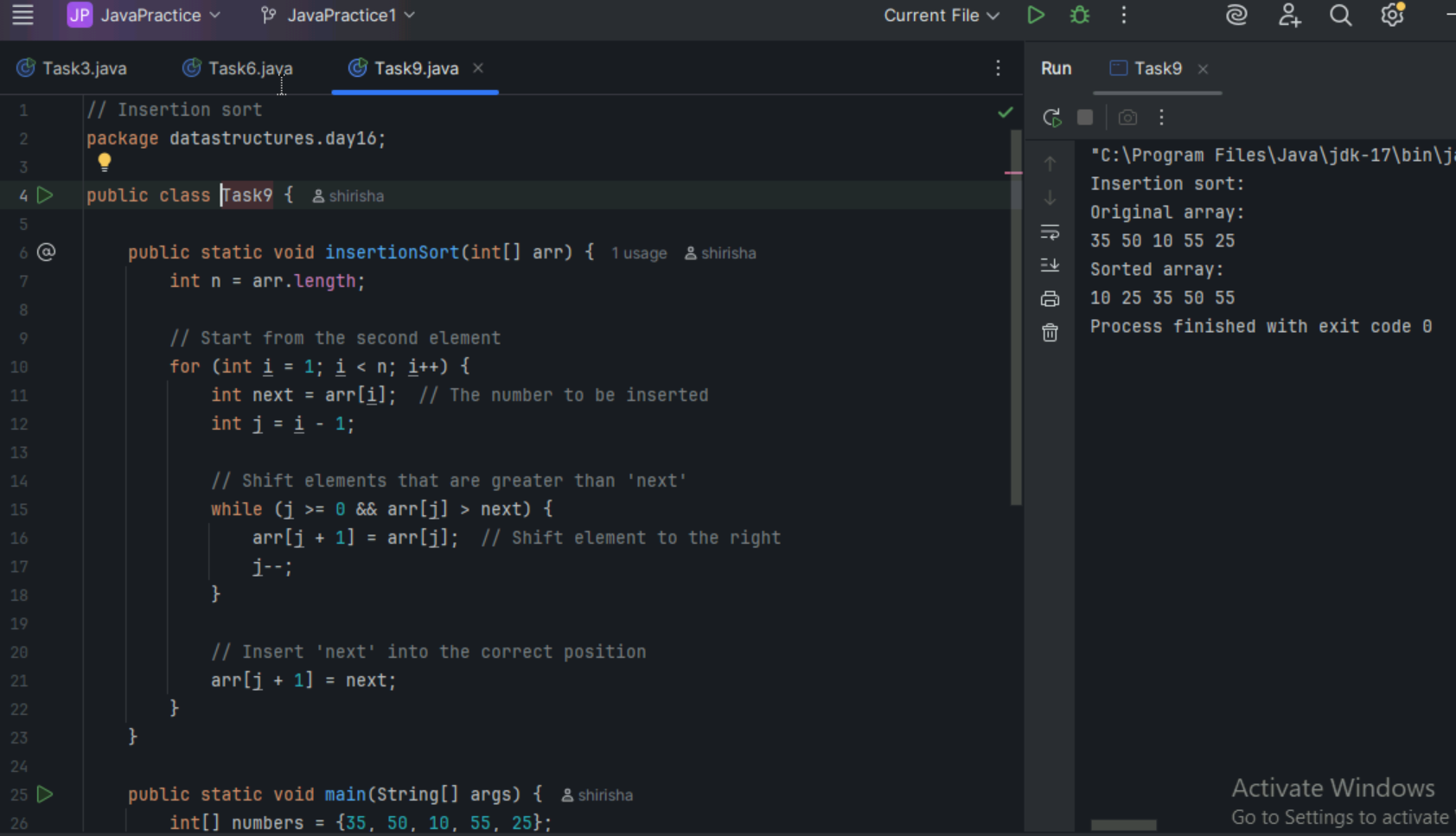
array[j + 1] = next

end for

end procedure

**Task 9**

Insertion sort code



**Task 10**

Advantages and Disadvantages of Bubble sort.

Advantages:

1. Simple and easy to implement
2. Require minimal additional memory space.
3. Efficient for small lists

Disadvantages:

1. Inefficient for large datasets
2. Require many swaps
3. Slower than other algorithms

**Task 11**

What’s the issue with code? Fix it.

public class RecLoop {

     public int calc(int n) {

        if (n == 0) return 0;

        return n + calc(n);

    }

The issue is that the base case isn’t reached as return n + calc(n) will take the n value same every time it is called, hence making it infinite recursion.

So to fix it we must use calc(n-1) instead so that the n value will decrement until base case n=0 is reached.

Fixed code:

public class RecLoop {

     public int calc(int n) {

        if (n == 0) return 0;

**return n + calc(n-1);**  // if n-1 then no infinite recursion

    }

**Task 12**

Merge Sort - Algorithm

1. If the array has one element or empty, it’s sorted do nothing.
2. Divide the array into two halves i.e. the left and right subarrays and find middle point.
3. Recursively sort each half and keep dividing until each subarray has one element.
4. Merge the two sorted halves into one sorted array.

**Task 13**

Merge Sort – Pseudocode

procedure mergeSort(array)

if length of array <=1

return

mid = length of array / 2

create left array(from 0 to mid)

create right array(from mid to end)

Copy array[0 to mid] to left array using System.arraycopy()

Copy array[mid to end] to right array using System.arraycopy()

mergeSort(left)

mergeSort(right)

merge(array,left,right)

end procedure

procedure merge(array,left,right)

initialize i=0,j=0,k=0;

while i<length of left and j<length of right

if left[i] <= right[j]

array[k++] = left[i++]

else

array[k++]=right[j++]

add any remaining elements of left if any

while i<length of left

array[k++] = left[i++]

  add any remaining elements of right if any

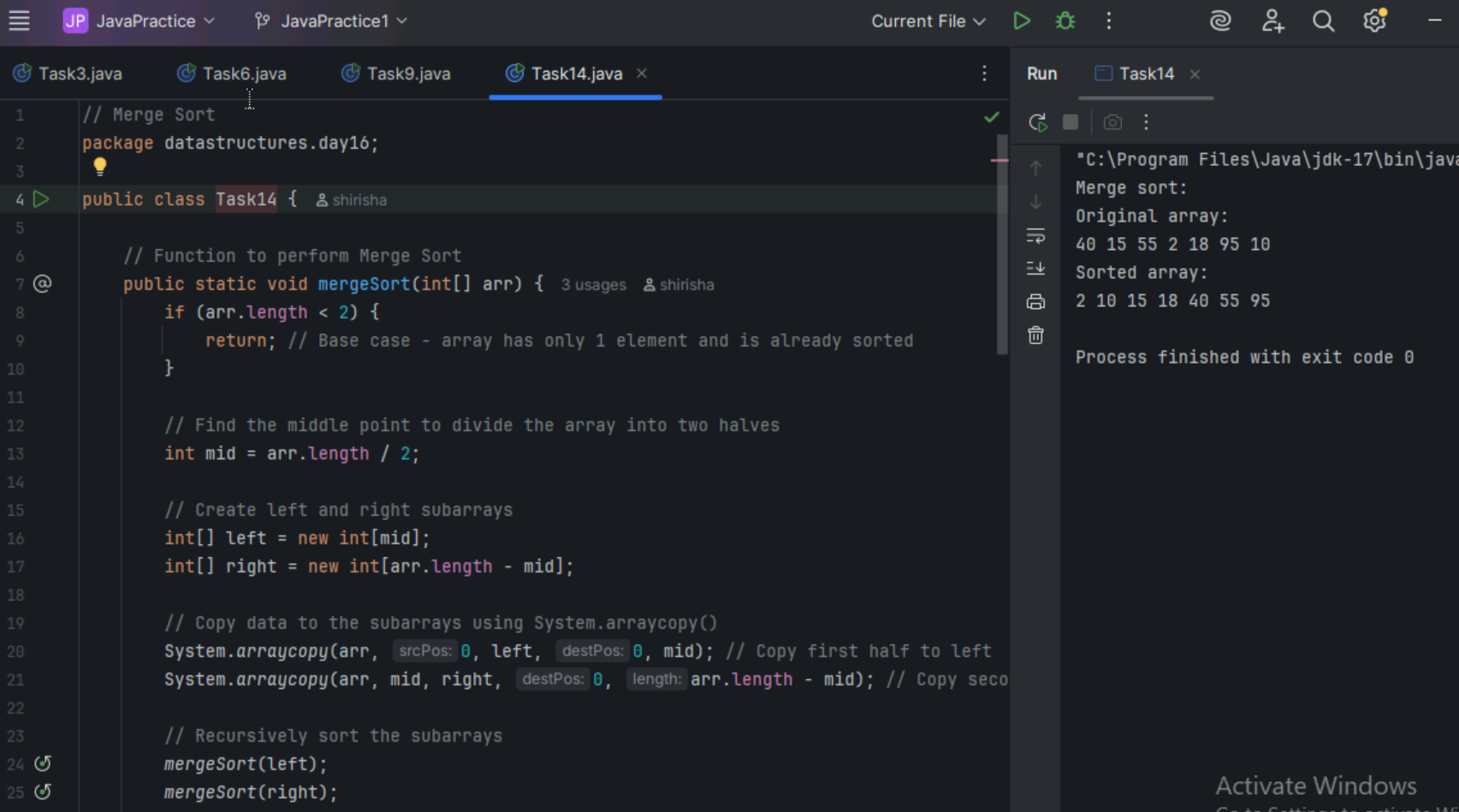
while j<length of right

array[k++]=right[j++]

end procedure

**Task 14**

Merge Sort code



**Task 15**

Quick Sort Algorithm

1. Choose a pivot usually the last element from array.
2. Partition the array i.e. elements smaller than pivot go left and elements larger than pivot go right.
3. Recursively sort the left part and the right part.
4. Stop until base case is reached i.e. when left and right sub-arrays have zero or 1 element, its already sorted.

**Task 16**

Quick Sort Pseudocode

procedure quicksort(array, low, high)

    if low < high then

        pivotIndex = partition(array, low, high)

        quicksort(array, low, pivotIndex - 1)    // sort left part

        quicksort(array, pivotIndex + 1, high)   // sort right part

    end if

end procedure

procedure partition(array, low, high)

    pivot = array[high]

    i = low - 1

    for j = low to high - 1 do

        if array[j] <= pivot then

            i = i + 1

            swap array[i] with array[j]

        end if

    end for

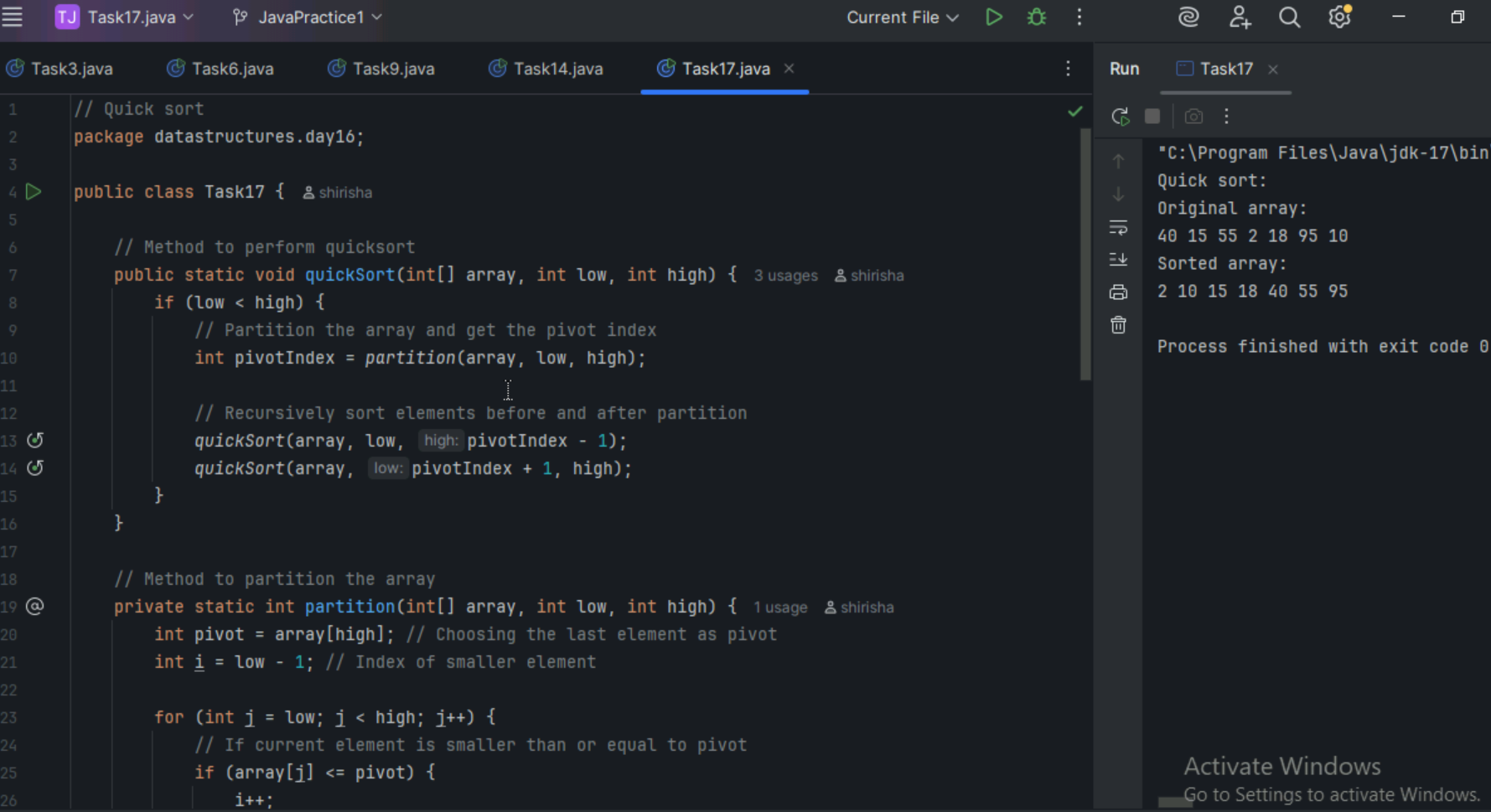
    swap array[i + 1] with array[high]

    return i + 1

end procedure

**Task 17**

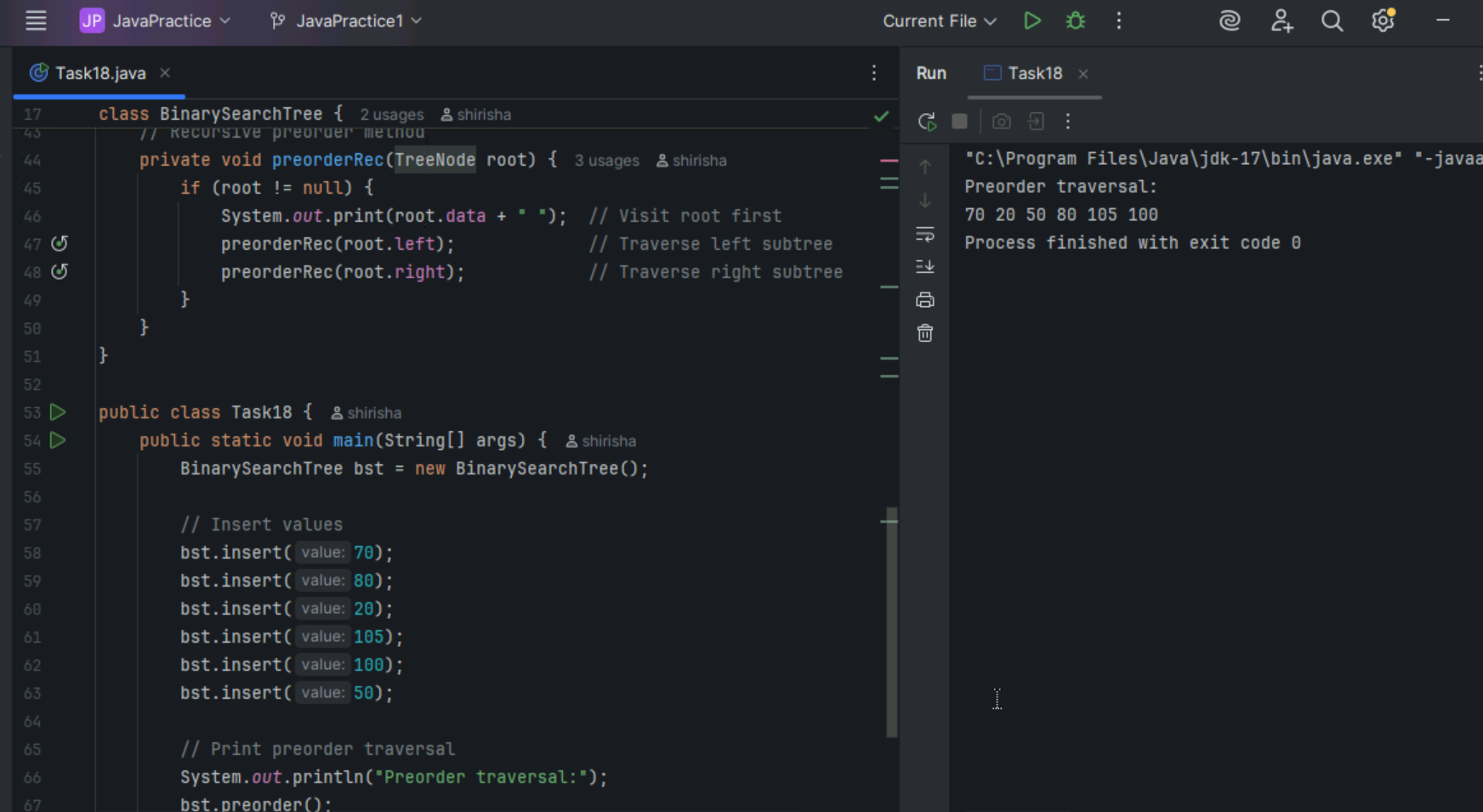
Quick Sort Code



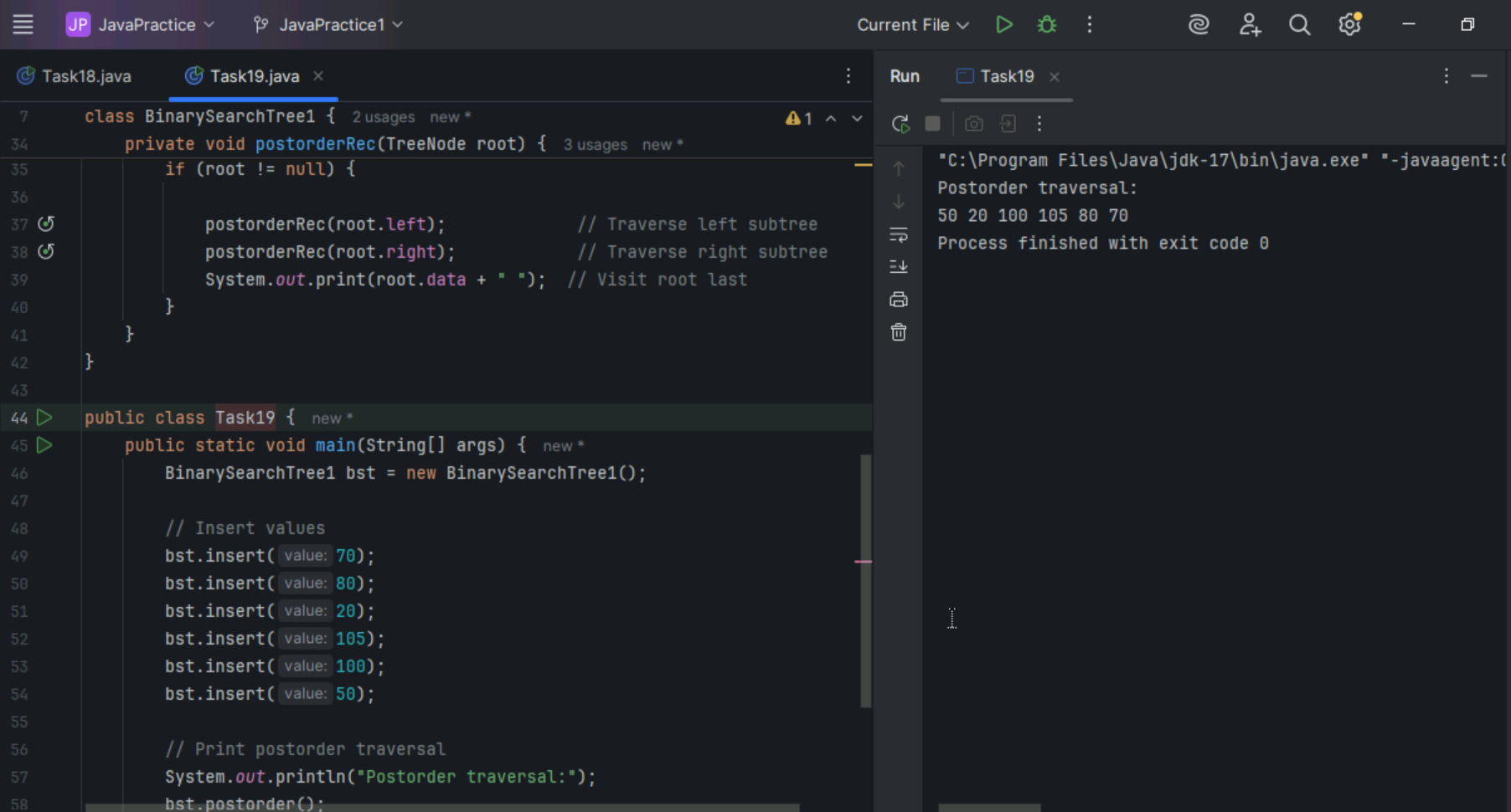
=========================================================================

**Home tasks**

Task18 : Preorder Tree Traversal

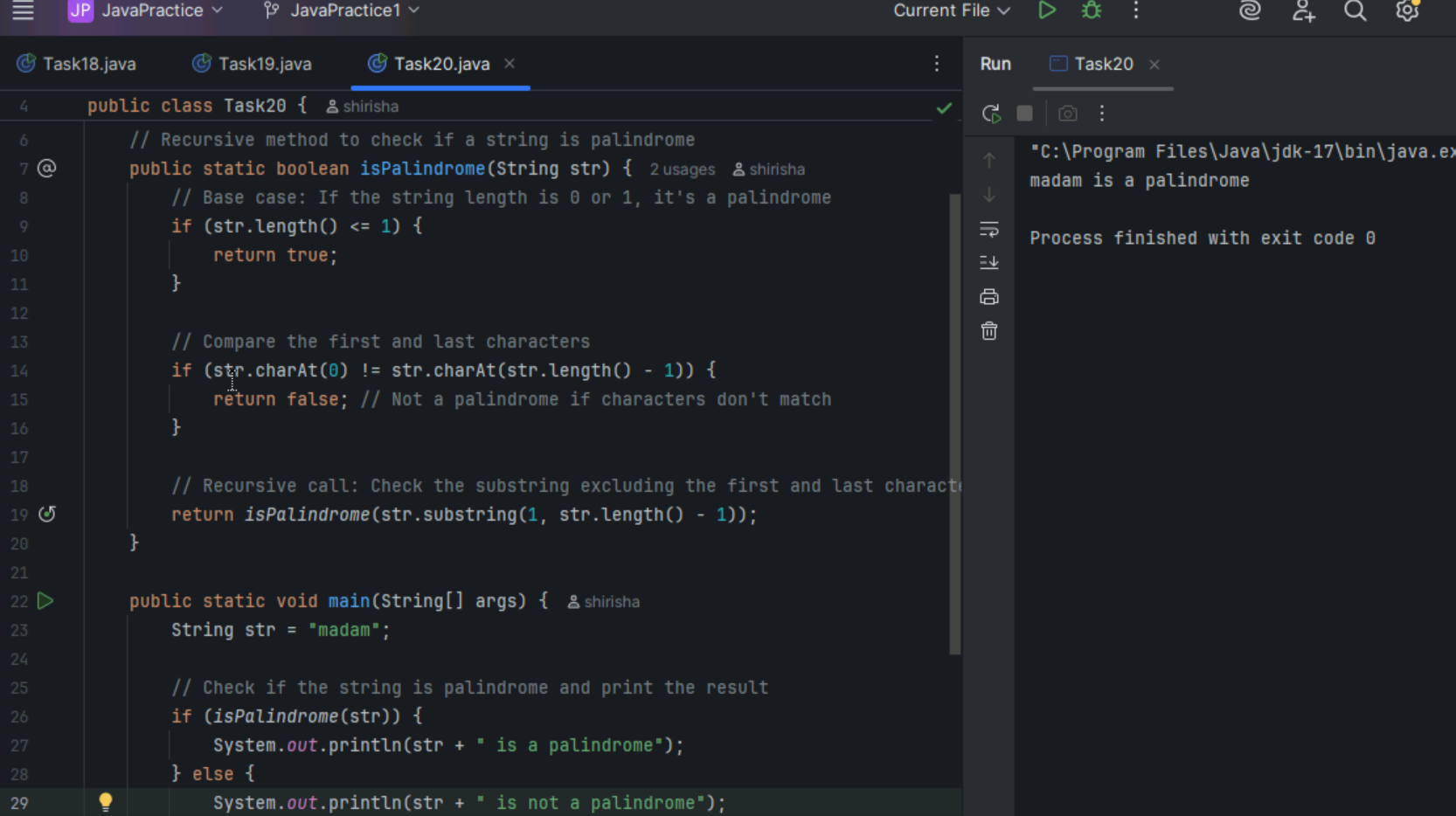


Task 19: Postorder Tree Traversal



Task 20 ( day 14 new Hometask )

Check if a string is a palindrome or not using recursion



Task 21 ( day 14 new Hometask )

Copy Array data to another using Recursion.

