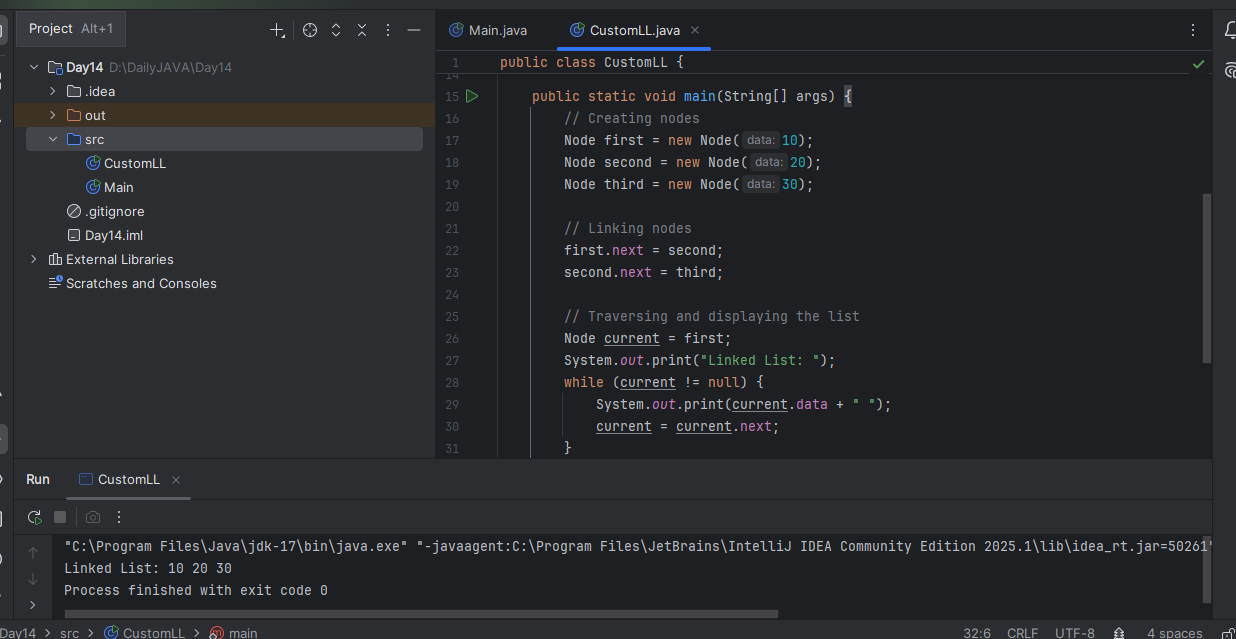
**ID: bvarshsp**

**Name: Varsha SP**

**Day 14 – 2nd July 2025**

**Task 1:**

Create a custom node, add elements to it and traverse it.



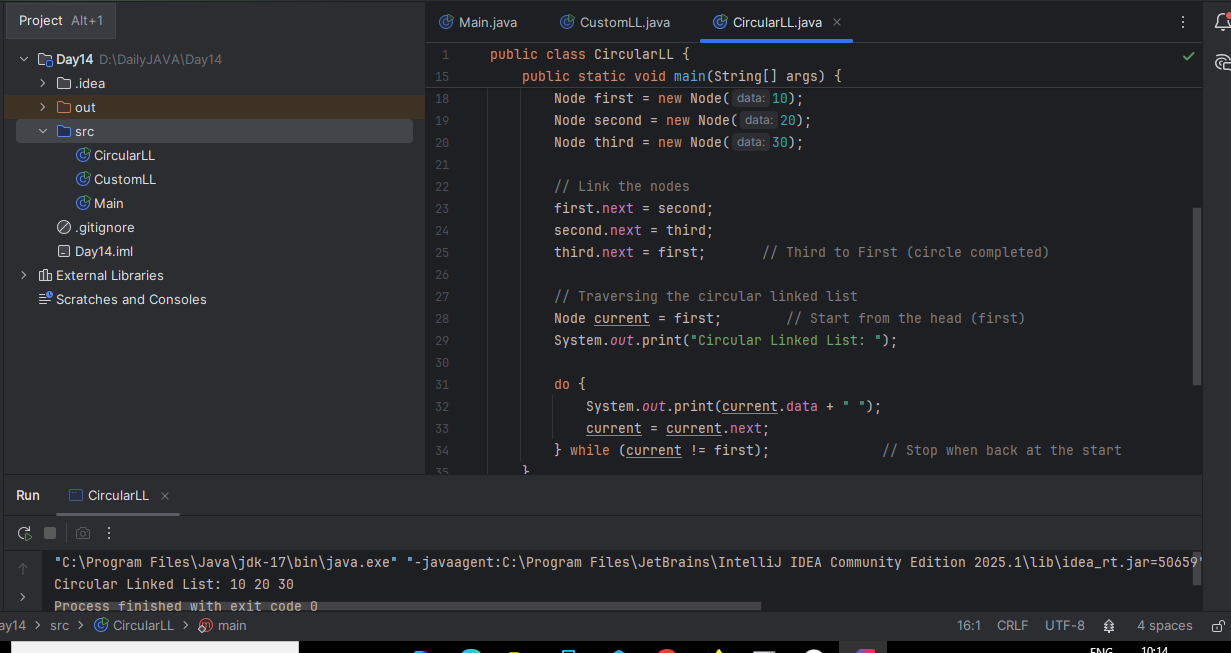
**Task 2**

What do you understand by traversing elements in a linked list?

Answer: Traversing a linked list means visiting each node one by one, starting from the head (first node) and following the next pointers, until we reach the end of the list (where next is null).

**Task 3:**

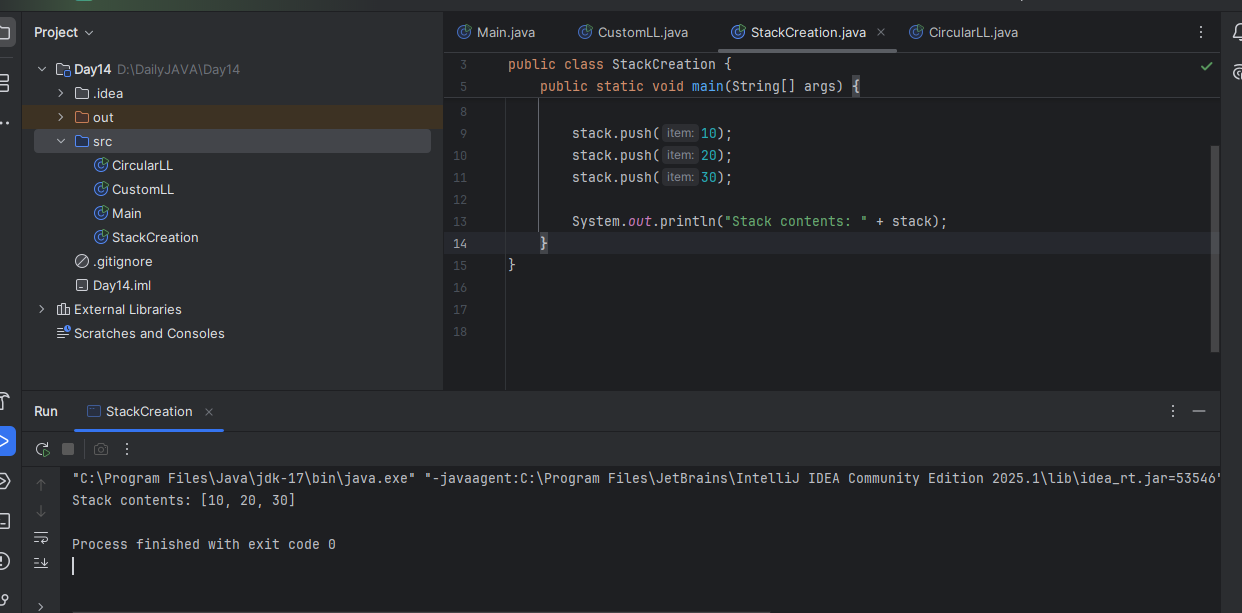
Create a Circular Linked list using Task 1 Singly linked list.



**Task 4:**

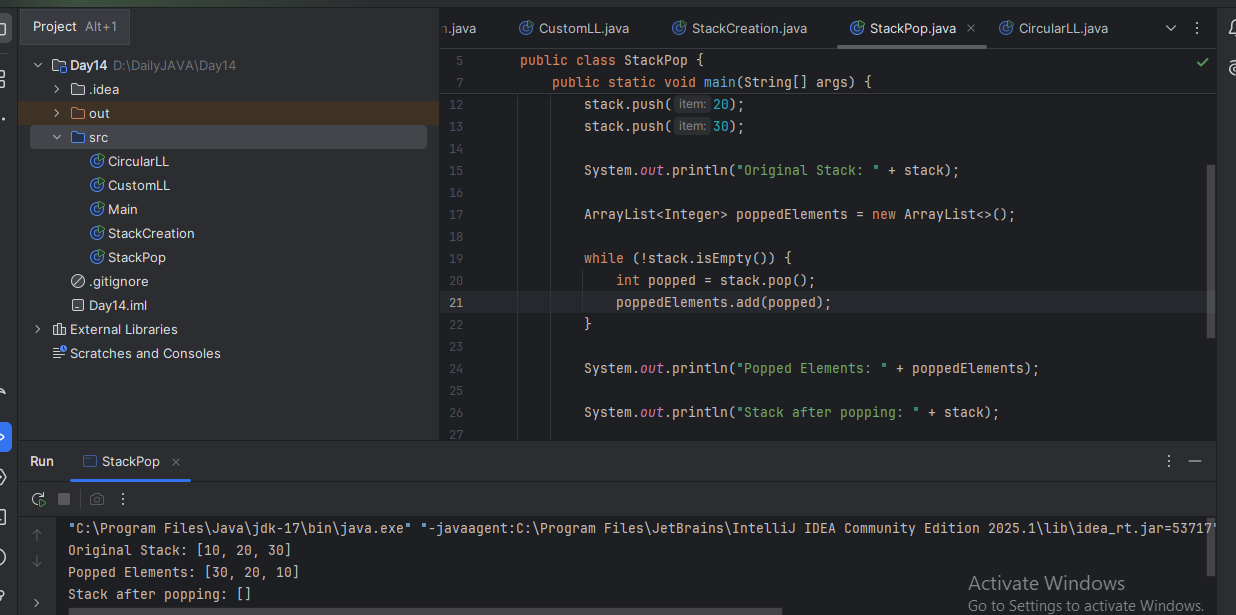
Stacks

Create a code to implement a stack



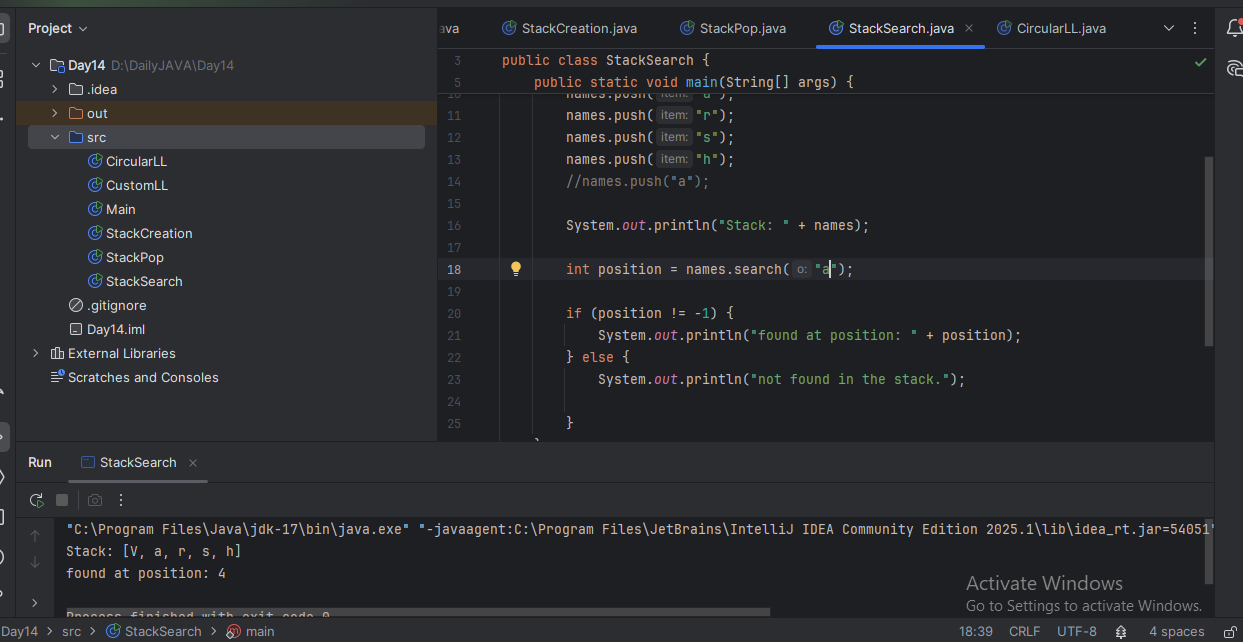
**Task 5:**

Create a stack and pop the element also print the popped element.



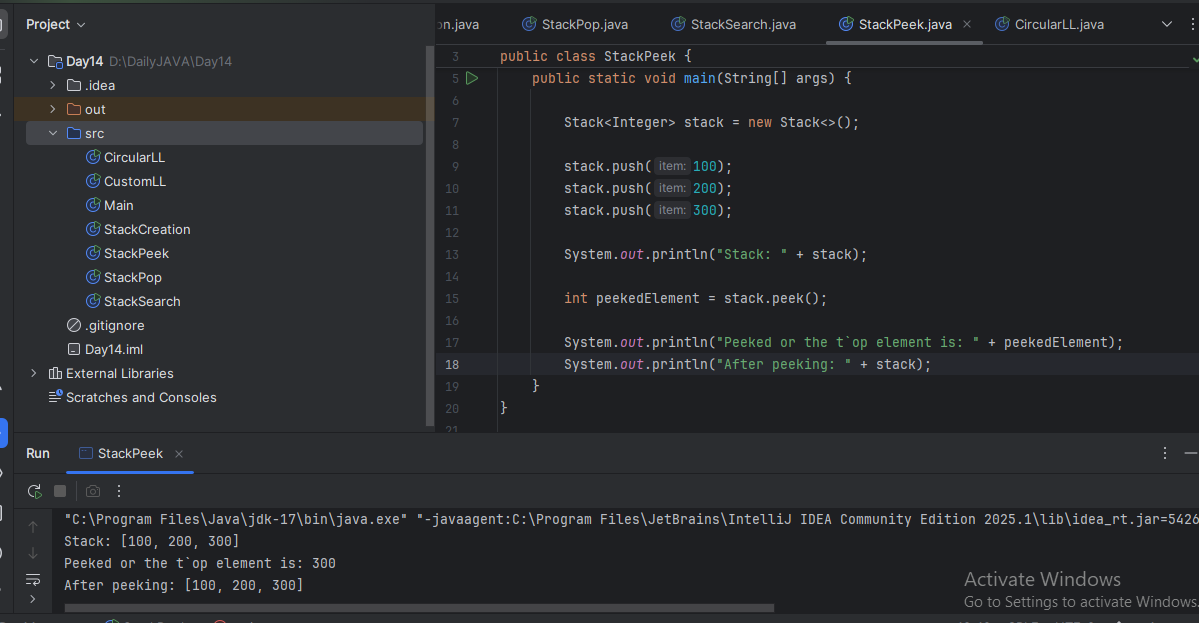
**Task 6:**

Find an element in the stack and display the position



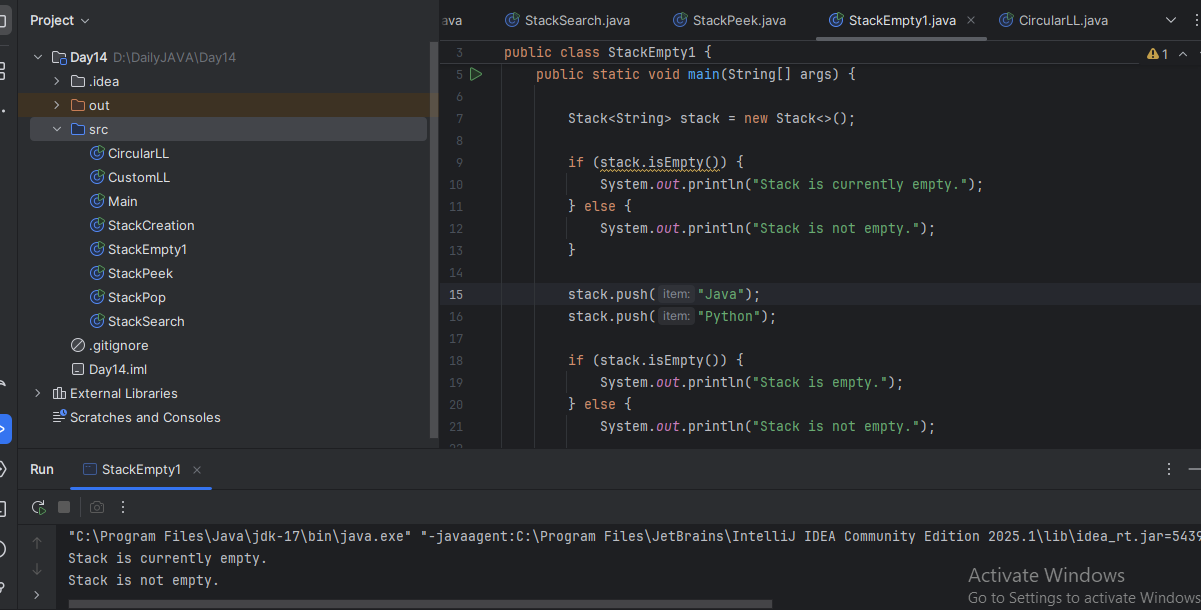
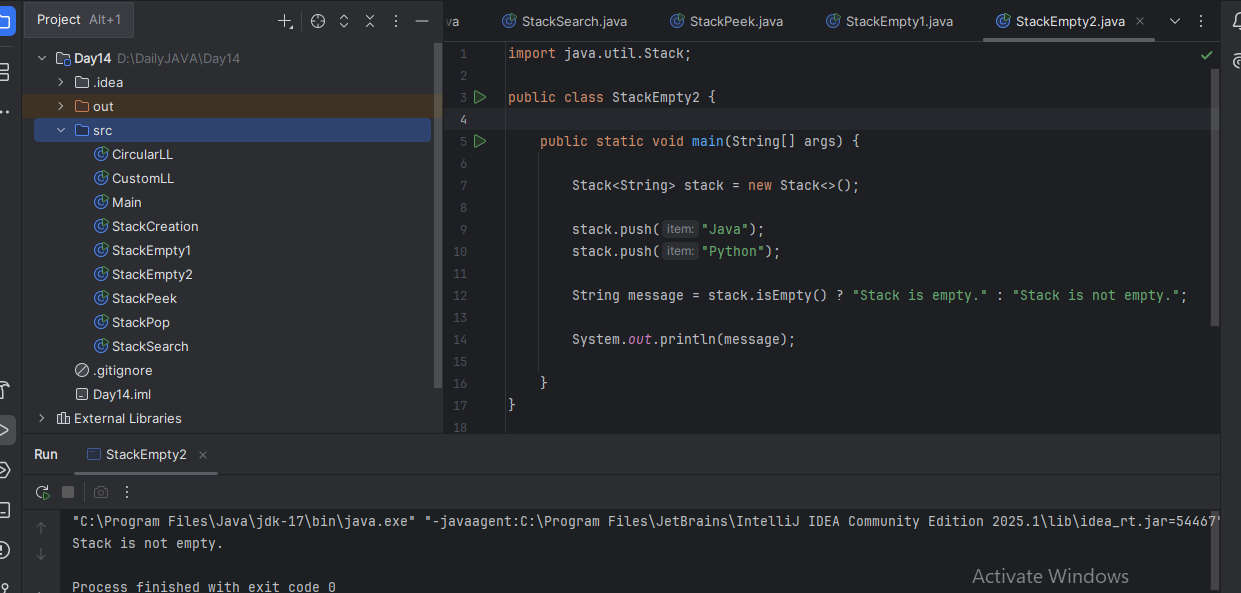
**Task 7:**

Peek the element and print it



**Task 8:**

Check if the stack is empty or not?



**Task 9:**

What are the methods of the stack class?

Answer:

push(E item) – Adds an item to the top of the stack.

pop() – Removes and returns the top element from the stack.

peek() – Returns the top element without removing it.

isEmpty() – Checks if the stack has no elements.

search(Object o) – Returns 1-based position from top if the object is found; otherwise -1.

size() – Returns the number of elements in the stack.

clear() – Removes all elements from the stack.

contains(Object o) – Checks if the stack contains the specified object.

iterator() – Returns an iterator to loop through the stack elements.

get(int index) – Returns the element at the specified position (inherited from Vector).

set(int index, E element) – Replaces the element at the given position.

add(E element) – Adds an element to the end of the stack (not commonly used with Stack).

remove(Object o) – Removes the first occurrence of the specified object.

remove(int index) – Removes the element at the specified index.

indexOf(Object o) – Returns the index of the first occurrence of the object.

lastIndexOf(Object o) – Returns the index of the last occurrence of the object.

toArray() – Converts the stack to an array.

**Task 10:**

What are the common operations in Queues?

Answer:

add(E e) -> Inserts the element at the rear (throws exception if full)

offer(E e) -> Inserts the element at the rear (returns false if full)

remove() -> Removes and returns the front element (throws exception if empty)

poll() -> Removes and returns the front element (returns null if empty)

element() -> Returns the front element without removing (throws exception if empty)

peek() -> Returns the front element without removing (returns null if empty)

**Task 11:**

Wap to create  a queue with custom methods

Is empty ()

Is full()

Enque

Deque

Peek

display()

public class CustomizedQueue {

static class Node {

int data;

Node next;

Node(int data) {

this.data = data;

this.next = null;}

}

Node front, rear;

boolean isEmpty() {

return front == null;

}

void enqueue(int data) {

Node newNode = new Node(data);

if (isEmpty()) {

front = rear = newNode;

} else {

rear.next = newNode;

rear = newNode;

}

System.*out*.println("Enqueued: " + data);

}

void dequeue() {

if (isEmpty()) {

System.*out*.println("Queue is empty. Cannot dequeue.");

return;

}

System.*out*.println("Dequeued: " + front.data);

front = front.next;

if (front == null) rear = null;

}

void peek() {

if (isEmpty()) {

System.*out*.println("Queue is empty.");

return;

}

System.*out*.println("Front element: " + front.data);

}

void display() {

if (isEmpty()) {

System.*out*.println("Queue is empty.");

return;

}

Node current = front;

System.*out*.print("Queue: ");

while (current != null) {

System.*out*.print(current.data + " ");current = current.next;

}

System.*out*.println();

}

public static void main(String[] args) {

CustomizedQueue queue = new CustomizedQueue();

queue.enqueue(10);

queue.enqueue(20);

queue.enqueue(30);

queue.display();

queue.peek();

queue.dequeue();

queue.display();

queue.enqueue(40);

queue.enqueue(50);

queue.display();

}

}

