**Task 1:**

#include <iostream>

int main() {

// declaration

int age; // variable

int\* ptr; // pointer variable

// assigning

age = 10;

ptr = &age;

printf("value of age is %d \n", age); // 10

printf("ptr is pointing to %d \n ", \*ptr); // 10

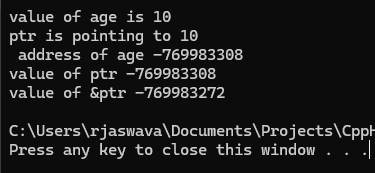
printf("address of age %d \n", &age);// 145245

printf("value of ptr %d \n", ptr);//same as above 145245

printf("value of &ptr %d \n", &ptr); // ptr address --- 45457

}

**Output:**



Task 02:

Linked list

Linked list in c++

Task 03:

Linked list

Linked list in java

public class task03 {

    // Node class definition

    static class Node {

        int data;

        Node next;

        // Constructor for Node

        Node(int value) {

            data = value;

            next = null;

        }

    }

    // LinkedList class definition

    static class LinkedList {

        private Node head;

        // Constructor to initialize empty list

        public LinkedList() {

            head = null;

        }

        // Function to insert a node at the end

        public void insertAtEnd(int value) {

            Node newNode = new Node(value);

            if (head == null) {

                head = newNode; // If list is empty, make newNode the head

            } else {

                Node temp = head;

                while (temp.next != null) {

                    temp = temp.next; // Traverse to the last node

                }

                temp.next = newNode; // Link the last node to newNode

            }

        }

        // Function to delete a Node by Value

        public void deleteByValue(int value) {

            if (head == null) {

                return;

            }

            if (head.data == value) {

                head = head.next; // Move head to the next node

                return;

            }

            Node temp = head;

            while (temp.next != null && temp.next.data != value) {

                temp = temp.next; // Traverse to find the node to delete

            }

            if (temp.next != null) {

                temp.next = temp.next.next; // Unlink the node

            }

        }

        // Function to display the list

        public void display() {

            Node temp = head;

            while (temp != null) {

                System.out.print(temp.data + "->");

                temp = temp.next;

            }

            System.out.println("NULL");

        }

    }

    public static void main(String[] args) {

        LinkedList list = new LinkedList();

        list.insertAtEnd(10);

        list.insertAtEnd(20);

        list.insertAtEnd(30);

        System.out.print("Linked List: ");

        list.display();

        list.deleteByValue(20);

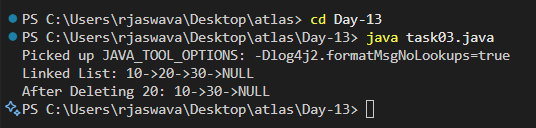
        System.out.print("After Deleting 20: ");

        list.display();

    }

}

Output:



Task 04:

Try to create a node and add a value to it.. Which can take any kind of data in the Node..

public class task04<T> {

    // Generic Node class

    private class Node<T> {

        T data;

        Node<T> next;

        Node(T data) {

            this.data = data;

            this.next = null;

        }

    }

    private Node<T> head;

    private int size;

    // Constructor

    public task04() {

        head = null;

        size = 0;

    }

    // Add element at the end of the list

    public void addLast(T data) {

        Node<T> newNode = new Node<>(data);

        size++;

        if (head == null) {

            head = newNode;

            return;

        }

        Node<T> current = head;

        while (current.next != null) {

            current = current.next;

        }

        current.next = newNode;

    }

    // Remove node by value

    public boolean remove(T data) {

        if (head == null) {

            return false;

        }

        if (head.data.equals(data)) {

            head = head.next;

            size--;

            return true;

        }

        Node<T> current = head;

        while (current.next != null && !current.next.data.equals(data)) {

            current = current.next;

        }

        if (current.next != null) {

            current.next = current.next.next;

            size--;

            return true;

        }

        return false;

    }

    // Get element at specific index

    public T get(int index) {

        if (index < 0 || index >= size) {

            throw new IndexOutOfBoundsException("Index: " + index + ", Size: " + size);

        }

        Node<T> current = head;

        for (int i = 0; i < index; i++) {

            current = current.next;

        }

        return current.data;

    }

    // Display all elements

    public void display() {

        if (head == null) {

            System.out.println("List is empty");

            return;

        }

        Node<T> current = head;

        while (current != null) {

            System.out.print(current.data + " -> ");

            current = current.next;

        }

        System.out.println("NULL");

    }

    // Get size of the list

    public int size() {

        return size;

    }

    public static void main(String[] args) {

        // Example with Integer type

        System.out.println("Integer LinkedList Example:");

        task04<Integer> intList = new task04<>();

        // Adding elements

        intList.addLast(1);

        intList.addLast(2);

        intList.addLast(3);

        System.out.println("After adding elements:");

        intList.display();

        System.out.println("Size: " + intList.size());

        // Removing element

        intList.remove(2);

        System.out.println("\nAfter removing 2:");

        intList.display();

        System.out.println("Size: " + intList.size());

        // Getting element by index

        try {

            System.out.println("\nElement at index 1: " + intList.get(1));

            // This will throw IndexOutOfBoundsException

            System.out.println("Element at index 5: " + intList.get(5));

        } catch (IndexOutOfBoundsException e) {

            System.out.println("Error: " + e.getMessage());

        }

        // Example with String type

        System.out.println("\nString LinkedList Example:");

        task04<String> stringList = new task04<>();

        // Adding elements

        stringList.addLast("Hello");

        stringList.addLast("World");

        stringList.addLast("Java");

        System.out.println("After adding elements:");

        stringList.display();

        System.out.println("Size: " + stringList.size());

        // Removing element

        stringList.remove("World");

        System.out.println("\nAfter removing 'World':");

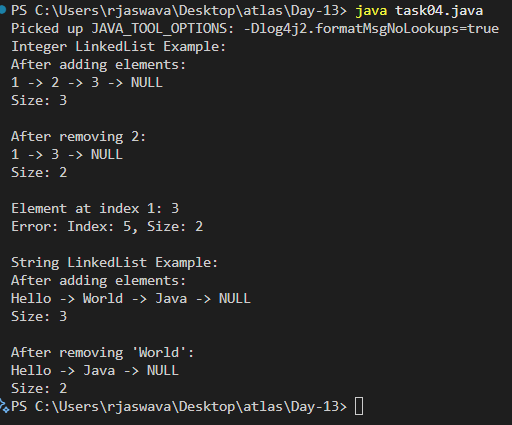
        stringList.display();

        System.out.println("Size: " + stringList.size());

    }

}

Output:



Task 05:

List down all the methods of Linked list

LinkedList Methods:

1. Adding Elements:

add(element) // Adds at end

add(index, element) // Adds at position

addFirst(element) // Adds at beginning

addLast(element) // Adds at end

2. Removing Elements:

remove(element) // Removes first occurrence

remove(index) // Removes at position

removeFirst() // Removes first element

removeLast() // Removes last element

clear() // Removes all elements

3. Getting Elements:

get(index) // Gets element at index

getFirst() // Gets first element

getLast() // Gets last element

indexOf(element) // Gets first position of element

4. Checking Elements:

contains(element) // Checks if present

isEmpty() // Checks if list empty

size() // Returns size of list

5. Setting Elements:

set(index, element) // Changes element at index

6. List Information:

peek() // Views first element

peekFirst() // Views first element

peekLast() // Views last element

7. Converting:

toArray() // Converts to array

toString() // Converts to string

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

list.add("First"); // Add

list.remove("First"); // Remove

String element = list.get(0); // Get

boolean exists = list.contains("First"); // Check

**Task 06**

Create linked list using Pre defined class and add elements to it.

Hint:

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

import java.util.LinkedList;

public class task06 {

    public static void main(String[] args) {

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

        System.out.println("Adding fruits to the LinkedList:");

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.addFirst("Orange");

        fruits.addLast("Mango");

        fruits.add(2, "Grapes");

        System.out.println("\nFruits in the LinkedList:");

        System.out.println(fruits);

        LinkedList<Integer> numbers = new LinkedList<>();

        numbers.add(10);

        numbers.add(20);

        numbers.add(30);

        numbers.addFirst(5);

        numbers.addLast(40);

        System.out.println("\nNumbers in the LinkedList:");

        System.out.println(numbers);

        System.out.println("\nDemonstrating LinkedList Operations:");

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

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

        System.out.println("Fruit at index 2: " + fruits.get(2));

        System.out.println("Number of fruits: " + fruits.size());

        System.out.println("Contains 'Apple'? " + fruits.contains("Apple"));

        System.out.println("\nRemoving elements:");

        fruits.removeFirst();

        System.out.println("After removing first: " + fruits);

        fruits.removeLast();

        System.out.println("After removing last: " + fruits);

        fruits.remove("Grapes");

        System.out.println("After removing 'Grapes': " + fruits);

        fruits.clear();

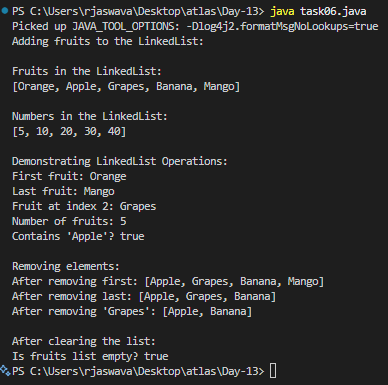
        System.out.println("\nAfter clearing the list:");

        System.out.println("Is fruits list empty? " + fruits.isEmpty());

    }

}

**Output**:



**Task 7:**

get first and get last elements and display all elements in the linked list

 import java.util.LinkedList;

public class task07 {

    public static void main(String[] args) {

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

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Orange");

        fruits.add("Mango");

        fruits.add("Grapes");

        System.out.println("LinkedList Operations:\n");

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

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

        System.out.println("\nMethod 1 - Direct printing:");

        System.out.println(fruits);

        System.out.println("\nMethod 2 - Using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nMethod 3 - Using traditional for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\nMethod 4 - Using forEach method:");

        fruits.forEach(fruit -> System.out.print(fruit + " -> "));

        System.out.println("null");

        System.out.println("\nAdditional Information:");

        System.out.println("Size of LinkedList: " + fruits.size());

        System.out.println("Is LinkedList empty? " + fruits.isEmpty());

        System.out.println("Does LinkedList contain 'Mango'? " + fruits.contains("Mango"));

        System.out.println("\nTesting with empty LinkedList:");

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

        try {

            System.out.println("First element of empty list: " + emptyList.getFirst());

        } catch (Exception e) {

            System.out.println("Error getting first element: " + e.getMessage());

        }

        try {

            System.out.println("Last element of empty list: " + emptyList.getLast());

        } catch (Exception e) {

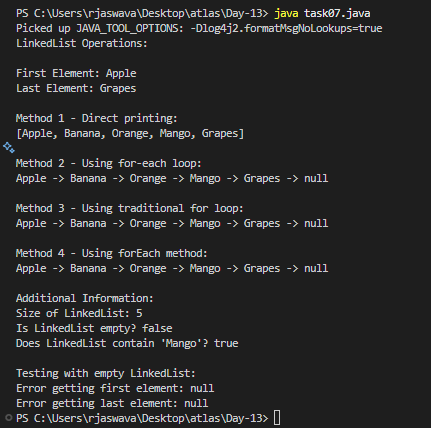
            System.out.println("Error getting last element: " + e.getMessage());

        }

    }

}

**Output**:



**Task 8:**

Remove first and remove last element and display all elements in the linked list Task 7

import java.util.LinkedList;

public class task08 {

    public static void main(String[] args) {

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

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Orange");

        fruits.add("Mango");

        fruits.add("Grapes");

        System.out.println("Original LinkedList:");

        displayList(fruits);

        System.out.println("\nRemoving First Element: " + fruits.removeFirst());

        System.out.println("After removing first element:");

        displayList(fruits);

        System.out.println("\nRemoving Last Element: " + fruits.removeLast());

        System.out.println("After removing last element:");

        displayList(fruits);

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

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

        System.out.println("\nDisplaying elements using different methods:");

        System.out.println("\nMethod 1 - Direct printing:");

        System.out.println(fruits);

        System.out.println("\nMethod 2 - Using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nMethod 3 - Using traditional for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\nAdditional Remove Operations:");

        System.out.println("Removing element at index 1: " + fruits.remove(1));

        System.out.println("After removing element at index 1:");

        displayList(fruits);

        System.out.println("\nRemoving 'Orange': " + fruits.remove("Orange"));

        System.out.println("After removing 'Orange':");

        displayList(fruits);

        System.out.println("\nTesting remove operations on empty list:");

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

        try {

            emptyList.removeFirst();

        } catch (Exception e) {

            System.out.println("Error removing first element: " + e.getMessage());

        }

        try {

            emptyList.removeLast();

        } catch (Exception e) {

            System.out.println("Error removing last element: " + e.getMessage());

        }

    }

    private static void displayList(LinkedList<String> list) {

        if (list.isEmpty()) {

            System.out.println("List is empty");

            return;

        }

        System.out.print("List elements: ");

        for (String item : list) {

            System.out.print(item + " -> ");

        }

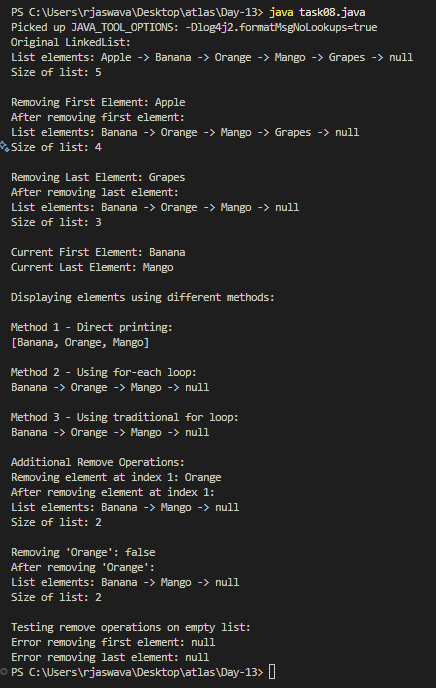
        System.out.println("null");

        System.out.println("Size of list: " + list.size());

    }

}

**Output:**



**Task 9:** in the list update the 1st element to a new value Hint: use set(1, "new value");

import java.util.LinkedList;

public class task09 {

    public static void main(String[] args) {

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

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Orange");

        fruits.add("Mango");

        fruits.add("Grapes");

        System.out.println("Original LinkedList:");

        displayList(fruits);

        System.out.println("\nUpdating elements:");

        System.out.println("Updating first element from '" + fruits.get(0) +

                         "' to 'Pineapple'");

        fruits.set(0, "Pineapple");

        System.out.println("Updating second element from '" + fruits.get(1) +

                         "' to 'Strawberry'");

        fruits.set(1, "Strawberry");

        System.out.println("\nLinkedList after updates:");

        displayList(fruits);

        System.out.println("\nRemoving First Element: " + fruits.removeFirst());

        System.out.println("After removing first element:");

        displayList(fruits);

        System.out.println("\nRemoving Last Element: " + fruits.removeLast());

        System.out.println("After removing last element:");

        displayList(fruits);

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

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

        System.out.println("\nDisplaying elements using different methods:");

        System.out.println("\nMethod 1 - Direct printing:");

        System.out.println(fruits);

        System.out.println("\nMethod 2 - Using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nMethod 3 - Using traditional for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\nTesting set() with invalid index:");

        try {

            fruits.set(10, "Invalid");

        } catch (Exception e) {

            System.out.println("Error updating element: " + e.getMessage());

        }

        if (!fruits.isEmpty()) {

            System.out.println("\nUpdating last element to 'Kiwi'");

            fruits.set(fruits.size() - 1, "Kiwi");

            System.out.println("After updating last element:");

            displayList(fruits);

        }

    }

    private static void displayList(LinkedList<String> list) {

        if (list.isEmpty()) {

            System.out.println("List is empty");

            return;

        }

        System.out.print("List elements: ");

        for (String item : list) {

            System.out.print(item + " -> ");

        }

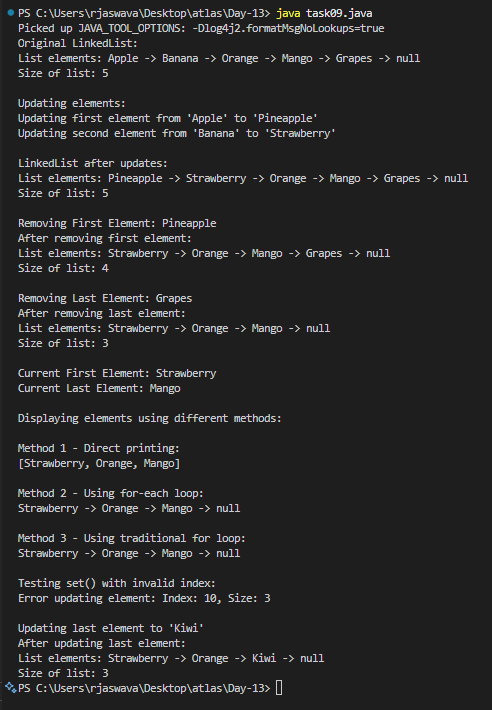
        System.out.println("null");

        System.out.println("Size of list: " + list.size());

    }

}

**Output:**



**Task 10**: display the list twice 1..... with get method in for loop and 2 ... for each loop

import java.util.LinkedList;

public class task10 {

    public static void main(String[] args) {

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

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Orange");

        fruits.add("Mango");

        fruits.add("Grapes");

        System.out.println("Original LinkedList:");

        System.out.println("\n1. Display using get() method in for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\n2. Display using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nUpdating elements:");

        fruits.set(0, "Pineapple");

        fruits.set(1, "Strawberry");

        System.out.println("\nAfter updates:");

        System.out.println("\n1. Display using get() method in for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\n2. Display using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nRemoving elements:");

        fruits.removeFirst();

        fruits.removeLast();

        System.out.println("\nAfter removing first and last elements:");

        System.out.println("\n1. Display using get() method in for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\n2. Display using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nError Handling Demonstration:");

        try {

            System.out.println("Trying to access invalid index:");

            System.out.println(fruits.get(10));

        } catch (Exception e) {

            System.out.println("Error: " + e.getMessage());

        }

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

        System.out.println("\nDisplaying empty list:");

        System.out.println("\n1. Display using get() method in for loop:");

        for(int i = 0; i < emptyList.size(); i++) {

            System.out.print(emptyList.get(i) + " -> ");

        }

        System.out.println("null");

        System.out.println("\n2. Display using for-each loop:");

        for(String item : emptyList) {

            System.out.print(item + " -> ");

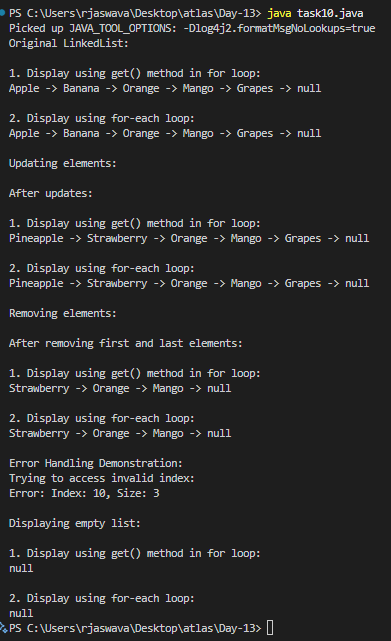
        }

        System.out.println("null");

    }

}

**Output:**



**Task 11**: display the elements of the linked list with out loops

import java.util.LinkedList;

import java.util.Arrays;

public class task11 {

    public static void main(String[] args) {

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

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Orange");

        fruits.add("Mango");

        fruits.add("Grapes");

        System.out.println("Displaying LinkedList in different ways:\n");

        System.out.println("1. Direct printing (toString method):");

        System.out.println(fruits);

        System.out.println("\n2. Using toString() explicitly:");

        System.out.println(fruits.toString());

        System.out.println("\n3. Converting to array and printing:");

        System.out.println(Arrays.toString(fruits.toArray()));

        System.out.println("\n4. Using String.join:");

        System.out.println(String.join(" -> ", fruits) + " -> null");

        System.out.println("\n5. Using get() method in for loop:");

        for(int i = 0; i < fruits.size(); i++) {

            System.out.print(fruits.get(i));

            if(i < fruits.size() - 1) {

                System.out.print(" -> ");

            }

        }

        System.out.println(" -> null");

        System.out.println("\n6. Using for-each loop:");

        for(String fruit : fruits) {

            System.out.print(fruit + " -> ");

        }

        System.out.println("null");

        System.out.println("\nAfter modifications:");

        fruits.set(0, "Pineapple");

        fruits.set(1, "Strawberry");

        System.out.println("\nAfter updating first two elements:");

        System.out.println("Direct print: " + fruits);

        System.out.println("String.join: " + String.join(" -> ", fruits) + " -> null");

        fruits.removeFirst();

        fruits.removeLast();

        System.out.println("\nAfter removing first and last elements:");

        System.out.println("Direct print: " + fruits);

        System.out.println("String.join: " + String.join(" -> ", fruits) + " -> null");

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

        System.out.println("\nDisplaying empty list:");

        System.out.println("Empty list direct print: " + emptyList);

        System.out.println("Empty list String.join: " + String.join(" -> ", emptyList) + " -> null");

        System.out.println("\nAdditional List Information:");

        System.out.println("Size of list: " + fruits.size());

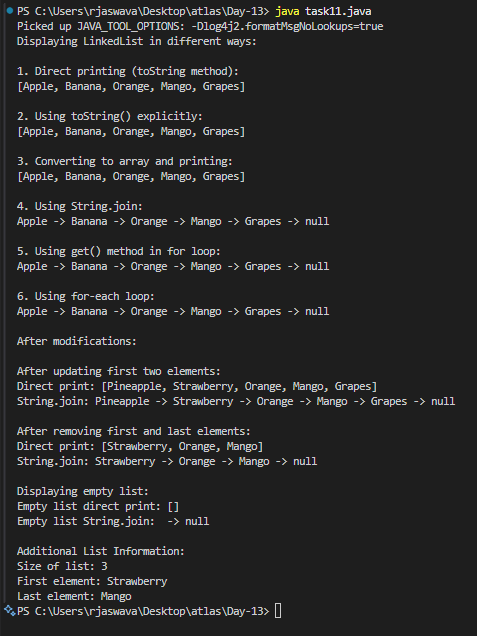
        System.out.println("First element: " + (fruits.isEmpty() ? "none" : fruits.getFirst()));

        System.out.println("Last element: " + (fruits.isEmpty() ? "none" : fruits.getLast()));

    }

}

**Output:**



**Task 12**: convert the linked list to an array and display Hint : Object[] a = llobj.toArray();

import java.util.LinkedList;

import java.util.Arrays;

public class task12 {

    public static void main(String[] args) {

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

        fruits.add("Apple");

        fruits.add("Banana");

        fruits.add("Orange");

        fruits.add("Mango");

        fruits.add("Grapes");

        System.out.println("Original LinkedList:");

        System.out.println(fruits);

        System.out.println("\n1. Converting to Object array:");

        Object[] array1 = fruits.toArray();

        System.out.println("Using Object array:");

        System.out.print("Array elements: ");

        for(Object fruit : array1) {

            System.out.print(fruit + " ");

        }

        System.out.println("\nArray length: " + array1.length);

        System.out.println("\n2. Converting to String array:");

        String[] array2 = fruits.toArray(new String[0]);

        System.out.println("Using String array:");

        System.out.print("Array elements: ");

        System.out.println(Arrays.toString(array2));

        System.out.println("Array length: " + array2.length);

        System.out.println("\n3. Converting to String array with exact size:");

        String[] array3 = fruits.toArray(new String[fruits.size()]);

        System.out.println("Using String array (exact size):");

        System.out.print("Array elements: ");

        System.out.println(Arrays.toString(array3));

        System.out.println("Array length: " + array3.length);

        System.out.println("\nAfter modifications:");

        fruits.set(0, "Pineapple");

        fruits.set(1, "Strawberry");

        Object[] modifiedArray = fruits.toArray();

        System.out.println("Modified list as array:");

        System.out.println(Arrays.toString(modifiedArray));

        fruits.removeFirst();

        fruits.removeLast();

        Object[] shorterArray = fruits.toArray();

        System.out.println("\nArray after removing elements:");

        System.out.println(Arrays.toString(shorterArray));

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

        Object[] emptyArray = emptyList.toArray();

        System.out.println("\nEmpty list to array:");

        System.out.println("Array length: " + emptyArray.length);

        System.out.println("Array content: " + Arrays.toString(emptyArray));

        LinkedList<Integer> numbers = new LinkedList<>();

        numbers.add(1);

        numbers.add(2);

        numbers.add(3);

        Object[] numberArray = numbers.toArray();

        System.out.println("\nInteger LinkedList to array:");

        System.out.println(Arrays.toString(numberArray));

        Integer[] intArray = numbers.toArray(new Integer[0]);

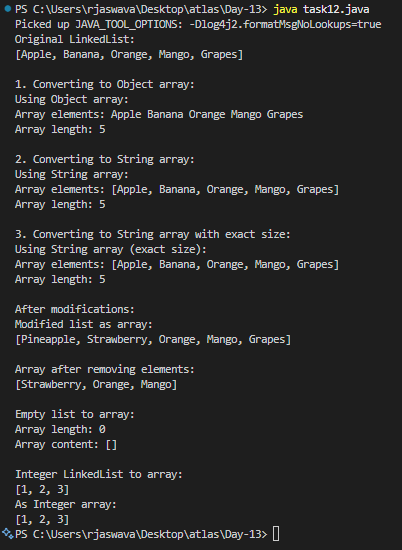
        System.out.println("As Integer array:");

        System.out.println(Arrays.toString(intArray));

    }

}

**Output:**



Task 13:

Use pop and push methods on linked list.. LIFO – just follow..

import java.util.LinkedList;

public class task13 {

    public static void main(String[] args) {

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

        System.out.println("Demonstrating LIFO (Stack) operations with LinkedList\n");

        System.out.println("1. Pushing elements:");

        stack.push("First");

        displayStack(stack);

        stack.push("Second");

        displayStack(stack);

        stack.push("Third");

        displayStack(stack);

        stack.push("Fourth");

        displayStack(stack);

        stack.push("Fifth");

        displayStack(stack);

        System.out.println("\n2. Popping elements:");

        try {

            System.out.println("Popped: " + stack.pop());

            displayStack(stack);

            System.out.println("Popped: " + stack.pop());

            displayStack(stack);

            System.out.println("Popped: " + stack.pop());

            displayStack(stack);

        } catch (Exception e) {

            System.out.println("Error: " + e.getMessage());

        }

        System.out.println("\n3. Peek operation:");

        System.out.println("Top element (peek): " + stack.peek());

        displayStack(stack);

        System.out.println("\n4. Pushing more elements:");

        stack.push("New Element 1");

        displayStack(stack);

        stack.push("New Element 2");

        displayStack(stack);

        System.out.println("\n5. Popping all elements:");

        while (!stack.isEmpty()) {

            System.out.println("Popped: " + stack.pop());

            displayStack(stack);

        }

        System.out.println("\n6. Trying to pop from empty stack:");

        try {

            stack.pop();

        } catch (Exception e) {

            System.out.println("Error: Cannot pop from empty stack");

        }

        System.out.println("\n7. Demonstrating with Integer type:");

        LinkedList<Integer> numberStack = new LinkedList<>();

        numberStack.push(10);

        numberStack.push(20);

        numberStack.push(30);

        System.out.println("After pushing numbers:");

        System.out.println(numberStack);

        System.out.println("\nPopping numbers:");

        while (!numberStack.isEmpty()) {

            System.out.println("Popped: " + numberStack.pop());

        }

        System.out.println("\n8. Additional Stack Operations:");

        stack.push("Element A");

        stack.push("Element B");

        stack.push("Element C");

        System.out.println("Current stack:");

        displayStack(stack);

        System.out.println("First element: " + stack.getFirst());

        System.out.println("Last element: " + stack.getLast());

        System.out.println("\n9. Clearing the stack");

        stack.clear();

        displayStack(stack);

    }

    private static void displayStack(LinkedList<?> stack) {

        if (stack.isEmpty()) {

            System.out.println("Stack is empty");

            return;

        }

        System.out.print("Stack (top to bottom): ");

        for (Object item : stack) {

            System.out.print(item + " -> ");

        }

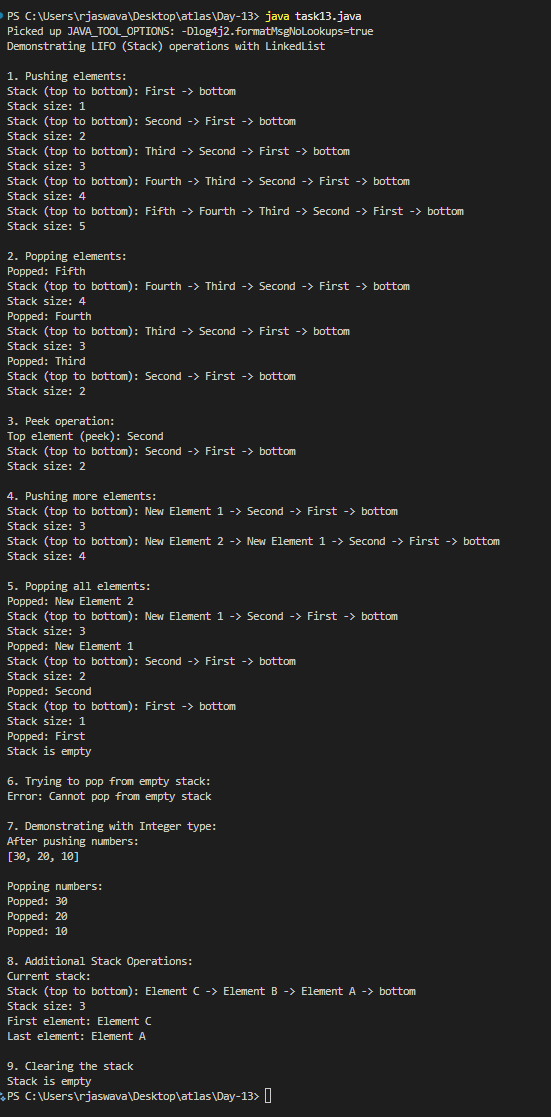
        System.out.println("bottom");

        System.out.println("Stack size: " + stack.size());

    }

}

Output:



Task 14:

Splititerator

import java.util.\*;

public class task14 {

    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();

        //forEachRemaining is a method of Spliterator

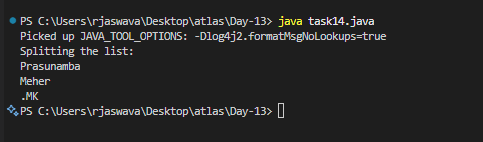
        System.out.println("Splitting the list:");

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

    }

}

Output:



Task 15:

import java.util.LinkedList;

import java.util.Spliterator;

public class task15 {

    public static void main(String[] args) {

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

        llobj.add("Prasunamba");

        llobj.add("Meher");

        llobj.add(".MK");

        llobj.add("MP");

        Spliterator<String> itobj1 = llobj.spliterator();

        Spliterator<String> itobj2 = itobj1.trySplit();

        System.out.println("spliterator 1");

        while( itobj1.tryAdvance( (n) -> { System.out.println(n); } ) );

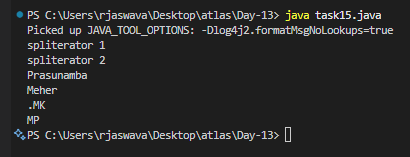
        System.out.println("spliterator 2");

        while( itobj2.tryAdvance( (n) -> { System.out.println(n); } ) );

    }

}

Output:



Task 16:

Create a doubly linked list..

import java.util.\*;

public class task16 {

    class Node {

        int data;

        Node prev;

        Node next;

        Node(int data) {

            this.data = data;

        }

    }

    Node head, tail = null;

    public void addNode(int data) {

        Node newNode = new Node(data);

        if(head == null) {

            head = tail = newNode;

            head.prev = null;

            tail.next = null;

        }

        else {

            tail.next = newNode;

            newNode.prev = tail;

            tail = newNode;

            tail.next = null;

        }

    }

    public void display() {

        Node current = head;

        if(head == null) {

            System.out.println("List is empty");

            return;

        }

        System.out.println("Nodes of doubly linked list: ");

        while(current != null) {

            System.out.print(current.data + " ");

            current = current.next;

        }

    }

    public static void main(String[] args) {

        task16 dList = new task16();

        dList.addNode(10);

        dList.addNode(20);

        dList.addNode(30);

        dList.addNode(40);

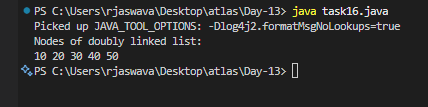
        dList.addNode(50);

        dList.display();

    }

}

Output:



Task 17:

Create a Hash MAp of capacity 10.

Hint:

HashMap<String, Integer> hm2 = new HashMap<String, Integer>(10);

import java.util.HashMap;

import java.util.Map;

public class task17 {

    public static void main(String[] args) {

        HashMap<String, Integer> hm = new HashMap<String, Integer>(10);

        hm.put("One", 1);

        hm.put("Two", 2);

        hm.put("Three", 3);

        hm.put("Four", 4);

        hm.put("Five", 5);

        System.out.println("HashMap with capacity 10:");

        System.out.println("Size: " + hm.size());

        System.out.println("\nElements in HashMap:");

        for(Map.Entry<String, Integer> entry : hm.entrySet()) {

            System.out.println(entry.getKey() + ": " + entry.getValue());

        }

        System.out.println("\nGet value for key 'Three': " + hm.get("Three"));

        hm.remove("Two");

        System.out.println("\nAfter removing 'Two':");

        System.out.println(hm);

        System.out.println("\nContains key 'Four': " + hm.containsKey("Four"));

        System.out.println("Contains value 2: " + hm.containsValue(2));

        hm.clear();

        System.out.println("\nAfter clearing HashMap:");

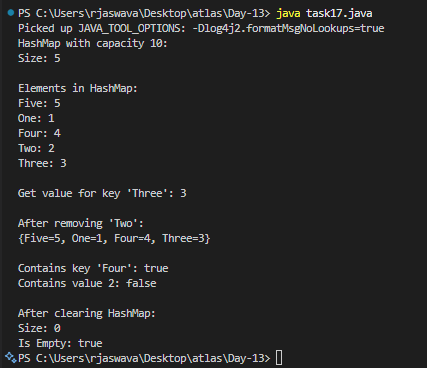
        System.out.println("Size: " + hm.size());

        System.out.println("Is Empty: " + hm.isEmpty());

    }

}

Output:



Task 18:

Copy data from one map to another map.

hint:

HashMap<String, Integer> hm3 = new HashMap<String, Integer>( hm2);

import java.util.HashMap;

import java.util.Map;

public class task18 {

    public static void main(String[] args) {

        HashMap<String, Integer> firstMap = new HashMap<String, Integer>();

        firstMap.put("One", 1);

        firstMap.put("Two", 2);

        firstMap.put("Three", 3);

        firstMap.put("Four", 4);

        firstMap.put("Five", 5);

        HashMap<String, Integer> secondMap = new HashMap<String, Integer>(firstMap);

        HashMap<String, Integer> thirdMap = new HashMap<String, Integer>();

        thirdMap.putAll(firstMap);

        System.out.println("Original Map:");

        System.out.println(firstMap);

        System.out.println("\nSecond Map (copied using constructor):");

        System.out.println(secondMap);

        System.out.println("\nThird Map (copied using putAll):");

        System.out.println(thirdMap);

        firstMap.put("Six", 6);

        System.out.println("\nAfter modifying first map:");

        System.out.println("First Map: " + firstMap);

        System.out.println("Second Map: " + secondMap);

        System.out.println("Third Map: " + thirdMap);

        secondMap.clear();

        System.out.println("\nAfter clearing second map:");

        System.out.println("First Map: " + firstMap);

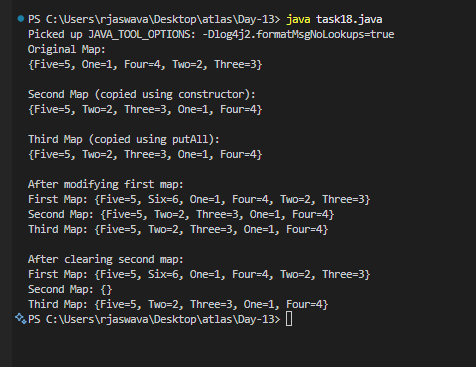
        System.out.println("Second Map: " + secondMap);

        System.out.println("Third Map: " + thirdMap);

    }

}

Output:



**Task 19:**

Task020\_DS\_HashMapCreateMethods:

Different methods to create a hashmap in java :

1) Constructing a hashmap with default capacity

ex:

 HashMap<String, Integer> hm1 = new HashMap<String, Integer>();

2) Constructing a hashmap with a capacity 10

ex:

HashMap<String, Integer> hm2 = new HashMap<String, Integer>(10);

3)copy one map to another map

ex:

HashMap<String, Integer> hm3 = new HashMap<String, Integer>( hm2);

4)

Specifying load factor along with the capacity

ex:

 HashMap<String, Integer> hm4= new HashMap<String, Integer>(10, 0.75f);

Initial capacity  ===10

Load factor  === 0.75f

import java.util.HashMap;

public class task19 {

    public static void main(String[] args) {

        HashMap<String, Integer> hm1 = new HashMap<String, Integer>();

        hm1.put("A", 1);

        hm1.put("B", 2);

        System.out.println("1. Default HashMap:");

        System.out.println(hm1);

        HashMap<String, Integer> hm2 = new HashMap<String, Integer>(10);

        hm2.put("X", 10);

        hm2.put("Y", 20);

        hm2.put("Z", 30);

        System.out.println("\n2. HashMap with capacity 10:");

        System.out.println(hm2);

        HashMap<String, Integer> hm3 = new HashMap<String, Integer>(hm2);

        System.out.println("\n3. HashMap copied from hm2:");

        System.out.println(hm3);

        HashMap<String, Integer> hm4 = new HashMap<String, Integer>(10, 0.75f);

        hm4.put("P", 100);

        hm4.put("Q", 200);

        hm4.put("R", 300);

        System.out.println("\n4. HashMap with capacity 10 and load factor 0.75:");

        System.out.println(hm4);

        hm2.put("New", 40);

        System.out.println("\nAfter modifying hm2:");

        System.out.println("hm2: " + hm2);

        System.out.println("hm3 (copy of original hm2): " + hm3);

        System.out.println("\nSizes of HashMaps:");

        System.out.println("hm1 size: " + hm1.size());

        System.out.println("hm2 size: " + hm2.size());

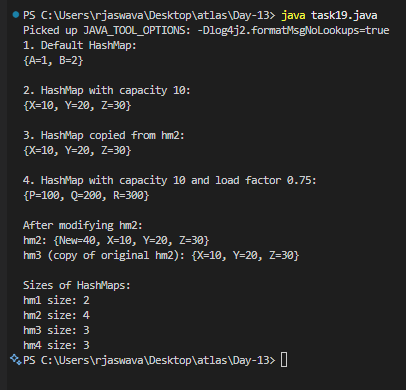
        System.out.println("hm3 size: " + hm3.size());

        System.out.println("hm4 size: " + hm4.size());

    }

}

**Output:**

****

Task 20:

Use custom method of Creating  a circular linked list and traverse the elements (display)

public class task20 {

    class Node {

        int data;

        Node next;

        Node(int data) {

            this.data = data;

            next = null;

        }

    }

    Node head = null;

    Node tail = null;

    public void addNode(int data) {

        Node newNode = new Node(data);

        if(head == null) {

            head = newNode;

            tail = newNode;

            tail.next = head;

        }

        else {

            tail.next = newNode;

            tail = newNode;

            tail.next = head;

        }

    }

    public void display() {

        Node current = head;

        if(head == null) {

            System.out.println("List is empty");

            return;

        }

        System.out.println("Nodes of circular linked list: ");

        do {

            System.out.print(current.data + " -> ");

            current = current.next;

        } while(current != head);

        System.out.println(head.data);

    }

    public static void main(String[] args) {

        task20 cList = new task20();

        cList.addNode(10);

        cList.addNode(20);

        cList.addNode(30);

        cList.addNode(40);

        cList.addNode(50);

        cList.display();

        System.out.println("\nVerifying circular nature:");

        Node temp = cList.head;

        for(int i = 0; i < 7; i++) {

            System.out.print(temp.data + " -> ");

            temp = temp.next;

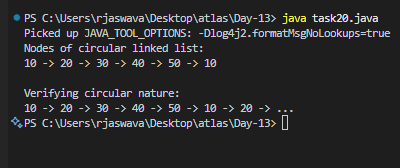
        }

        System.out.println("...");

    }

}

**Output:**

****