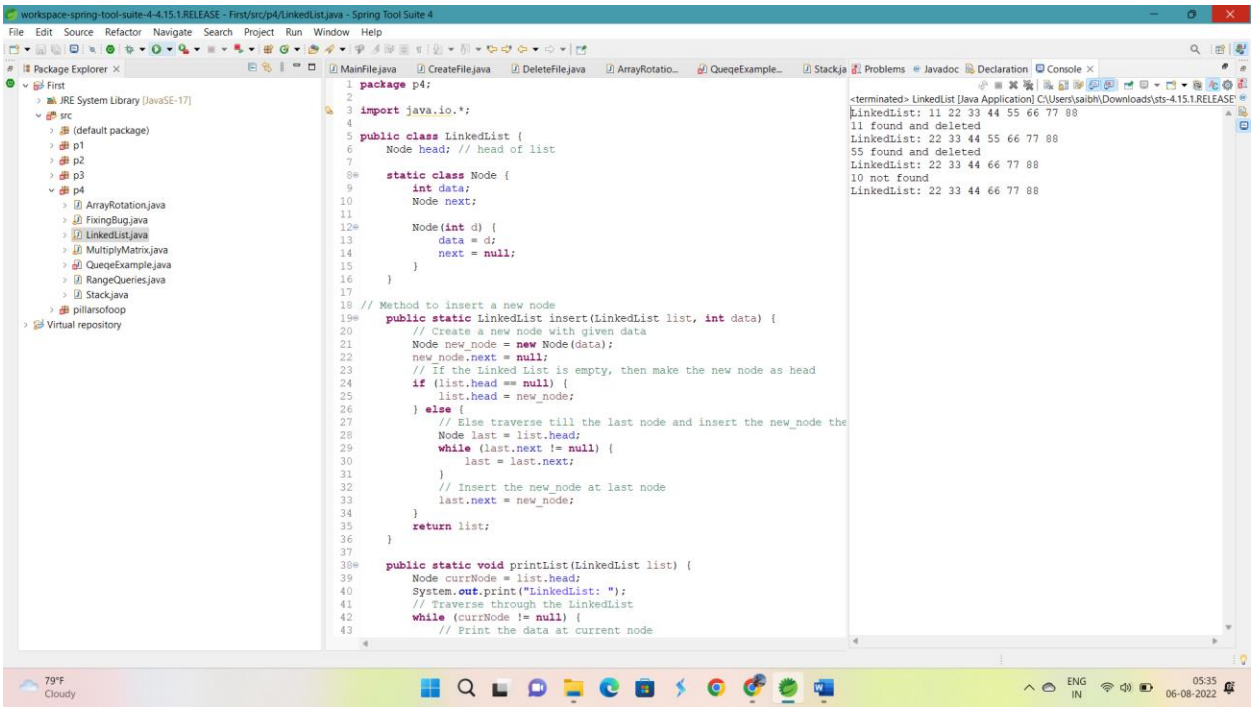
Writing the program in Java to understand the working of the singly linked list

```
package p4;

import java.io.*;

public class LinkedList {
	Node head; // head of list

	static class Node {
		int data;
		Node next;

		Node(int d) {
			data = d;
			next = null;
		}
	}

// Method to insert a new node
	public static LinkedList insert(LinkedList list, int data) {
		// Create a new node with given data
		Node new_node = new Node(data);
		new_node.next = null;
		// If the Linked List is empty, then make the new node as head
		if (list.head == null) {
			list.head = new_node;
		} else {
			// Else traverse till the last node and insert the new_node
there
			Node last = list.head;
			while (last.next != null) {
				last = last.next;
			}
			// Insert the new_node at last node
			last.next = new_node;
		}
		return list;
	}

	public static void printList(LinkedList list) {
		Node currNode = list.head;
		System.out.print("LinkedList: ");
		// Traverse through the LinkedList
		while (currNode != null) {
			// Print the data at current node
			System.out.print(currNode.data + " ");
			// Go to next node
			currNode = currNode.next;
		}
		System.out.println();
	}

	// Method to delete a node in the LinkedList by KEY
	public static LinkedList deleteByKey(LinkedList list, int key) {
```

```
		// Store head node
		Node currNode = list.head, prev = null;
		if (currNode != null && currNode.data == key) {
			list.head = currNode.next; // Changed head
			System.out.println(key + " found and deleted");
			return list;
		}
		;
		while (currNode != null && currNode.data != key) {
			prev = currNode;
			currNode = currNode.next;
		}
		if (currNode != null) {
			prev.next = currNode.next;
			System.out.println(key + " found and deleted");
		}
		if (currNode == null) {
			System.out.println(key + " not found");
		}
		return list;
	}

	// method to create a Singly linked list with n nodes
	public static void main(String[] args) {
		/* Start with the empty list. */
		LinkedList list = new LinkedList();
		// Insert the values
		list = insert(list, 11);
		list = insert(list, 22);
		list = insert(list, 33);
		list = insert(list, 44);
		list = insert(list, 55);
		list = insert(list, 66);
		list = insert(list, 77);
		list = insert(list, 88);
		// Print the LinkedList
		printList(list);
		// Delete node with value 1
		deleteByKey(list, 11);
		// Print the LinkedList
		printList(list);
		// Delete node with value 4
		deleteByKey(list, 55);
		// Print the LinkedList
		printList(list);
		// Delete node with value 10
		deleteByKey(list, 10);
		// Print the LinkedList
		printList(list);
	}
}
```




```java
package p4;

import java.io.*;

public class LinkedList {
    Node head; // head of list

    static class Node {
        int data;
        Node next;

        Node(int d) {
            data = d;
            next = null;
        }
    }

// Method to insert a new node
    public static LinkedList insert(LinkedList list, int data) {
        // Create a new node with given data
        Node new_node = new Node(data);
        new_node.next = null;
        // If the Linked List is empty, then make the new node as head
        if (list.head == null) {
            list.head = new_node;
        } else {
            // Else traverse till the last node and insert the new_node the
            Node last = list.head;
            while (last.next != null) {
                last = last.next;
            }
            // Insert the new_node at last node
            last.next = new_node;
        }
        return list;
    }

    public static void printList(LinkedList list) {
        Node currNode = list.head;
        System.out.print("LinkedList: ");
        // Traverse through the LinkedList
        while (currNode != null) {
            // Print the data at current node
```