Day 16 – 11/07/2025

Q1. Write an algorithm/step for selection sort.  
Ans.

Selection Sort Algorithm

1. Start with the first element: Consider the first element of the array as the minimum value.
2. Compare with remaining elements: Compare the current minimum value with the remaining elements in the array.
3. Find the minimum value: If a smaller value is found, update the minimum value and its index.
4. Swap the minimum value: After comparing all elements, swap the minimum value with the current element.
5. Repeat the process: Repeat steps 1-4 for the remaining unsorted elements in the array.

Q2. Write pseudocode for selection sort.  
Ans. code –

FOR i FROM 0 TO n-2

minIndex = i

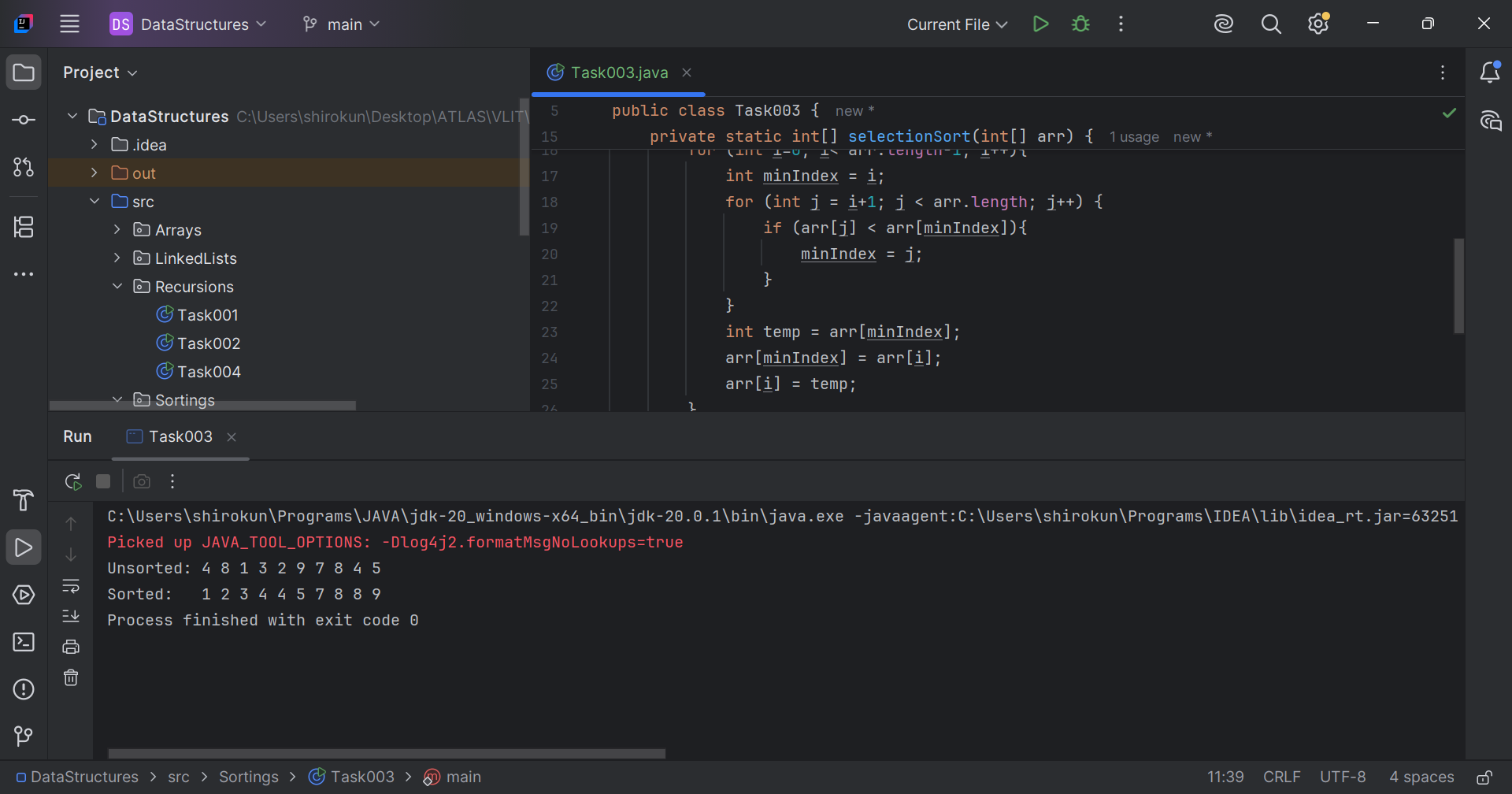
FOR j FROM i+1 TO n-1

IF arr[j] < arr[minIndex]

minIndex = j

SWAP arr[i] WITH arr[minIndex]

// Task003: Selection sort  
  
package Sortings;  
  
public class Task003 {  
 public static void main(String[] args) {  
 int[] sortArr, arr = {4, 8, 1, 3, 2, 9, 7, 8, 4, 5};  
 System.*out*.print("Unsorted: ");  
 for (int a: arr) System.*out*.print(a+" ");  
 sortArr = *selectionSort*(arr);  
 System.*out*.print("\nSorted: ");  
 for (int a: sortArr) System.*out*.print(a+" ");  
 }  
  
 private static int[] selectionSort(int[] arr) {  
 for (int i=0; i< arr.length-1; i++){  
 int minIndex = i;  
 for (int j = i+1; j < arr.length; j++) {  
 if (arr[j] < arr[minIndex]){  
 minIndex = j;  
 }  
 }  
 int temp = arr[minIndex];  
 arr[minIndex] = arr[i];  
 arr[i] = temp;  
 }  
 return arr;  
 }  
}



Q4. Write algorithm for Bubble sort.  
Ans. Bubble Sort Algorithm -

1. Start with the first element: Compare the first element with the next element.
2. Compare adjacent elements: If the current element is greater than the next element, swap them.
3. Repeat the process: Continue comparing and swapping adjacent elements until the end of the array is reached.
4. Repeat passes: Repeat the process until no more swaps are needed, indicating that the array is sorted.

Q5. Write pseudocode for Bubble sort.  
Ans.   
n = length of array

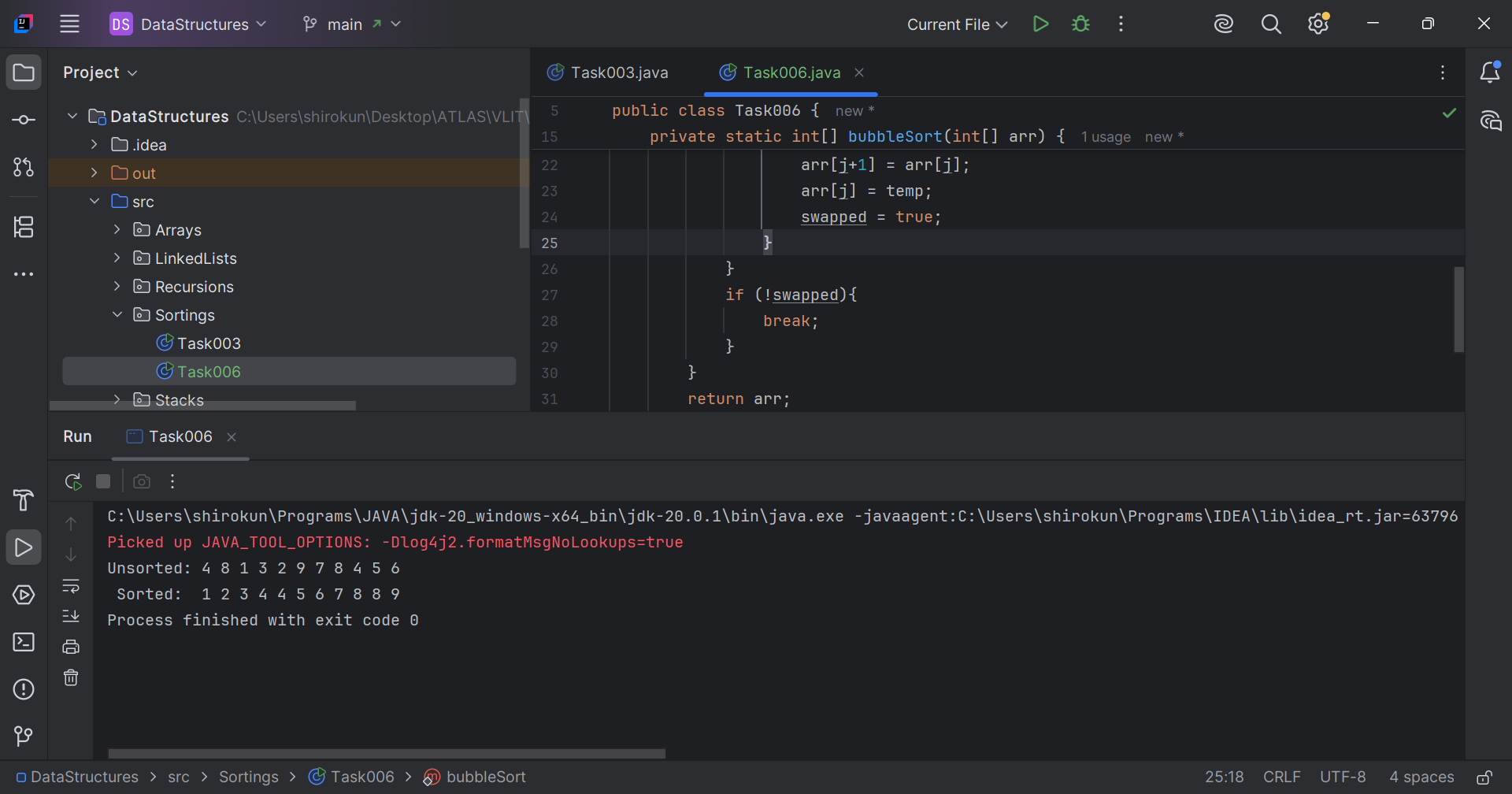
FOR i FROM 0 TO n-2

FOR j FROM 0 TO n-i-2

IF arr[j] > arr[j+1]

SWAP arr[j] WITH arr[j+1]

// Task006: Bubble sort  
  
package Sortings;  
  
public class Task006 {  
 public static void main(String[] args) {  
 int[] sortArr, arr = {4, 8, 1, 3, 2, 9, 7, 8, 4, 5, 6};  
 System.*out*.print("Unsorted: ");  
 for (int a: arr) System.*out*.print(a+" ");  
 System.*out*.print("\n Sorted: ");  
 sortArr = *bubbleSort*(arr);  
 for (int a: sortArr) System.*out*.print(a+" ");  
 }  
  
 private static int[] bubbleSort(int[] arr) {  
 int n = arr.length;  
 for (int i = 0; i < n-1; i++) {  
 for (int j = 0; j < n-i-1; j++) {  
 if (arr[j] > arr[j+1]){  
 int temp = arr[j+1];  
 arr[j+1] = arr[j];  
 arr[j] = temp;  
 }  
 }  
 }  
 return arr;  
 }  
}



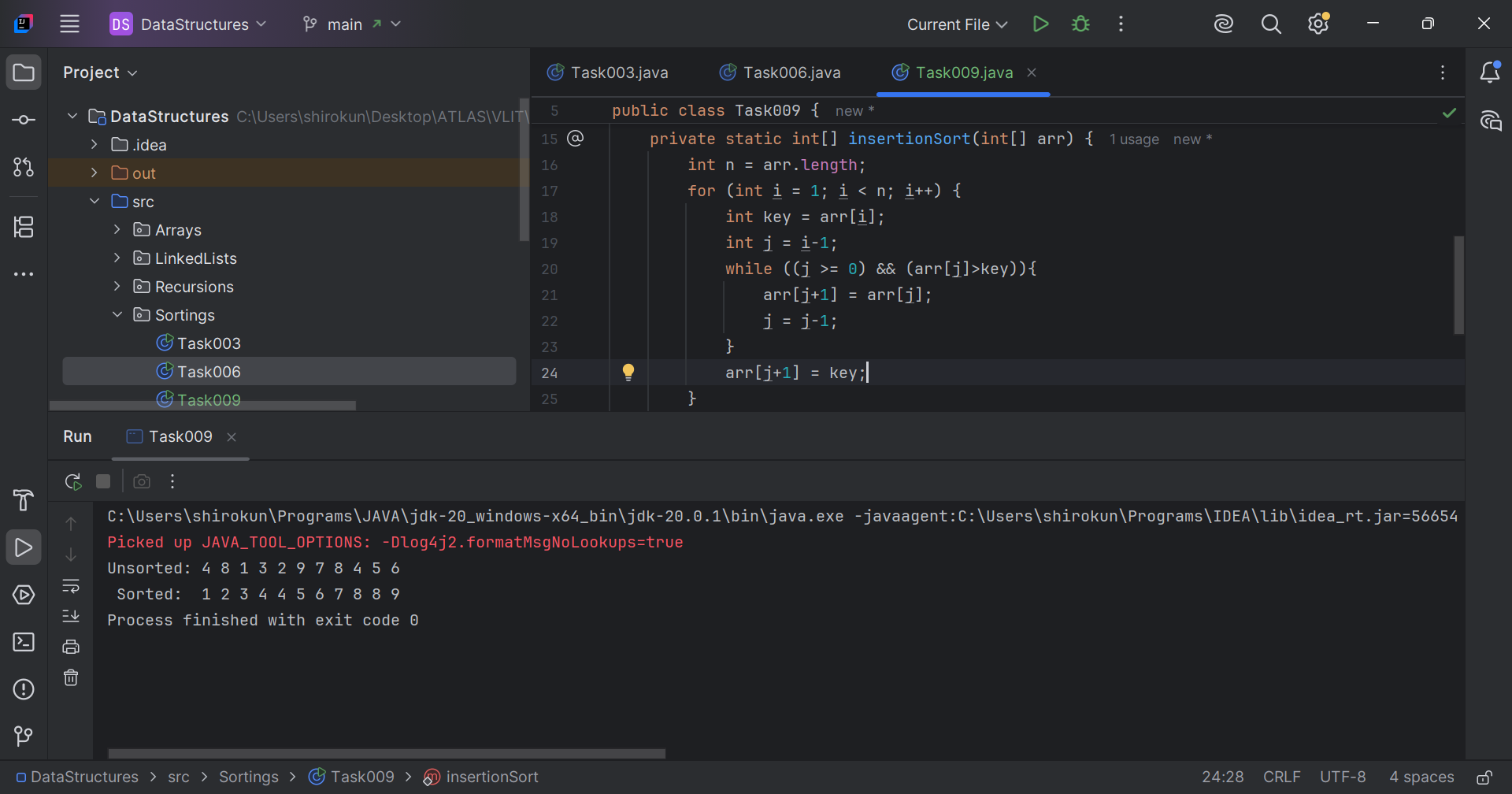
Q7. Write an algorithm for an insertion sort.   
Ans. Algorithm –

1. Start with the second element: Consider the second element as the key.
2. Compare with previous elements: Compare the key with the previous elements in the sorted portion.
3. Shift elements: If the key is smaller than a previous element, shift that element one position to the right.
4. Insert the key: Insert the key at the correct position in the sorted portion.
5. Repeat the process: Repeat steps 1-4 for the remaining elements.

Q8. Write pseudo code for insertion sort.  
Ans. Pseudo code –

* FOR i FROM 1 TO n-1
  + key = arr[i]
  + j = i - 1
  + WHILE j >= 0 AND arr[j] > key
    - arr[j + 1] = arr[j]
    - j = j - 1
  + arr[j + 1] = key

// Task009: Insertion sort.  
  
package Sortings;  
  
public class Task009 {  
 public static void main(String[] args) {  
 int[] sortArr, arr = {4, 8, 1, 3, 2, 9, 7, 8, 4, 5, 6};  
 System.*out*.print("Unsorted: ");  
 for (int a: arr) System.*out*.print(a+" ");  
 System.*out*.print("\n Sorted: ");  
 sortArr = *insertionSort*(arr);  
 for (int a: sortArr) System.*out*.print(a+" ");  
 }  
  
 private static int[] insertionSort(int[] arr) {  
 int n = arr.length;  
 for (int i = 1; i < n; i++) {  
 int key = arr[i];  
 int j = i-1;  
 while ((j >= 0) && (arr[j]>key)){  
 arr[j+1] = arr[j];  
 j = j-1;  
 }  
 arr[j+1] = key;  
 }  
 return arr;  
 }  
}



Q10. What are the advantages and disadvantages of Bubble sort?  
Ans.

Advantages –

* Simple to implement: Bubble Sort is one of the simplest sorting algorithms to understand and implement.
* Space-efficient: Bubble Sort only requires a single additional memory space for temporary swapping, making it space-efficient.
* Stable sorting algorithm: Bubble Sort is a stable sorting algorithm, meaning that the order of equal elements is preserved.

Disadvantages –

* Slow performance: Bubble Sort has a worst-case and average time complexity of O(n^2), making it inefficient for large datasets.
* Not suitable for large datasets: Due to its slow performance, Bubble Sort is not suitable for sorting large datasets or applications that require fast sorting.
* Not efficient for reverse-sorted arrays: Bubble Sort performs poorly on arrays that are reverse-sorted, as it requires the maximum number of swaps.