**ID: bvarshsp**

**Name: Varsha SP**

**Day 13 – 26th June 2025**

**Task 001**

import java.util.LinkedList;

public class Task001\_DS\_LinkedList {

    public static void main(String[] args) {

        LinkedList<String> fruits = new LinkedList<>();

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.addFirst("Orange");

        fruits.addLast("Grapes");

        System.out.println("First Element: " + fruits.getFirst());

        System.out.println("Last Element: " + fruits.getLast());

        fruits.removeFirst();

        fruits.removeLast();

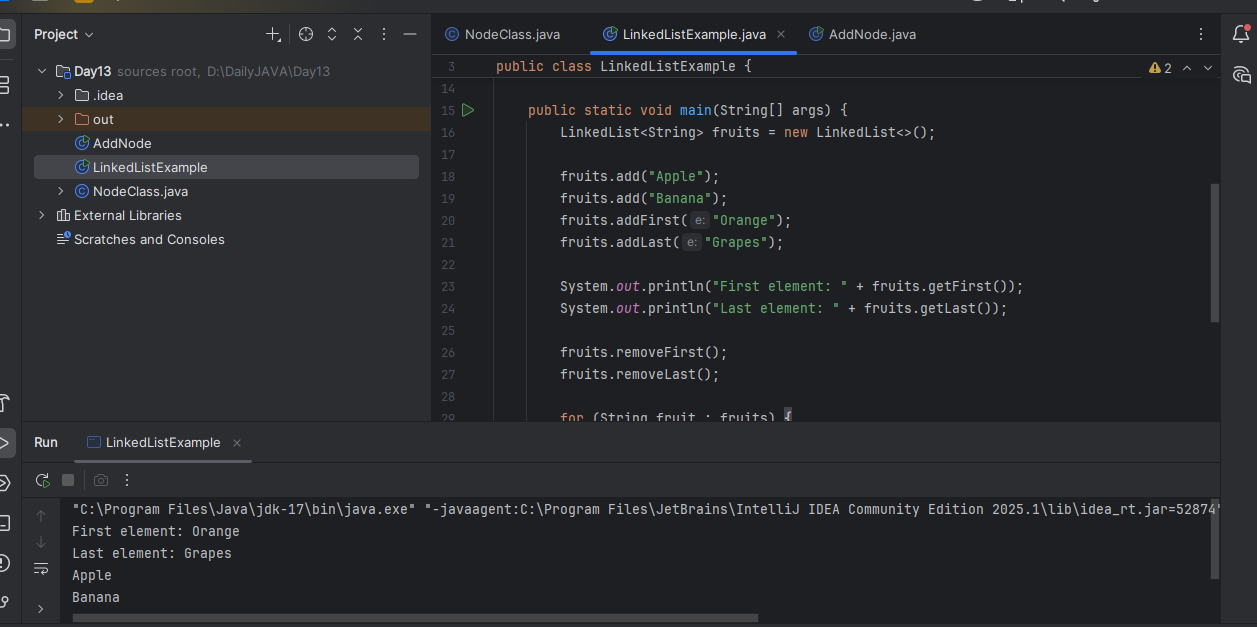
        for (String fruit : fruits) {

            System.out.println(fruit);

        }

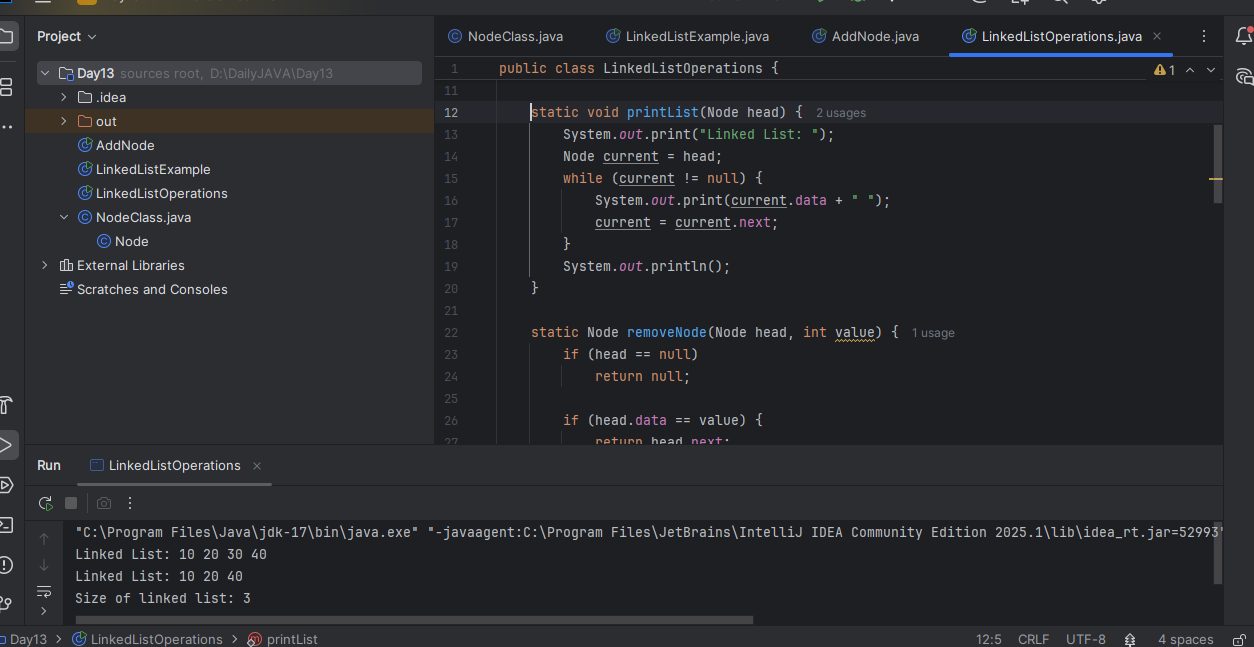
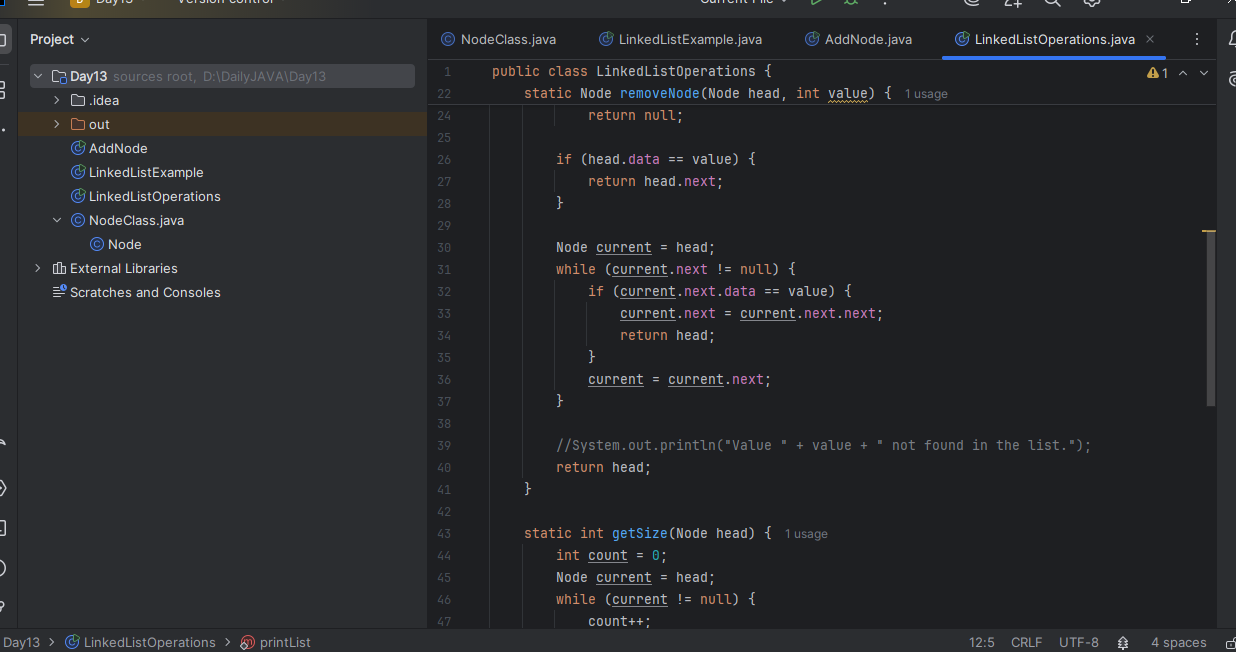
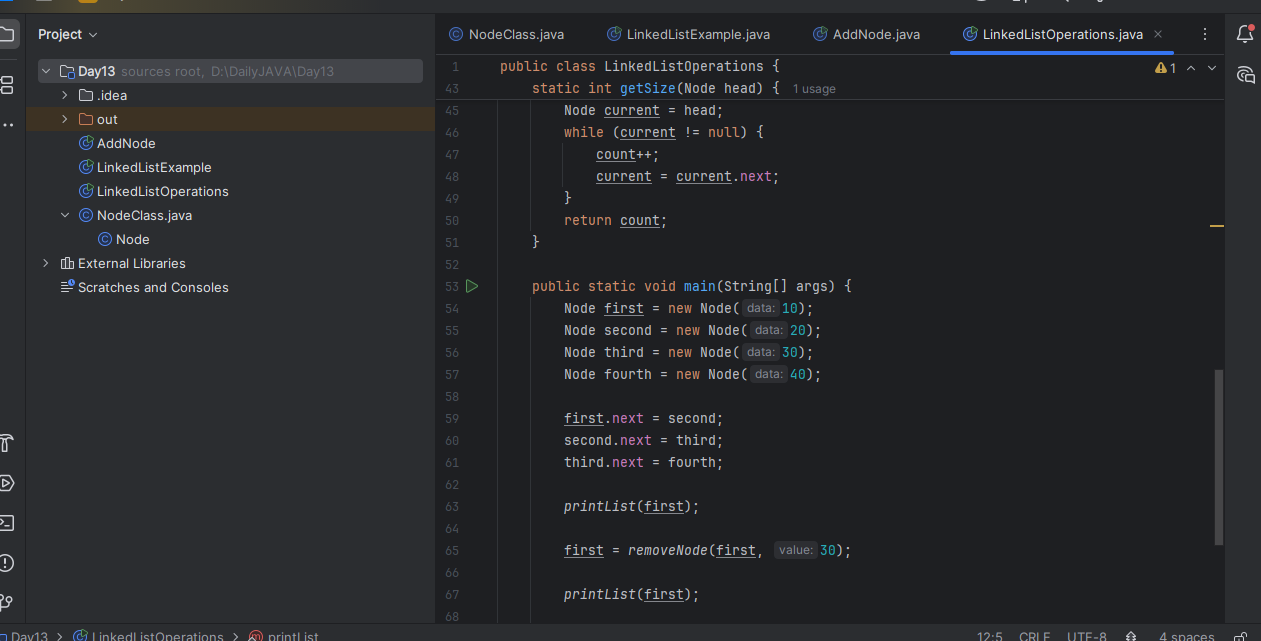
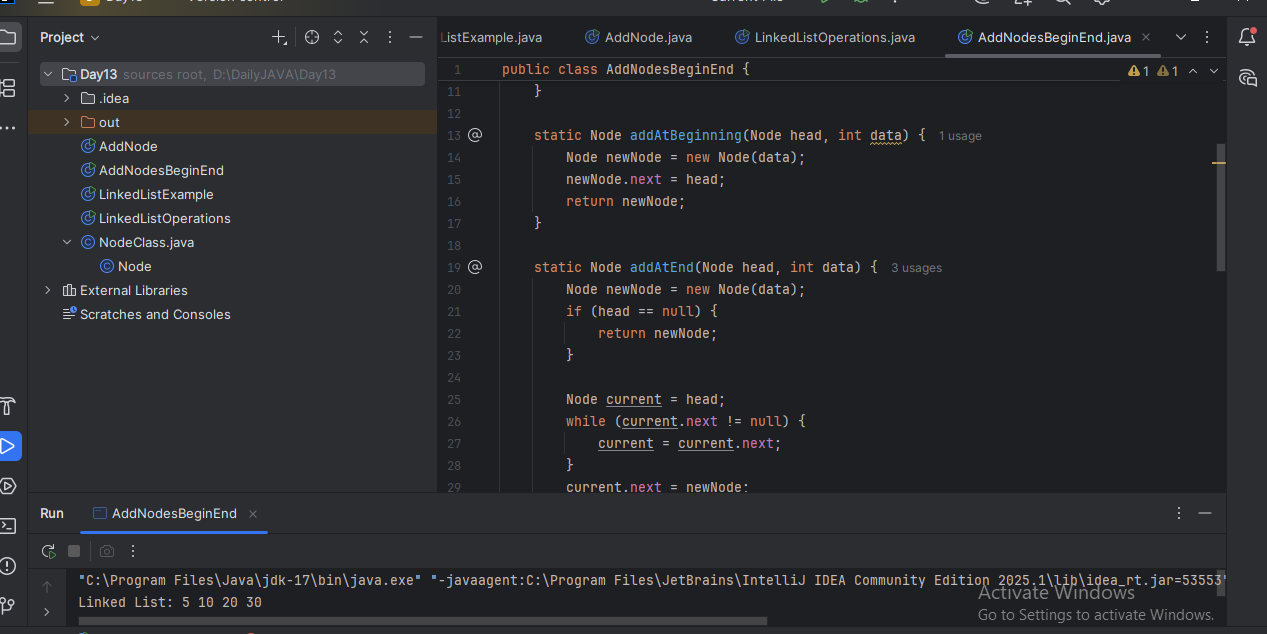
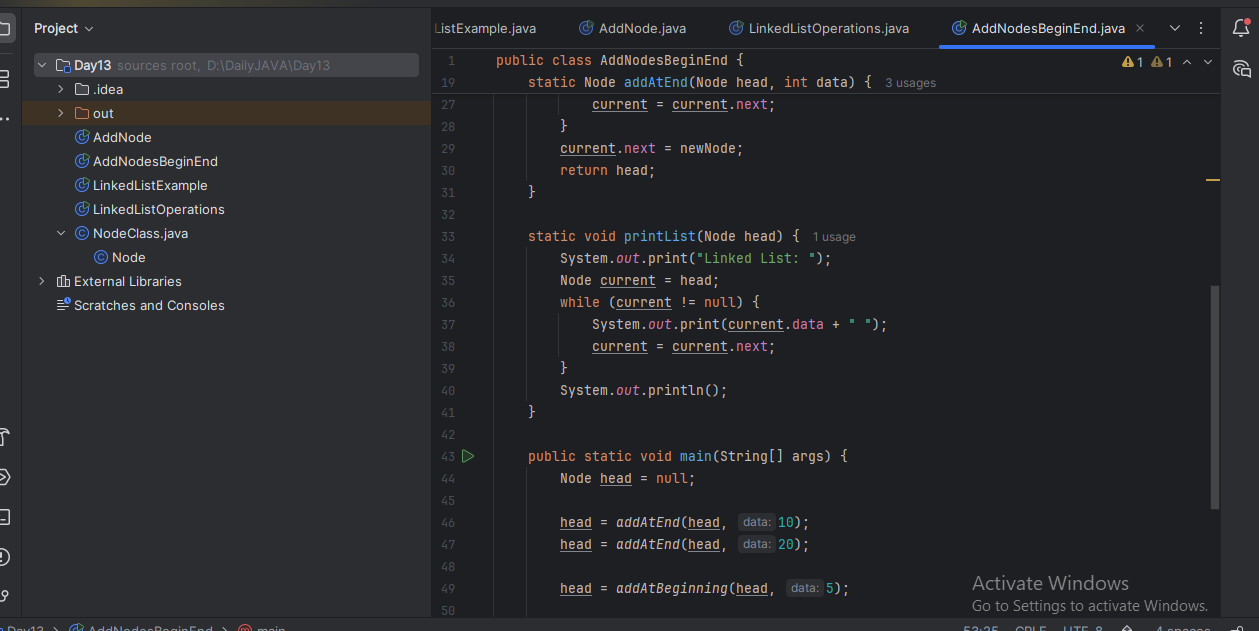
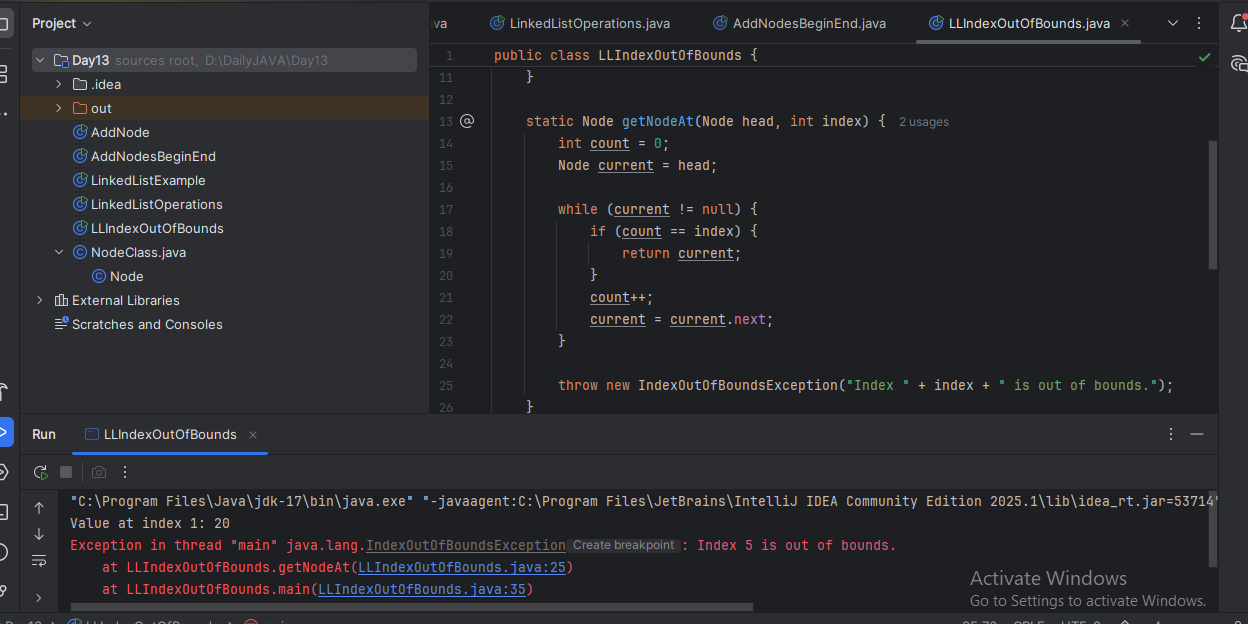
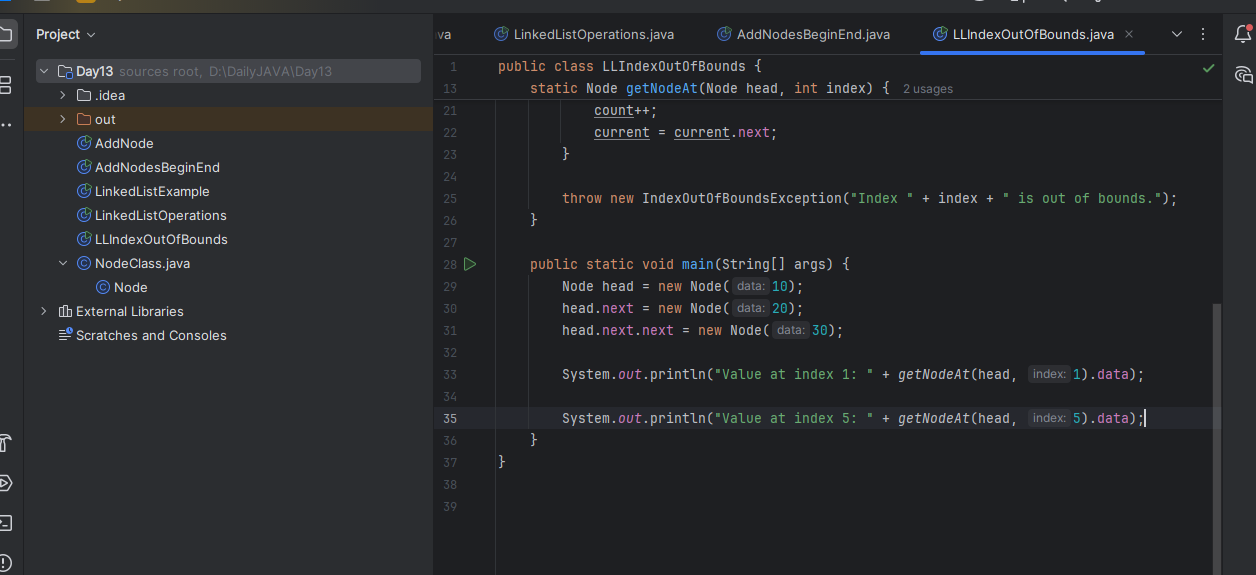
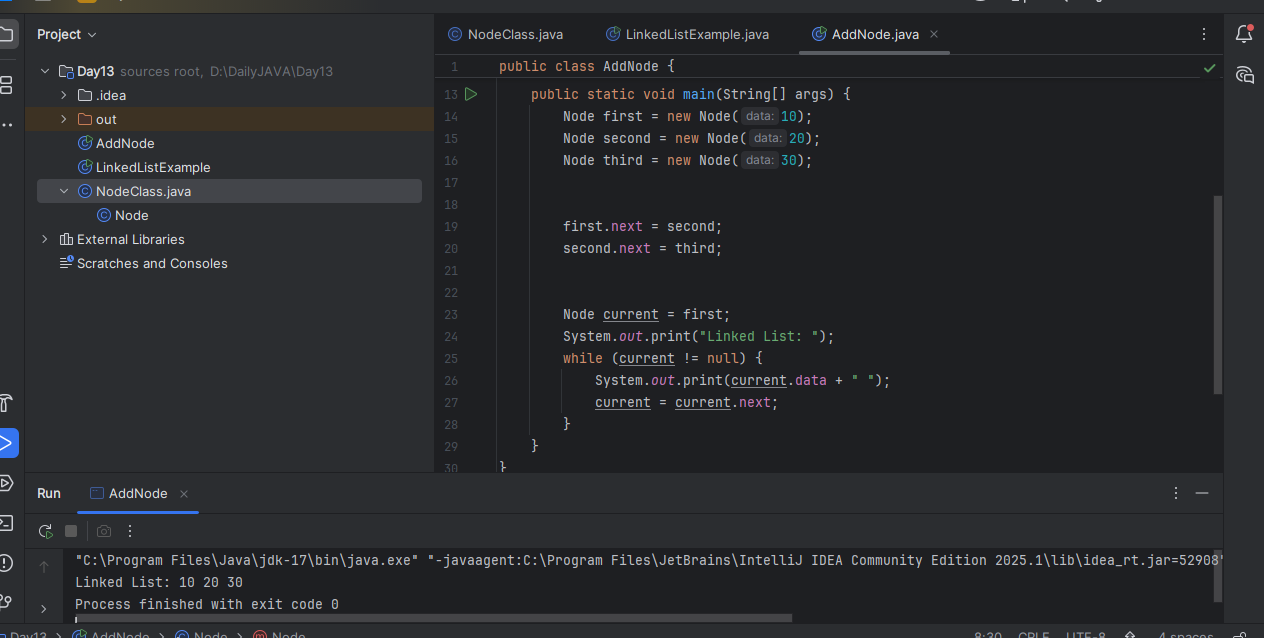
    }

}



**Task 002**

Try to create a node and add a value to it.



**Task 3**

List down the methods of linked lists.

Answer:

**Add Methods:**

add(E e)

add(int index, E element)

addFirst(E e)

addLast(E e)

offer(E e)

offerFirst(E e)

offerLast(E e)

**Remove Methods:**

remove()

remove(int index)

remove(Object o)

removeFirst()

removeLast()

poll()

pollFirst()

pollLast()

clear()

**Access Methods:**

get(int index)

getFirst()

getLast()

peek()

peekFirst()

peekLast()

**Search/Check Methods:**

contains(Object o)

indexOf(Object o)

lastIndexOf(Object o)

isEmpty()

size()

**Iteration Methods:**

iterator()

descendingIterator()

listIterator()

**Other Utility Methods:**

toArray()

toString()

clone()

equals(Object o)

hashCode()

stream()

parallelStream()

spliterator()

**Task 4:**

What are the operations of data structures?

Answer:

1. Insertion – Add a new element to the data structure.

2. Deletion – Remove an existing element from the structure.

3. Traversal – Visit and process each element one by one.

4. Searching – Find whether an element exists or locate its position.

5. Sorting – Arrange elements in a specific order (ascending/descending).

6. Access (Retrieval) – Get the value of an element at a given index or position.

7. Update (Modification) – Change the value of an existing element.

8. Merging – Combine two or more data structures into one.

9. Splitting – Divide a data structure into two or more parts.

10. Resizing – Expand or shrink the capacity of a dynamic structure.

11. Push – Add an element to the top of a stack.

12. Pop – Remove the top element from a stack.

13. Enqueue – Add an element to the rear of a queue.

14. Dequeue – Remove an element from the front of a queue.

15. InsertLeft – Insert an element at the left end of a deque.

16. InsertRight – Insert an element at the right end of a deque.

17. Peek – View the top/front element without removing it.

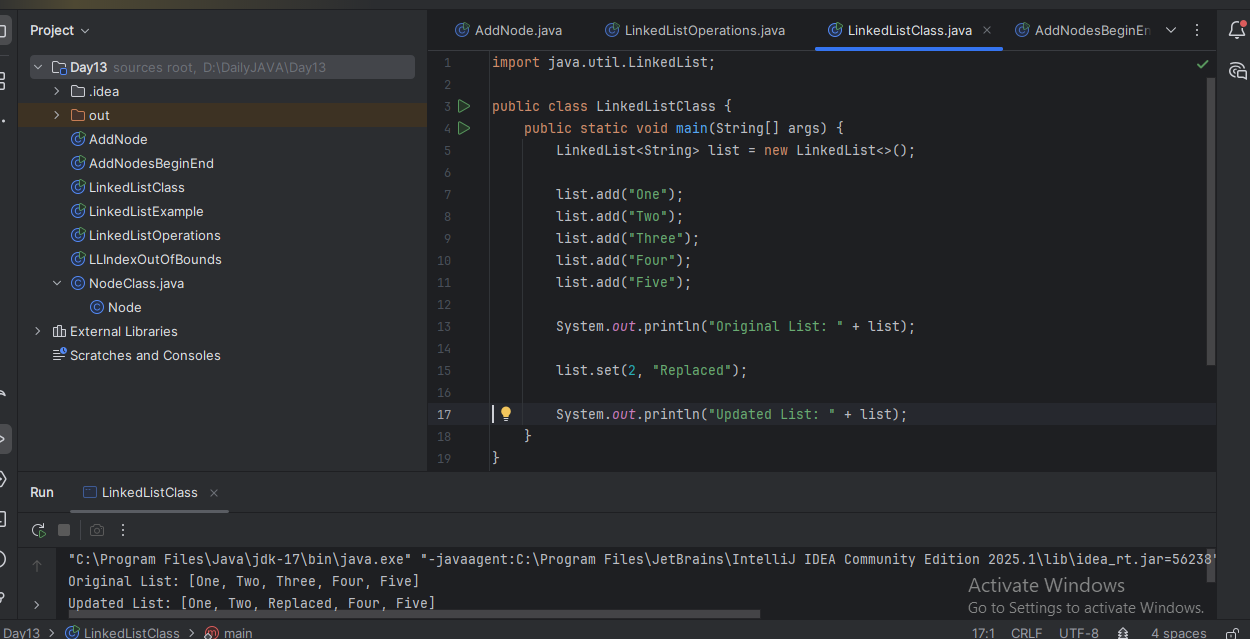
18. isEmpty – Check if the data structure has no elements.

19. isFull – Check if the structure has reached its capacity (for fixed-size).

20. Size – Return the number of elements currently stored.

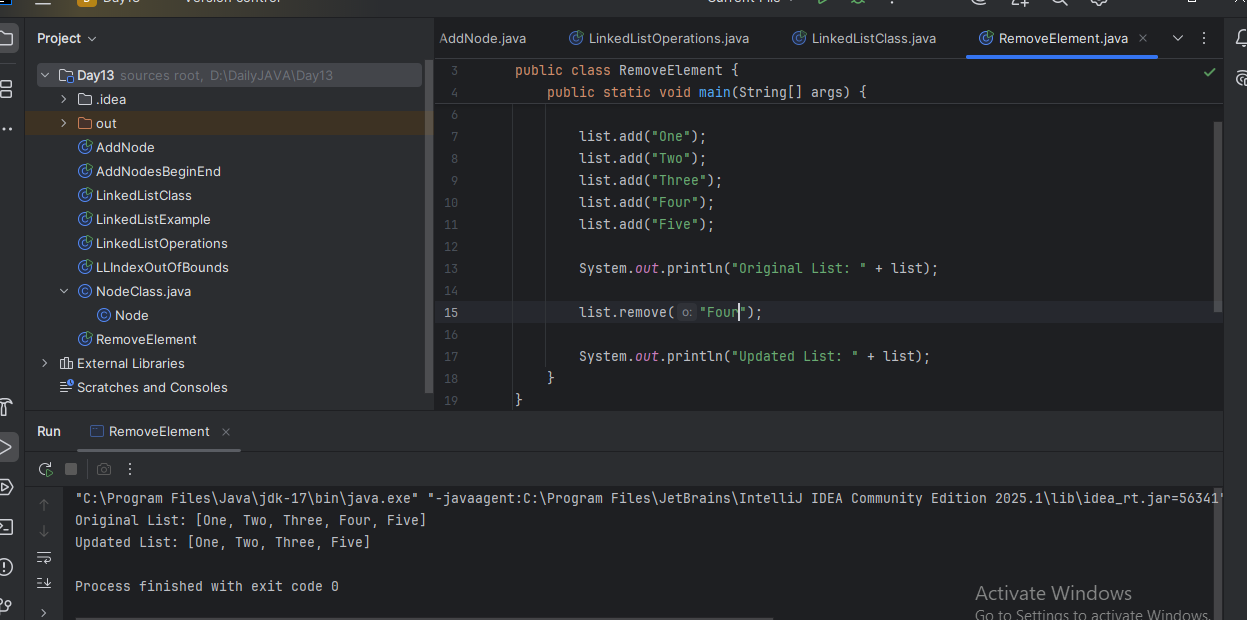
**Task 5:**

Wap to create a linked list add 5 elements to it and replace 3 rd element with different value.



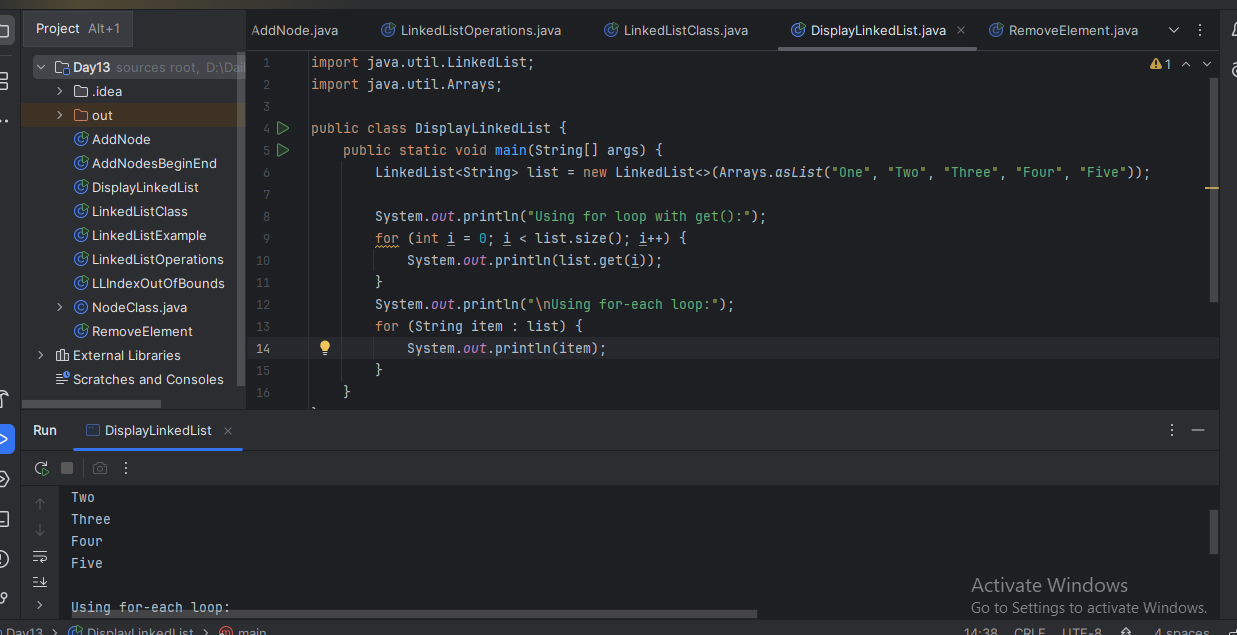
**Task 6:**

Wap to create a linked list to add 5 elements and remove any element and display



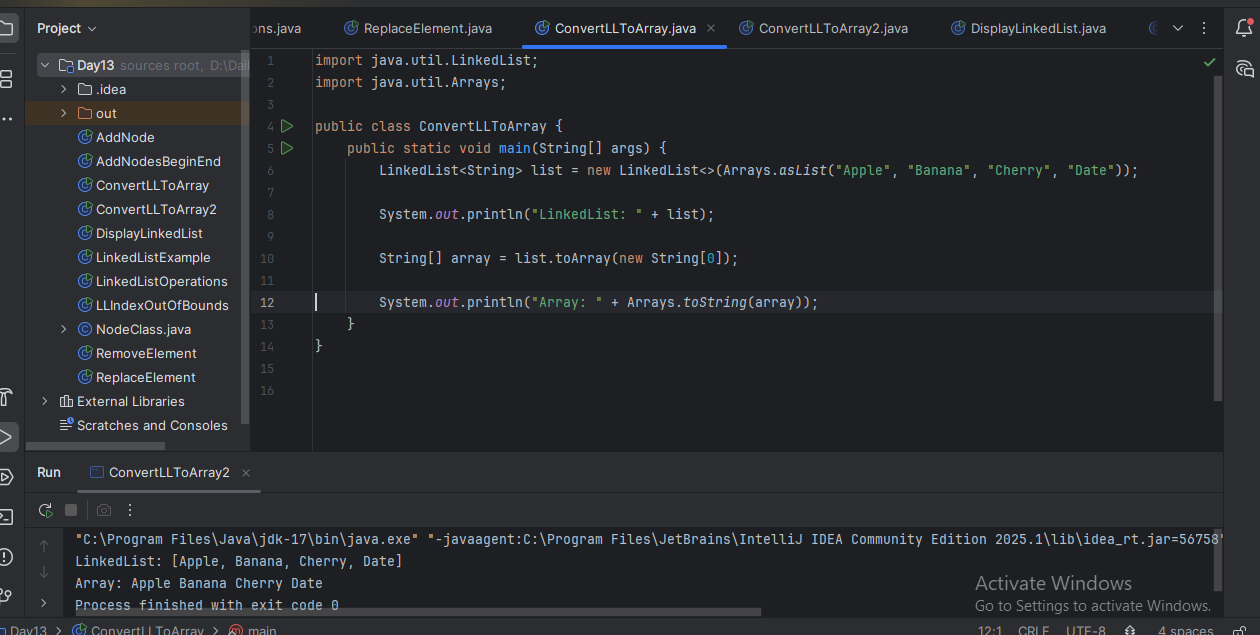
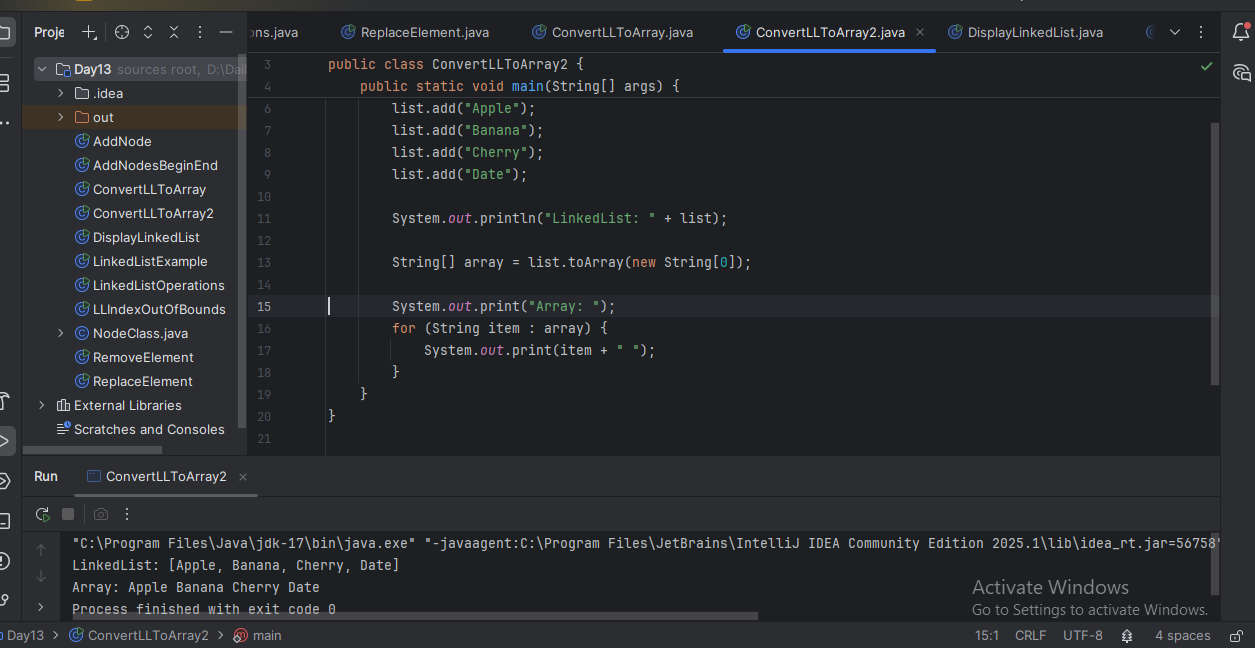
**Task 7:**

Wap to create a linked list to add 5 elements and display the list using for (use get() ) and for each loops



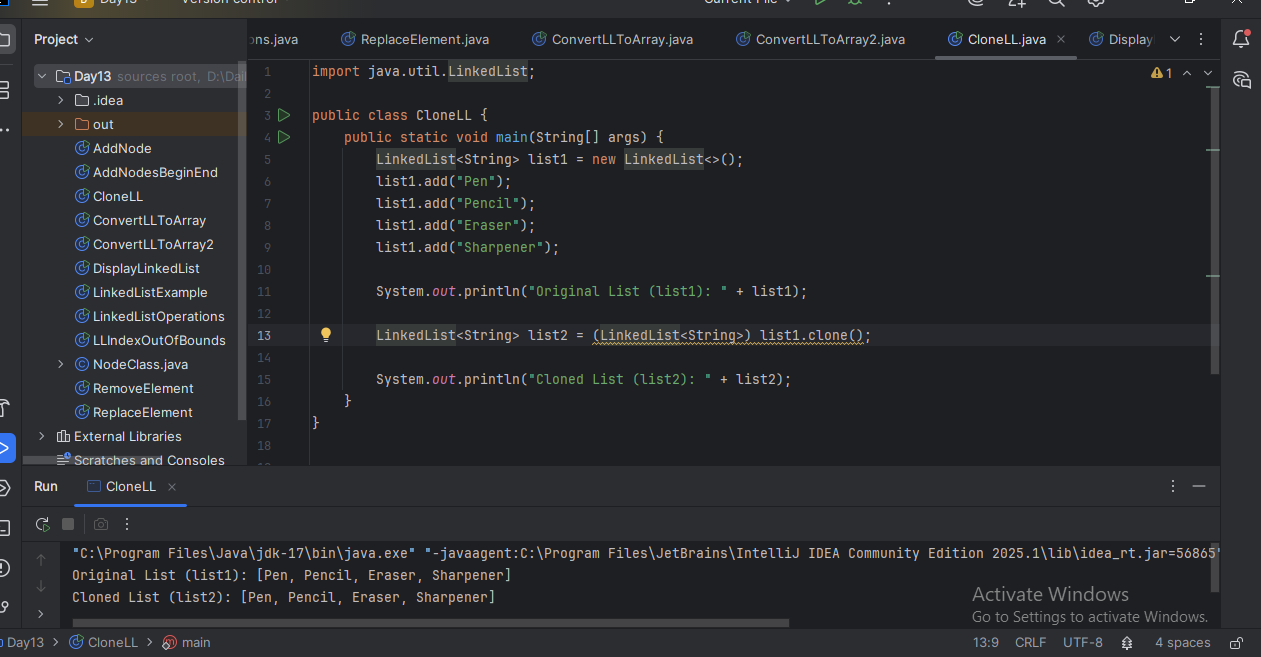
**Task 8:**

Create a linked list and few items and convert it into an array.



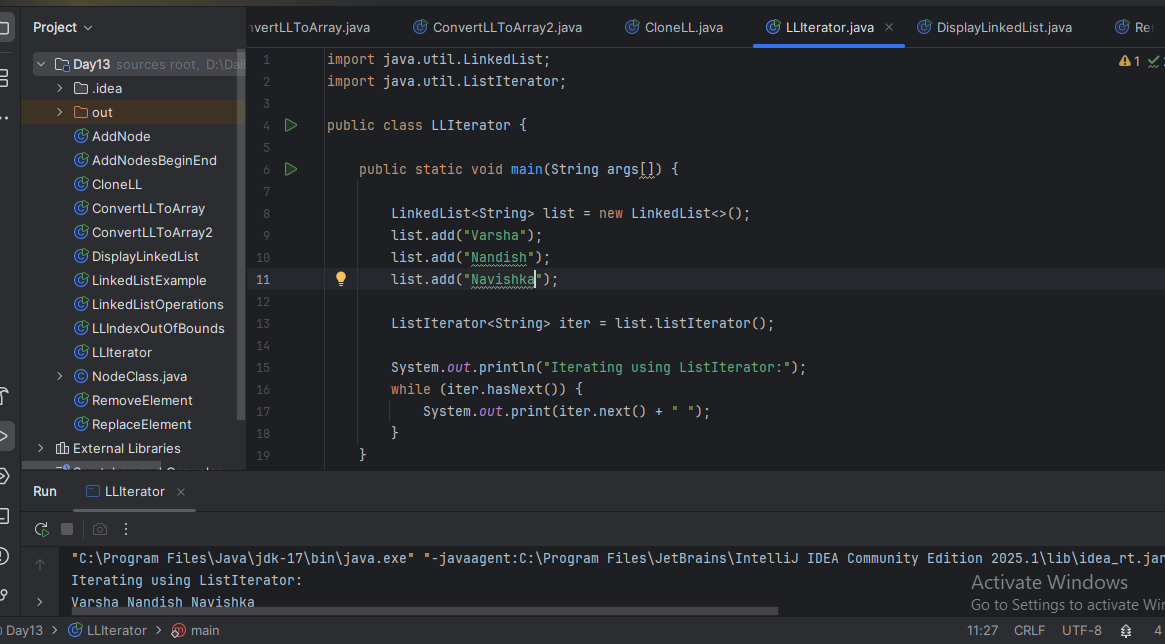
**Task 9:**

Create a linked list add few items and clone the 1st linked list with the 2nd linked list



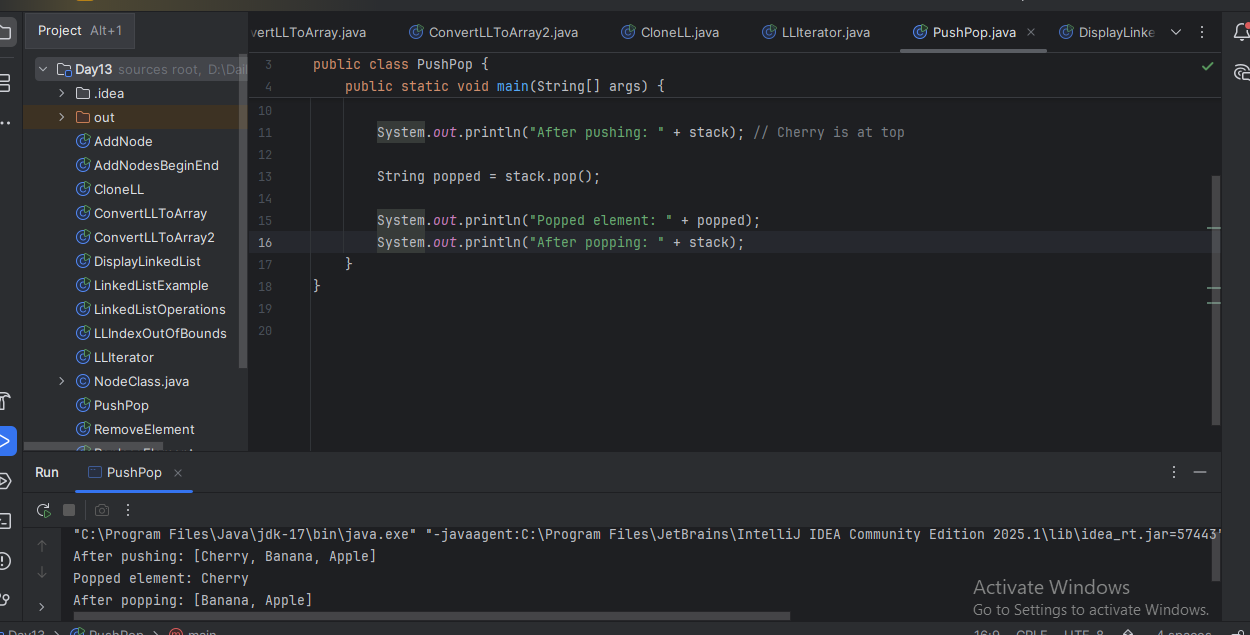
**Task 10:**

Create a linked list and iterate the values using ListIterator class in util package



**Task 11:**

Create a linked list and use push and pop methods.



**Task 12:**

Difference between Iterator and splitIterator

Answer:

Iterator is used to sequentially traverse elements in a collection, one by one.

Spliterator (Split + Iterator) is used to traverse and also split the collection, making it suitable for parallel processing.

Iterator: hasNext() next() remove()

Spliterator: tryAdvance() forEachRemaining() trySplit()

Iterator: Cannot split the collection.

Spliterator: Can split the collection for parallel processing using trySplit().

We can use Iterator for simple sequential iteration over a list, set, or map.

We can use Spliterator when working with Streams, especially for parallelStream() to improve performance on large datasets.

Examples:

Iterator:

Iterator<String> it = list.iterator();

while (it.hasNext()) {

System.out.println(it.next());

}

Spliterator:

Spliterator<String> sp = list.spliterator();

sp.forEachRemaining(System.out::println);

**Task 13:**

Below is the code for Split iterator… run it to see the output..

Can you it to sout()... and see ..

import java.util.\*;

public class Task0013\_DS\_Linkedlist\_SplitIterator {

    public static void main(String[] args) {

        LinkedList<String> lobj = new LinkedList<>();

        lobj.add("Prasunamba");

        lobj.add("Meher");

        lobj.add(".MK");

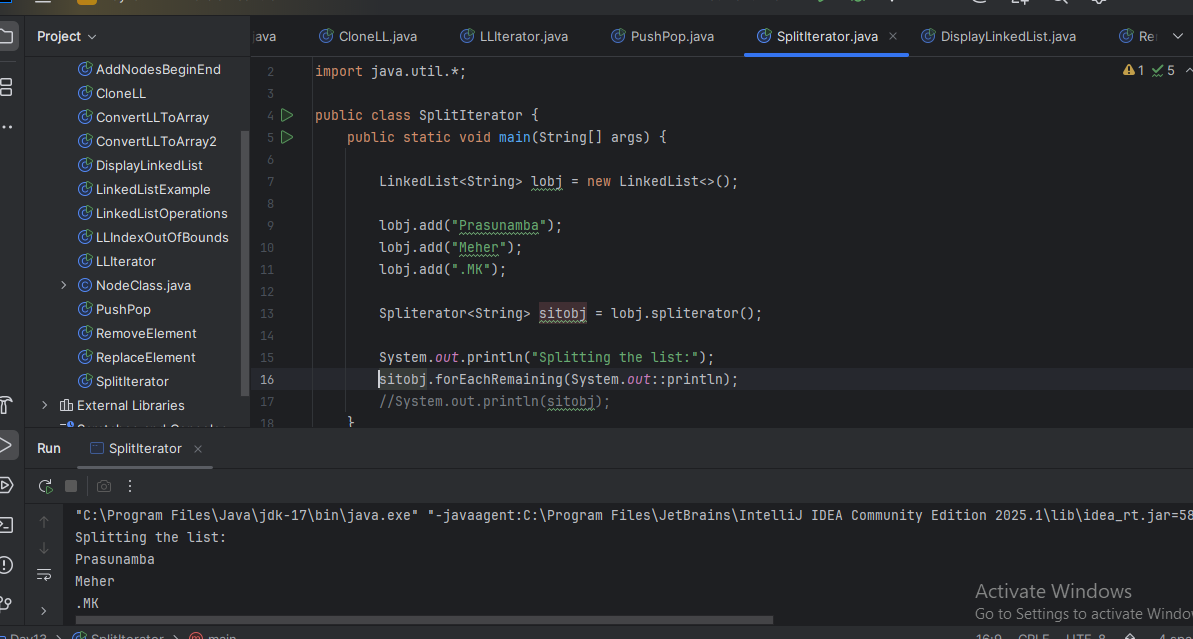
        Spliterator<String> sitobj = lobj.spliterator();

        System.out.println("Splitting the list:");

        sitobj.forEachRemaining(System.out::println);

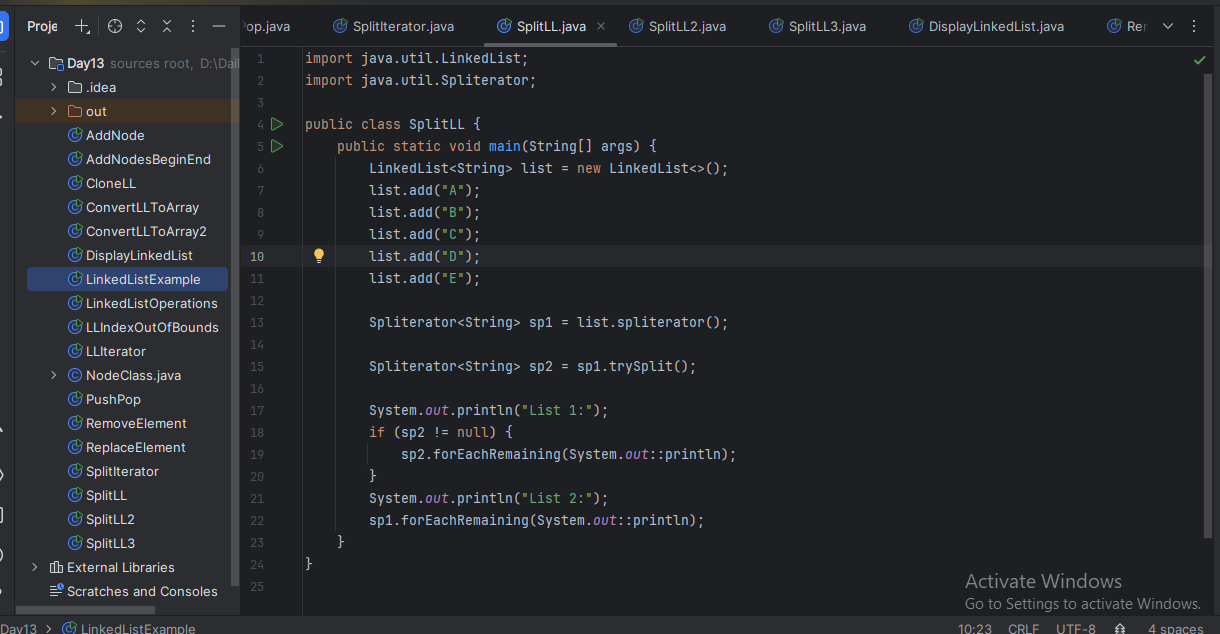
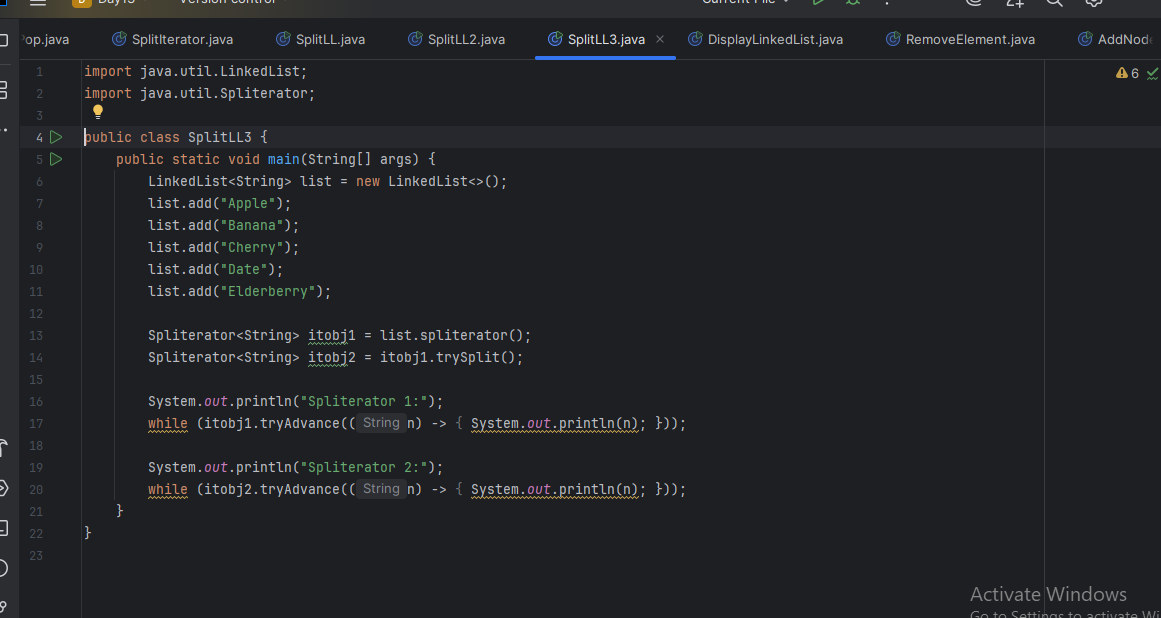
    }

}



**Task 14:**

Create a linkedlist and display items into 2 lists using split iterator



**Task 15:**

What do you understand by a pointer?

Answer:

A pointer in programming (especially in languages like C or C++) is a variable that stores the memory address of another variable. A pointer "points to" the location of another variable in memory.

**Task 16:**

Difference between \* and & in pointers?

Answer:

\* (asterisk): Used to declare or dereference a pointer

& (ampersand): Used to get the address of a variable

Example in C:

int x = 10;

int \*ptr = &x;

x is a normal integer variable, &x is the address of x, ptr is a pointer to an int, and it stores the address of x.

**Task 17:**Wap in c or c++ to implement the use of pointers.

**#include <stdio.h>**

**int main() {**

**int number = 10; // Declare an integer variable**

**int \*ptr = &number; // Pointer stores the address of number**

**// Print value using variable and pointer**

**printf("Value of number: %d\n", number);**

**printf("Value using pointer: %d\n", \*ptr);**

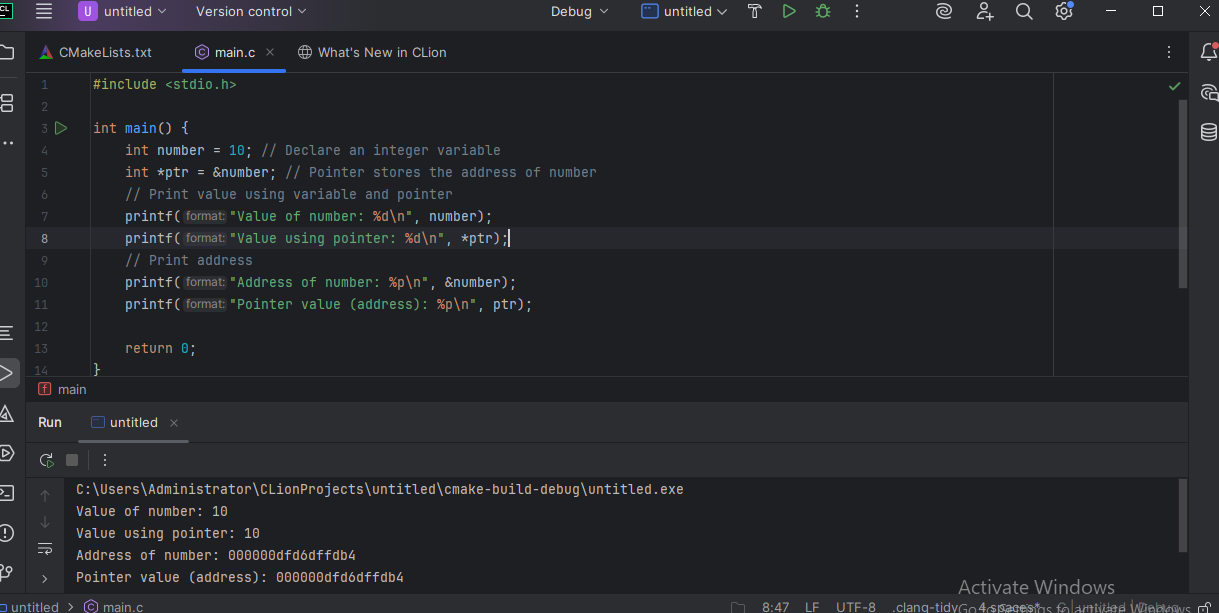
**// Print address**

**printf("Address of number: %p\n", &number);**

**printf("Pointer value (address): %p\n", ptr);**

**return 0;**

**}**

****

**Task 18:**

Wap to create a doubly linked list

