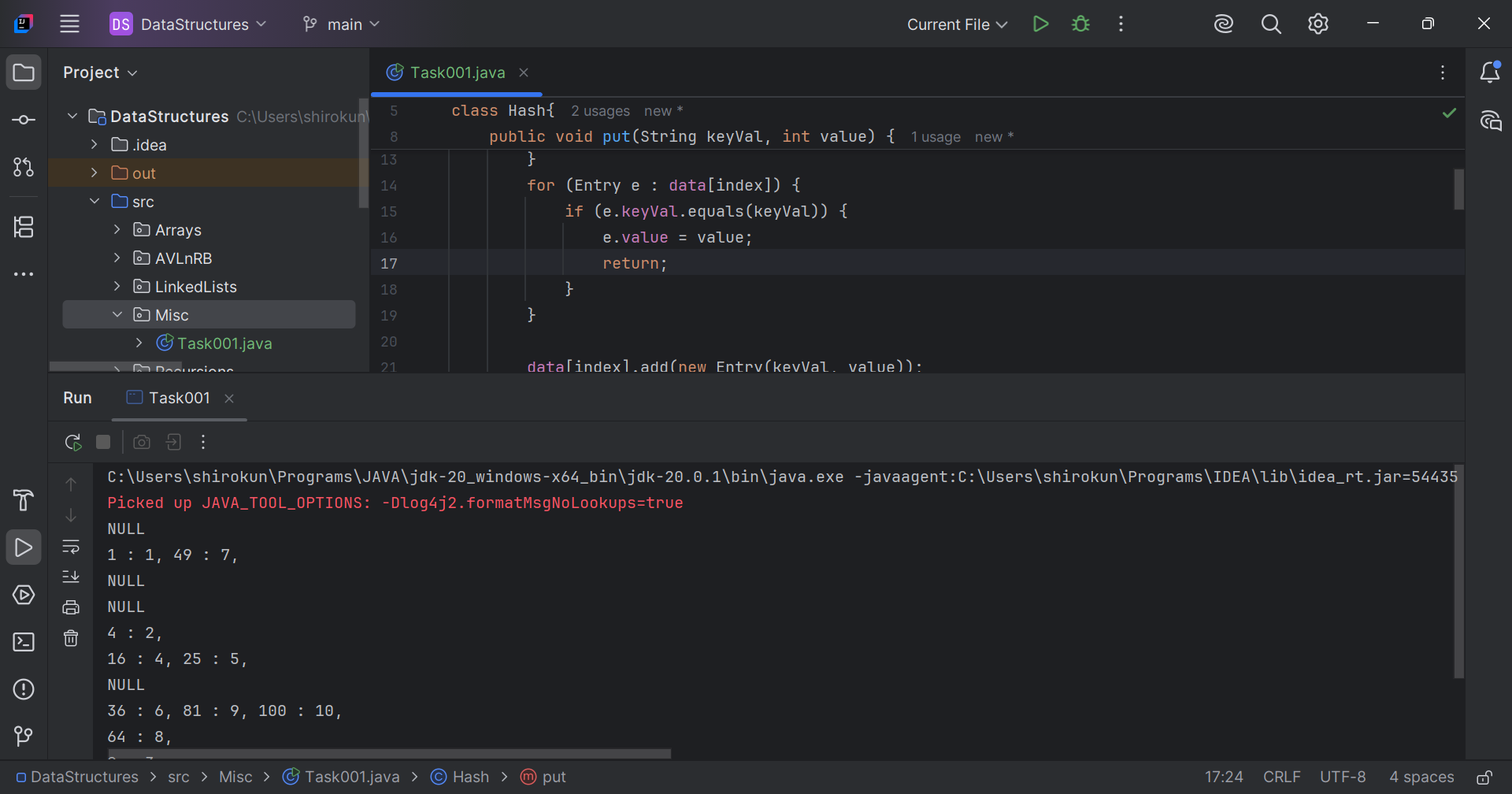
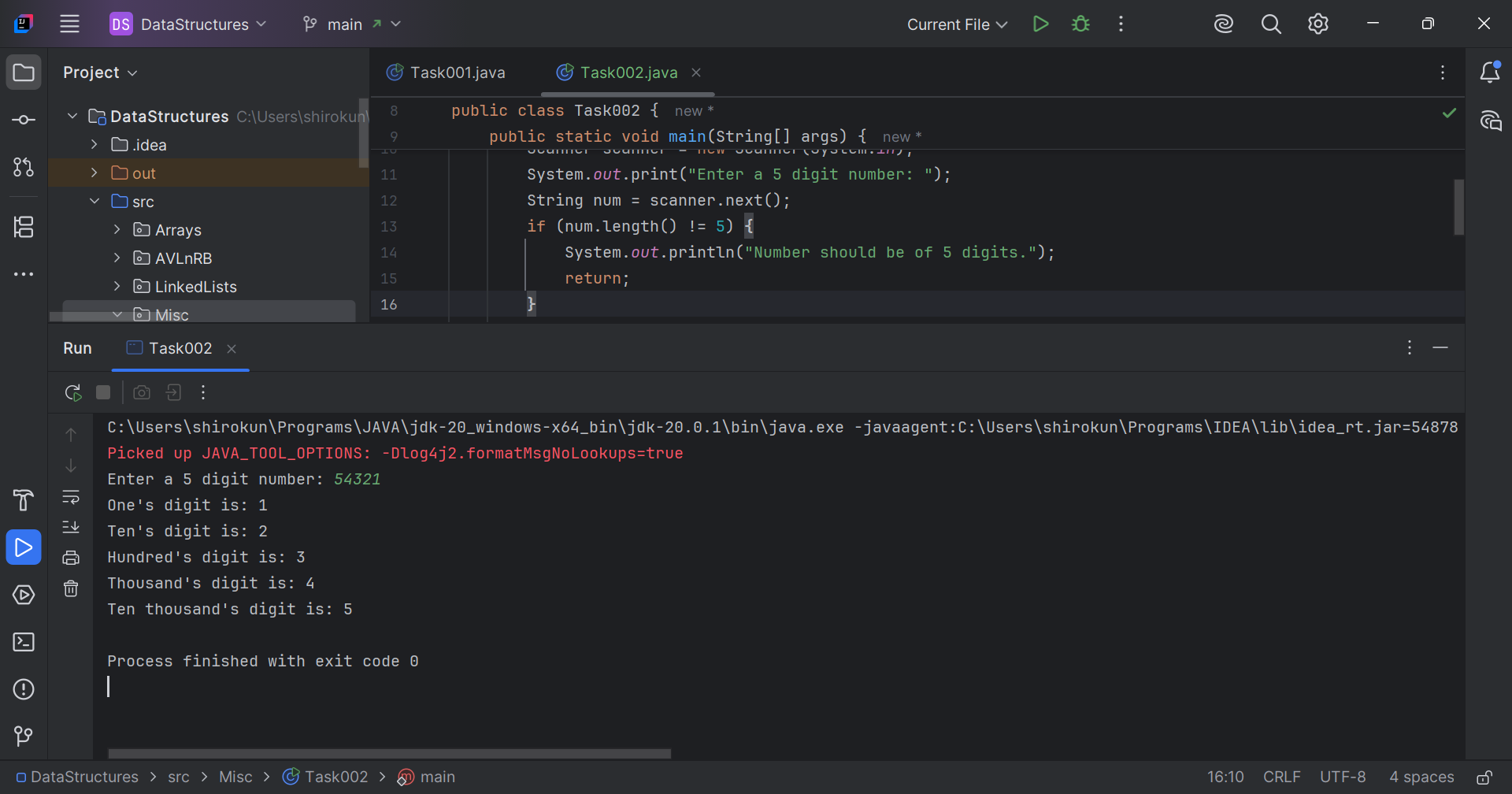
Day18 – 18/07/2025

// Task001: Hashing

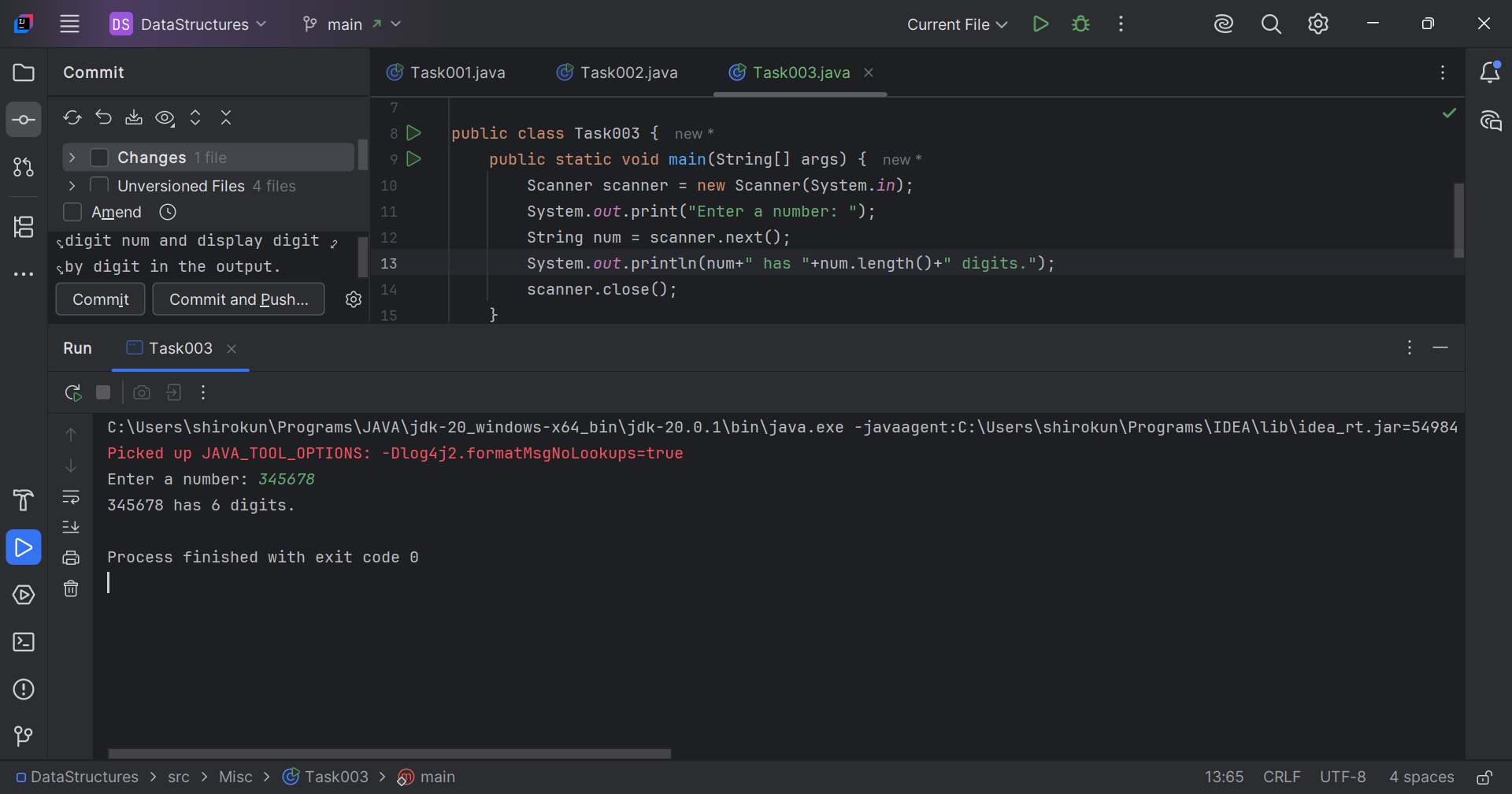
package Misc;  
import java.util.LinkedList;  
  
class Hash{  
 LinkedList<Entry>[] data = new LinkedList[10];  
 public void put(String keyVal, int value) {  
 int index = Math.*abs*(keyVal.hashCode() % data.length);  
  
 if (data[index] == null) {  
 data[index] = new LinkedList<>();  
 }  
 for (Entry e : data[index]) {  
 if (e.keyVal.equals(keyVal)) {  
 e.value = value;  
 return;  
 }  
 }  
 data[index].add(new Entry(keyVal, value));  
 }  
  
 static class Entry {  
 String keyVal;  
 int value;  
  
 Entry(String k, int v) {  
 keyVal = k;  
 value = v;  
 }  
 }  
 void printData(){  
 for (LinkedList<Entry> entries: data){  
 if (entries != null){  
 for (Entry entry: entries){  
 System.*out*.print(entry.keyVal+ ": "+entry.value+", ");  
 }  
 } else {  
 System.*out*.print("NULL");  
 }  
 System.*out*.println();  
 }  
 }  
}  
  
public class Task001 {  
 public static void main(String[] args) {  
 Hash hash = new Hash();  
 for (int i = 1; i <= 10; i++) {  
 hash.put((i\*i)+" ", i);  
 }  
 hash.printData();  
 }  
}



// Task002: Take input from the user a 5 digit num  
// and display digit by digit in the output.  
  
package Misc;  
  
import java.util.Scanner;  
  
public class Task002 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter a 5 digit number: ");  
 String num = scanner.next();  
 if (num.length() != 5) {  
 System.*out*.println("Number should be of 5 digits.");  
 return;  
 }  
 String[] places = {"Ten thousand's", "Thousand's", "Hundred's", "Ten's", "One's"};  
 for (int i = num.length()-1; i > -1 ; i--) {  
 System.*out*.println(places[i]+ " digit is: " +num.charAt(i));  
 }  
 scanner.close();  
 }  
}



// Task003: Take number from the user  
// and display the no of digit it has.  
  
package Misc;  
  
import java.util.Scanner;  
  
public class Task003 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 System.*out*.print("Enter a number: ");  
 String num = scanner.next();  
 System.*out*.println(num+" has "+num.length()+" digits.");  
 scanner.close();  
 }  
}



Q4. What are the applications of heap sort?  
Ans. Applications of Heap Sort –

* Sorting large datasets
* Priority queues
* Finding top-K elements
* Event handling
* Resource allocation
* Database query optimization
* Scientific computing
* Embedded systems

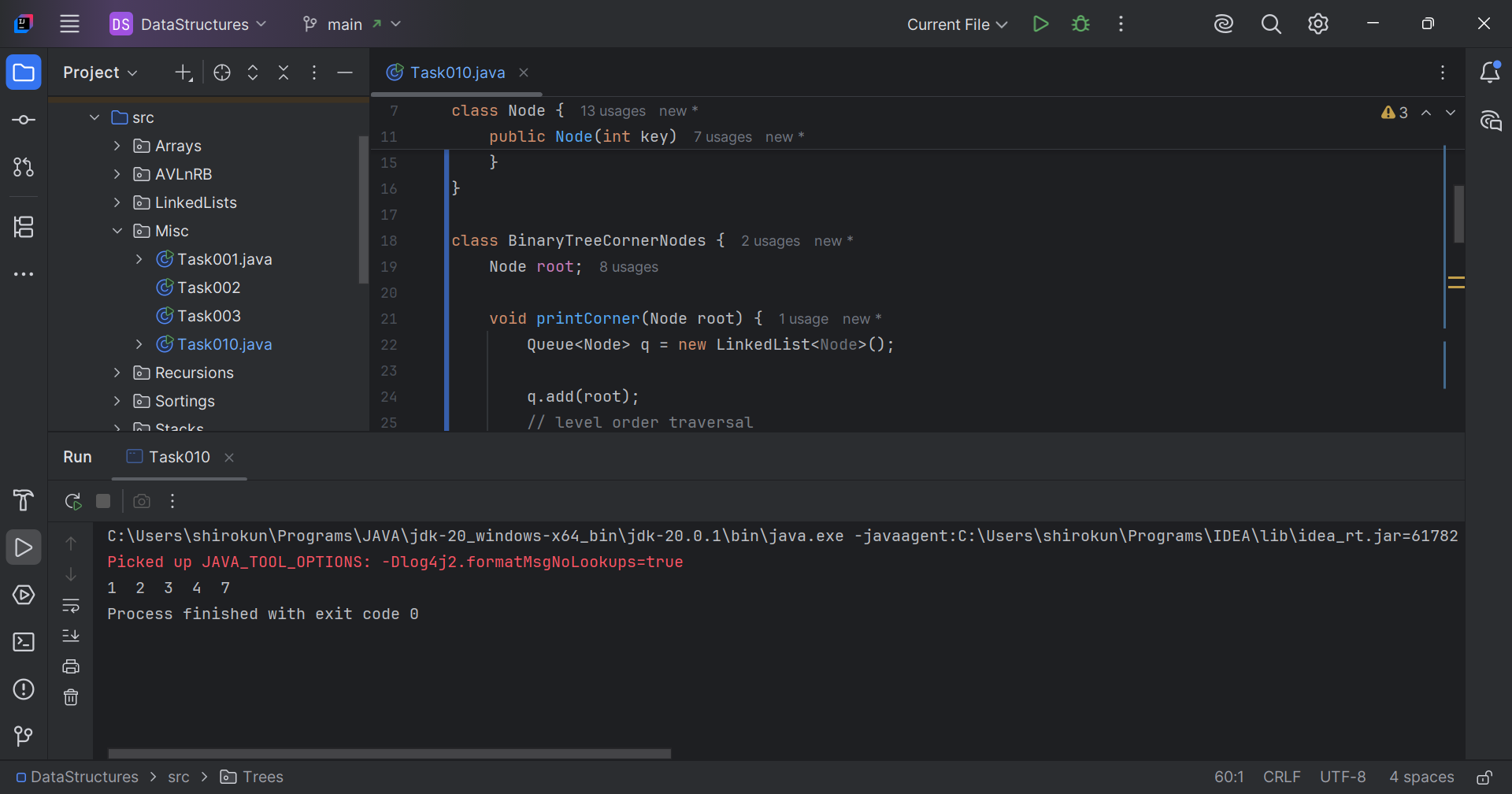
Q5. Do you find any significance change between the breadthFirstSearchRecursive() approach compared to the standard BFS?  
Ans. It achieves same result but emphasizes on recursive style using the same level-order logic with explicit queue management.

Q7. How can you say recursive functions maintain the state of each call during execution?  
Ans. The system call stack tracks local variables and return addresses for each recursive invocation.

Q6. How does heap sort work? Explain the technique.  
Ans. Heap Sort Technique –

1. Build a Heap: Convert the input array into a heap data structure, where the parent node is either larger (max heap) or smaller (min heap) than its child nodes.
2. Extract Elements: Remove the root element (maximum or minimum value) from the heap and place it at the end (or beginning) of the sorted array.
3. Heapify: Restore the heap property by heapifying the remaining elements.
4. Repeat: Repeat steps 2-3 until the entire array is sorted.

// Task010: Print the corner nodes of a binary search tree.  
  
package Misc;  
  
import java.util.\*;  
  
class Node {  
 int key;  
 Node left, right;  
  
 public Node(int key)  
 {  
 this.key = key;  
 left = right = null;  
 }  
}  
  
class BinaryTreeCornerNodes {  
 Node root;  
  
 void printCorner(Node root) {  
 Queue<Node> q = new LinkedList<Node>();  
  
 q.add(root);  
 // level order traversal  
 while (!q.isEmpty()) {  
 int n = q.size();  
 for(int i = 0 ; i < n ; i++){  
 Node temp = q.peek();  
 q.poll();// retrieve and remove the node  
  
 if(i==0 || i==n-1)  
 System.*out*.print(temp.key + " ");  
  
 if (temp.left != null)  
 q.add(temp.left);  
 if (temp.right != null)  
 q.add(temp.right);  
 }  
 }  
  
 }  
}  
  
  
public class Task010 {  
 public static void main(String[] args){  
 BinaryTreeCornerNodes tree = new BinaryTreeCornerNodes();  
 tree.root = new Node(1);  
 tree.root.left = new Node(2);  
 tree.root.right = new Node(3);  
 tree.root.left.left = new Node(4);  
 tree.root.left.right = new Node(5);  
 tree.root.right.left = new Node(6);  
 tree.root.right.right = new Node(7);  
  
 tree.printCorner(tree.root);  
 }  
}



Q15. Write algo for radix sort.  
Ans. Algorithm –

1. Find the maximum element in the input array to determine the number of digits.
2. Initialize 10 empty buckets (one for each digit from 0 to 9).
3. Iterate through each digit position (from least significant to most significant):
   1. Distribute the elements into the buckets based on the current digit.
   2. Collect the elements from the buckets back into the input array.
4. Repeat step 3 until all digit positions have been processed.

Q16. Write pseudocode for radix sort.  
Ans. Pseudocode –

* Procedure RadixSort(arr):
  + max\_element = FindMax(arr)
  + max\_digits = NumberOfDigits(max\_element)
  + For digit\_position = 0 to max\_digits - 1:
    - buckets = InitializeEmptyBuckets(10)
    - For each element in arr:
      * digit = GetDigit(element, digit\_position)
      * buckets[digit].Add(element)
    - index = 0
    - For each bucket in buckets:
      * For each element in bucket:
        + arr[index] = element
        + index = index + 1
  + Return arr