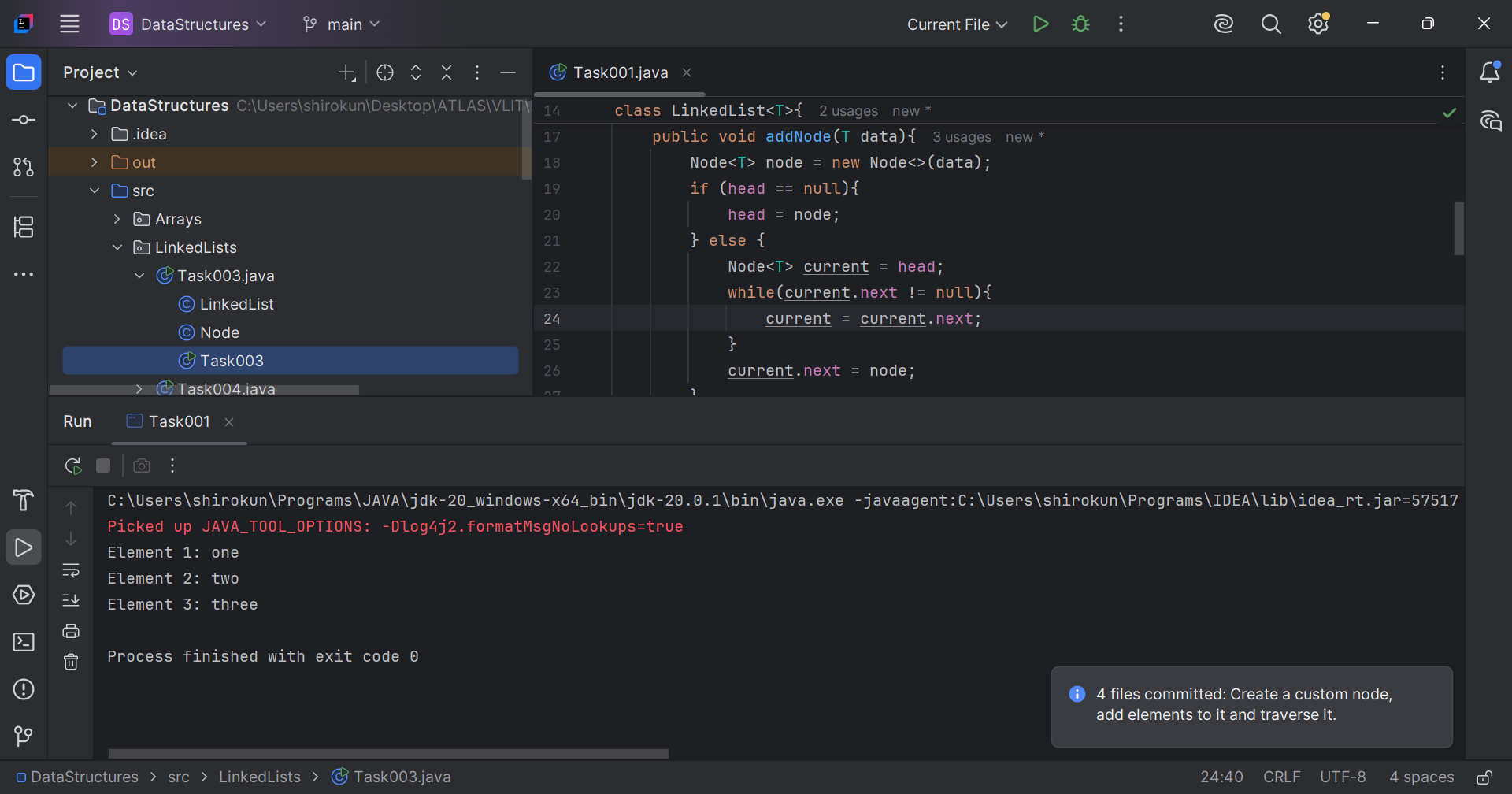
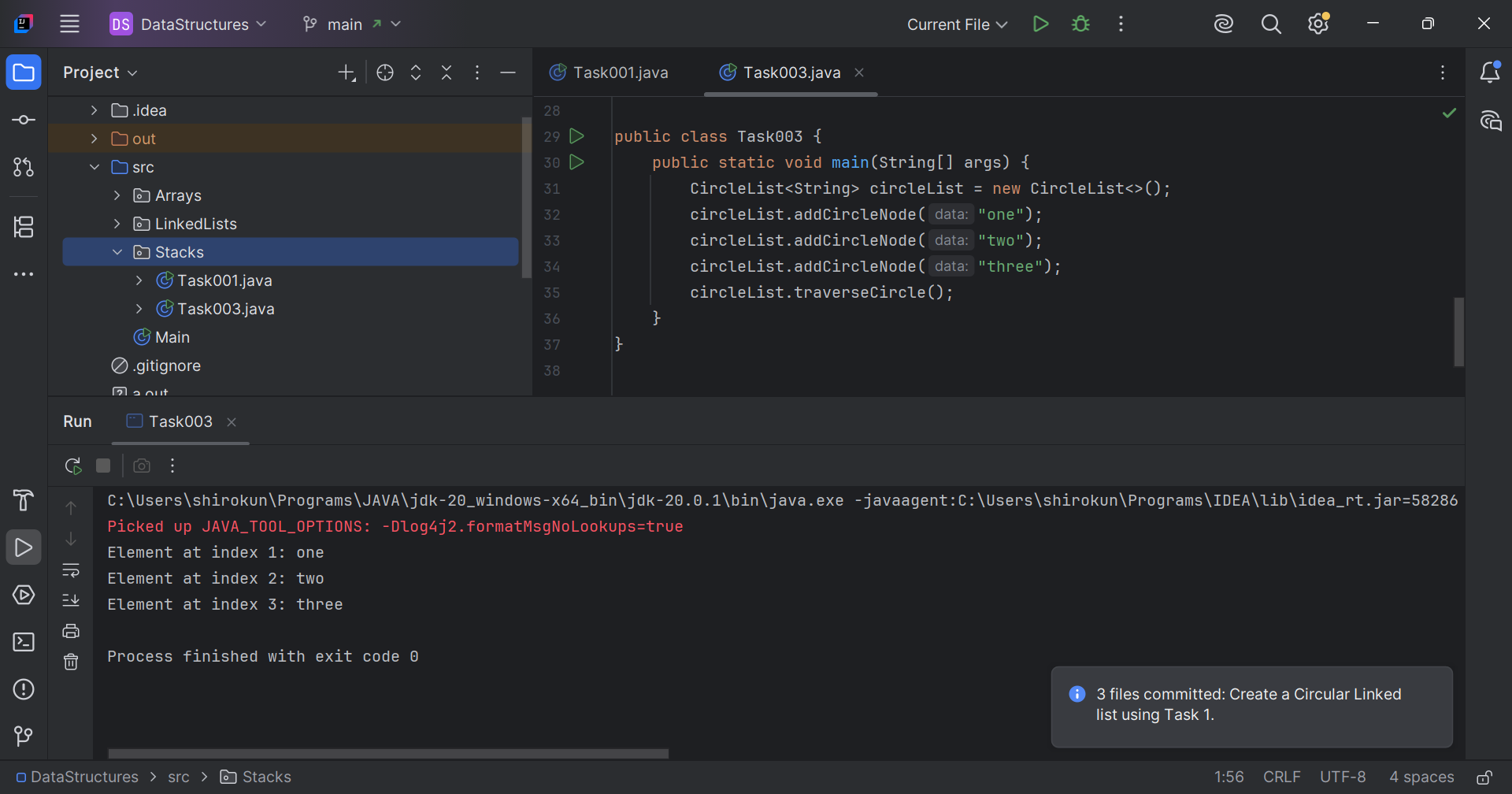
Day 14 – 04/07/2025

// Task001: Create a custom node, add elements to it and traverse it.  
  
package Stacks;  
  
class Node<T>{  
 T data;  
 Node<T> next;  
 public Node(T data){  
 this.data = data;  
 this.next = null;  
 }  
}  
  
class LinkedList<T>{  
 Node<T> head;  
  
 public void addNode(T data){  
 Node<T> node = new Node<>(data);  
 if (head == null){  
 head = node;  
 } else {  
 Node<T> current = head;  
 while(current.next != null){  
 current = current.next;  
 }  
 current.next = node;  
 }  
 }  
  
 public void traverse(){  
 int index = 1;  
 Node<T> current = head;  
 while (current != null){  
 System.*out*.println("Element "+index+": "+current.data);  
 current = current.next;  
 index++;  
 }  
 }  
}  
  
public class Task001 {  
 public static void main(String[] args) {  
 LinkedList<String> list = new LinkedList<>();  
 list.addNode("one");  
 list.addNode("two");  
 list.addNode("three");  
 list.traverse();  
 }  
}



Q2. What do you understand by traversing a linked list?  
Ans. Traverse means travel. So, in a linked list traverse means travelling across the linked list element by element. Traversing helps us to search, delete or print elements.

// Task003: Create a Circular Linked list using Task 1.  
  
package Stacks;  
  
class CircleList<T> extends LinkedList<T>{  
 public void addCircleNode(T data) {  
 Node<T> node = new Node<>(data);  
  
 if(head == null){  
 head = node;  
 } else {  
 tail.next = node;  
 }  
 tail = node;  
 tail.next = head;  
 }  
  
 public void traverseCircle(){  
 Node<T> current = head;  
 int index=1;  
 do {  
 System.*out*.println("Element at index "+index+": "+current.data);  
 current = current.next;  
 index++;  
 }while (current != head);  
 }  
}  
  
public class Task003 {  
 public static void main(String[] args) {  
 CircleList<String> circleList = new CircleList<>();  
 circleList.addCircleNode("one");  
 circleList.addCircleNode("two");  
 circleList.addCircleNode("three");  
 circleList.traverseCircle();  
 }  
}



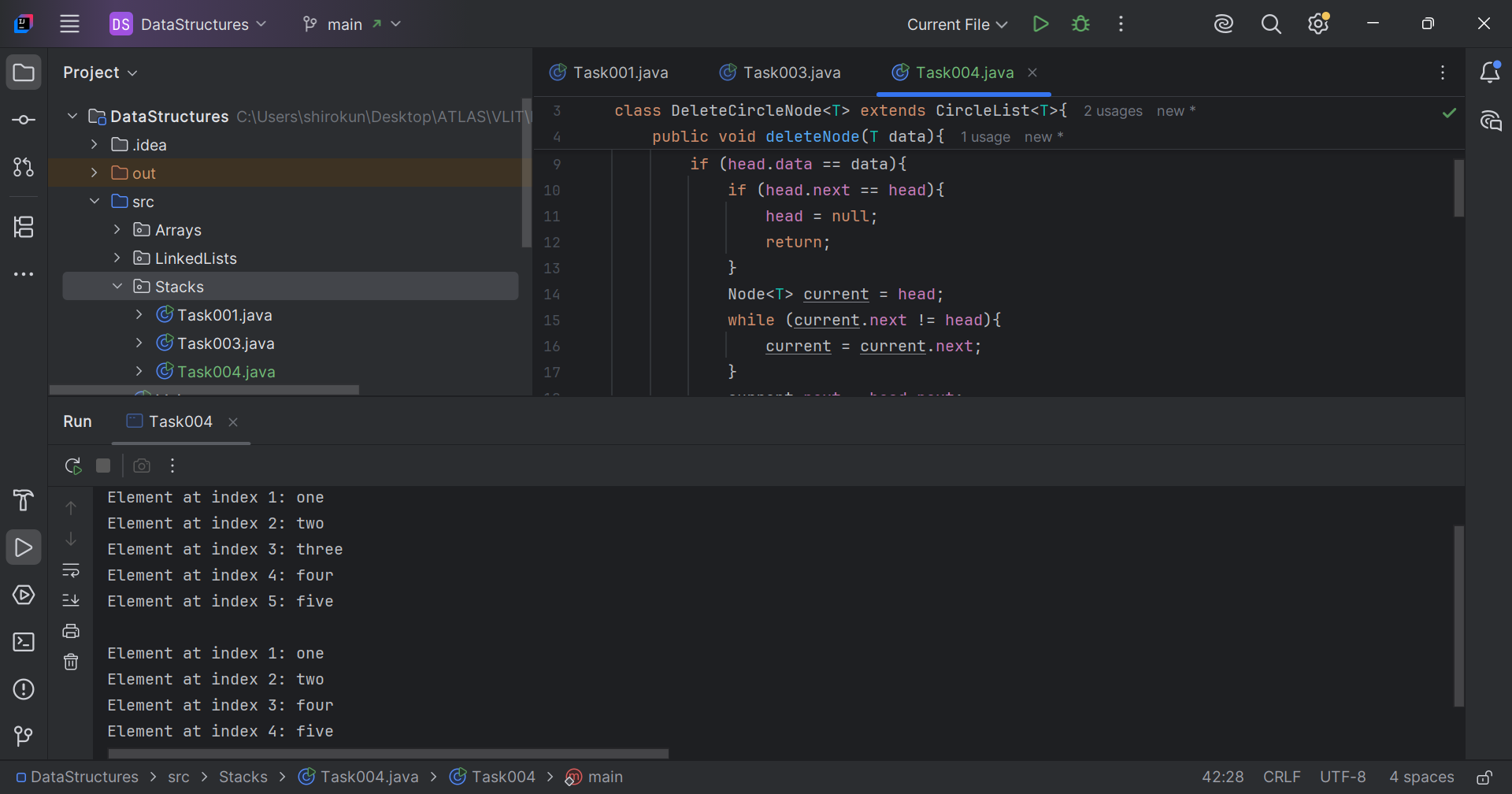
// Task004: Delete a node in the circular linked list.  
  
package Stacks;

class DeleteCircleNode<T> extends CircleList<T>{  
 public void deleteNode(T data){  
 if(head == null){  
 System.*out*.println("List is empty.");  
 return;  
 }

if (head.data == data){

if (head.next == head){  
 head = null;  
 return;  
 }

Node<T> current = head;  
 while (current.next != head){  
 current = current.next;  
 }  
 current.next = head.next;  
 head = head.next;  
 return;  
 }  
 Node<T> current = head;  
 while (current.next != head){  
 if(current.next.data == data){  
 current.next = current.next.next;  
 return;  
 }  
 current = current.next;  
 }  
 }  
}  
  
public class Task004 {  
 public static void main(String[] args) {  
 DeleteCircleNode<String> list = new DeleteCircleNode<>();  
 list.addCircleNode("one");  
 list.addCircleNode("two");  
 list.addCircleNode("three");  
 list.addCircleNode("four");  
 list.addCircleNode("five");  
 list.traverseCircle();  
 System.*out*.println();  
 list.deleteNode("three");  
 list.traverseCircle();  
 }  
}



// Task005: Create a stack, pop the element and print it.  
  
package Stacks;  
  
import java.util.Stack;  
  
public class Task005 {  
 public static void main(String[] args) {  
 Stack<String> stack = new Stack<>();  
 stack.push("1");  
 stack.push("2");  
 stack.push("3");  
 stack.push("4");  
 stack.push("5");  
 System.*out*.println(stack);  
 System.*out*.println("Top element: "+stack.peek());  
 System.*out*.println("Popped element: "+stack.pop());  
 System.*out*.println("Pushed: "+stack.push("end."));  
 System.*out*.println(stack);  
 }  
}

