

Linked list Assignment

Answer to question no.1

Java code

```
class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;

        Node(int data) {
            this.data = data;
            next = null;
        }
    }

    public boolean search(int X) {
        Node current = head;
        while (current != null) {
            if (current.data == X)
                return true;
            current = current.next;
        }
        return false;
    }

    public static void main(String[] args)
    {
        LinkedList list1 = new LinkedList();
        LinkedList list2 = new LinkedList();

        list1.head = new Node(14);
        list1.head.next = new Node(21);
        list1.head.next.next = new Node(11);
        list1.head.next.next.next = new Node(30);
        list1.head.next.next.next.next = new Node(10);
    }
}
```

```
list2.head = new Node(6);
list2.head.next = new Node(21);
list2.head.next.next = new Node(17);
list2.head.next.next.next = new Node(30);
list2.head.next.next.next.next = new Node(10);
list2.head.next.next.next.next.next = new Node(8);

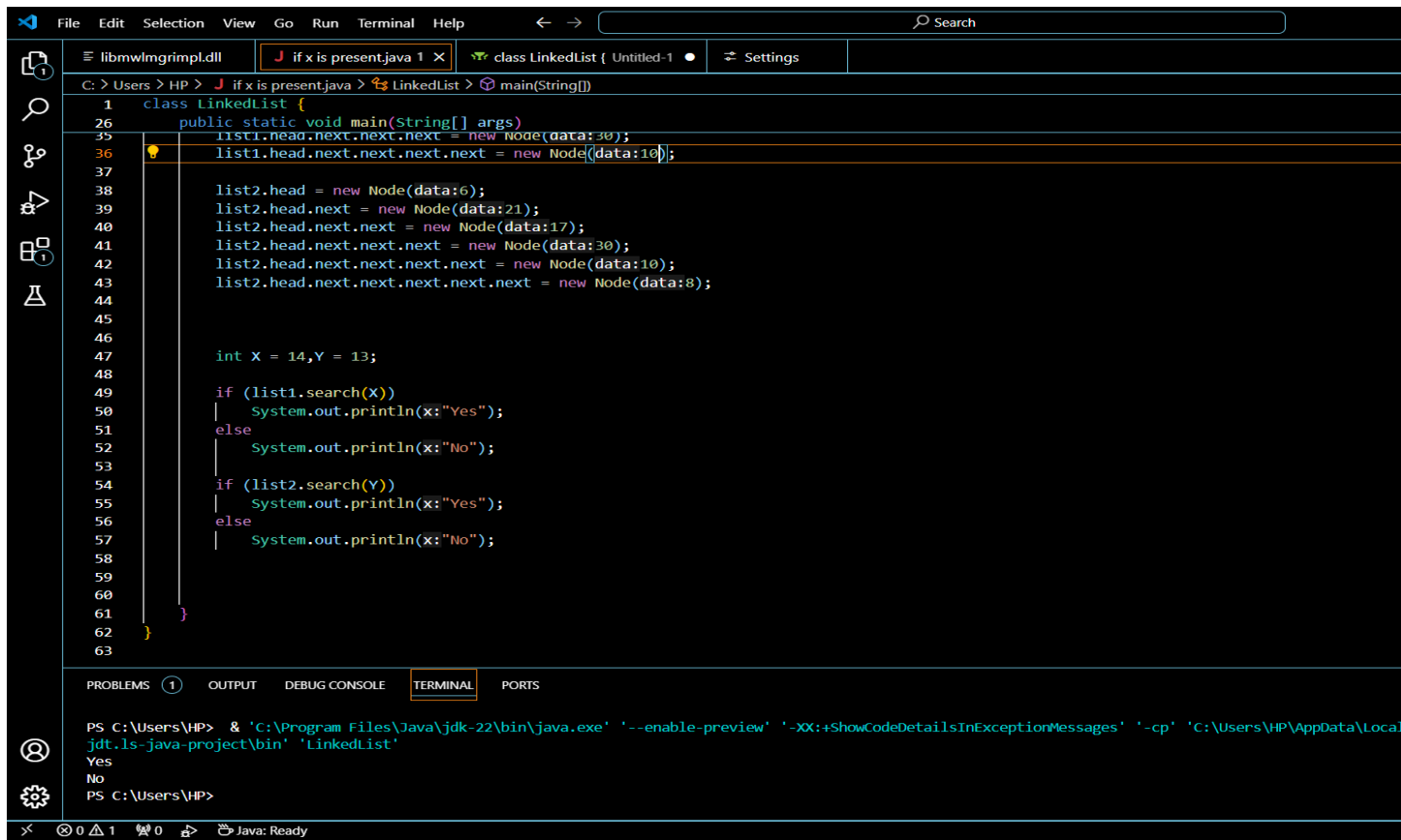
int X = 14,Y = 13;

if (list1.search(X))
    System.out.println("Yes");
else
    System.out.println("No");

if (list2.search(Y))
    System.out.println("Yes");
else
    System.out.println("No");

    }
}
```

Output :



```
File Edit Selection View Go Run Terminal Help
libmwingrimpl.dll J if x is present.java 1 X class LinkedList { Untitled-1 Settings
C: > Users > HP > J if x is present.java > LinkedList > main(String[])
1 class LinkedList {
26 public static void main(String[] args)
35 list1.head.next.next.next = new Node(data:30);
36 list1.head.next.next.next.next = new Node(data:10);
37
38 list2.head = new Node(data:6);
39 list2.head.next = new Node(data:21);
40 list2.head.next.next = new Node(data:17);
41 list2.head.next.next.next = new Node(data:30);
42 list2.head.next.next.next.next = new Node(data:10);
43 list2.head.next.next.next.next.next = new Node(data:8);
44
45
46
47 int X = 14,Y = 13;
48
49 if (list1.search(X))
50 | System.out.println(x:"Yes");
51 | else
52 | System.out.println(x:"No");
53 |
54 if (list2.search(Y))
55 | System.out.println(x:"Yes");
56 | else
57 | System.out.println(x:"No");
58
59
60
61 }
62
63

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\HP> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\HP\AppData\Local\jdt.ls-java-project\bin' 'LinkedList'
Yes
No
PS C:\Users\HP>
```

Answer to question no: 2

```
class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;

        Node(int data) {
            this.data = data;
            next = null;
        }
    }

    public void insertAfter(Node prev_node, int new_data) {
        if (prev_node == null) {
```

```

        System.out.println("Previous node cannot be null");
        return;
    }

    Node new_node = new Node(new_data);
    new_node.next = prev_node.next;
    prev_node.next = new_node;
}

public void printList() {
    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.head = new Node(1);
    list.head.next = new Node(2);
    list.head.next.next = new Node(4);
    list.head.next.next.next = new Node(5);
    list.head.next.next.next.next = new Node(6);

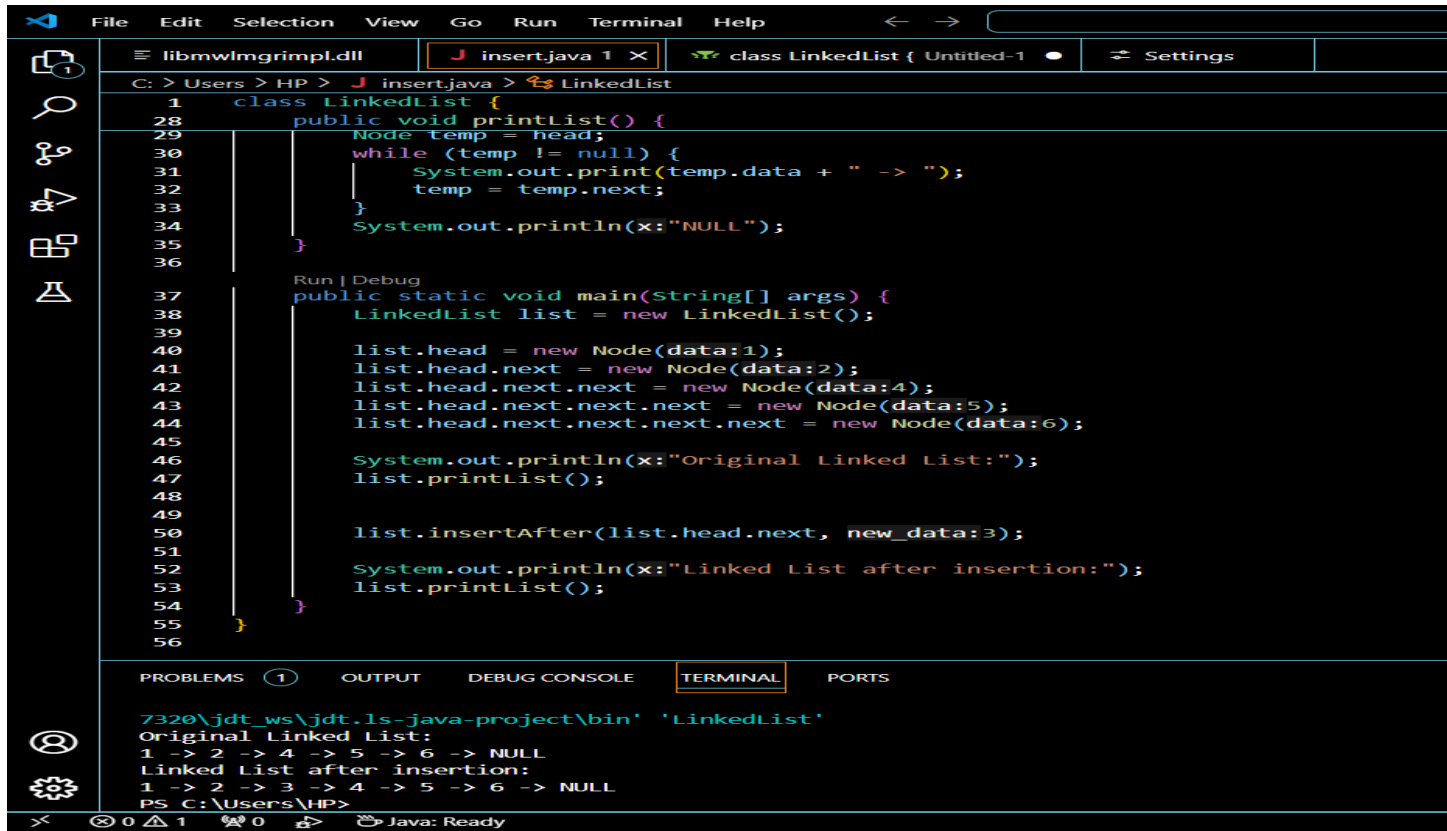
    System.out.println("Original Linked List:");
    list.printList();

    list.insertAfter(list.head.next, 3);

    System.out.println("Linked List after insertion:");
    list.printList();
}
}

```

Output:



The screenshot shows an IDE with a Java file named 'insert.java'. The code defines a 'LinkedList' class with a 'printList()' method and a 'main()' method. The 'main()' method creates a linked list with nodes containing values 1, 2, 4, 5, and 6. It then inserts a new node with value 3 after the node with value 2. The output in the terminal shows the original list and the list after insertion.

```
1 class LinkedList {
28     public void printList() {
29         Node temp = head;
30         while (temp != null) {
31             System.out.print(temp.data + " -> ");
32             temp = temp.next;
33         }
34         System.out.println(x:"NULL");
35     }
36
37     public static void main(String[] args) {
38         LinkedList list = new LinkedList();
39
40         list.head = new Node(data:1);
41         list.head.next = new Node(data:2);
42         list.head.next.next = new Node(data:4);
43         list.head.next.next.next = new Node(data:5);
44         list.head.next.next.next.next = new Node(data:6);
45
46         System.out.println(x:"Original Linked List");
47         list.printList();
48
49         list.insertAfter(list.head.next, new_data:3);
50
51         System.out.println(x:"Linked List after insertion:");
52         list.printList();
53     }
54 }
55
56
```

Terminal Output:

```
7320\jdt_ws\jdt.ls-java-project\bin' 'LinkedList'
Original Linked List:
1 -> 2 -> 4 -> 5 -> 6 -> NULL
Linked List after insertion:
1 -> 2 -> 3 -> 4 -> 5 -> 6 -> NULL
PS C:\Users\HP>
```

Answer to question no : 3

```
class LinkedList {
    Node head;
    static class Node {
        int data;
        Node next;

        Node(int data) {
            this.data = data;
            next = null;
        }
    }

    public void removeDuplicates() {
        Node current = head;
```

```

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

        current.next = current.next.next;
    } else {

        current = current.next;
    }
}

}

public void printList() {
    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.head = new Node(1);
    list.head.next = new Node(1);
    list.head.next.next = new Node(2);
    list.head.next.next.next = new Node(3);
    list.head.next.next.next.next = new Node(3);

    System.out.println("Original Linked List:");
    list.printList();

    list.removeDuplicates();

    System.out.println("Linked List after removing duplicates:");
    list.printList();
}

```

```
}
```

Output :

The screenshot shows an IDE with a Java file named `removedu.java` containing a `LinkedList` class. The class has a `main` method that creates two linked lists, `list1` and `list2`. `list1` contains nodes with data 1, 2, and 3. `list2` contains nodes with data 1, 2, 3, and 3. The program prints the original lists and then calls `removeDuplicates()` on both. The terminal output shows the original lists and the result after removing duplicates: `list1` becomes 1 -> 2 -> NULL and `list2` becomes 1 -> 2 -> 3 -> NULL.

```
1 class LinkedList {
41 public static void main(String[] args) {
50
51     list2.head = new Node(data:1);
52     list2.head.next = new Node(data:1);
53     list2.head.next.next = new Node(data:2);
54     list2.head.next.next.next = new Node(data:3);
55     list2.head.next.next.next.next = new Node(data:3);
56
57
58
59     System.out.println(x:"Original Linked List:");
60     list1.printList();
61
62
63     list1.removeDuplicates();
64
65     System.out.println(x:"Linked List after removing duplicates:");
66     list1.printList();
67
68     System.out.println(x:"Original Linked List:");
69     list2.printList();
70
71
72     list2.removeDuplicates();
73
74     System.out.println(x:"Linked List after removing duplicates:");
75     list2.printList();
76 }
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

Original Linked List:
1 -> 1 -> 2 -> NULL
Linked List after removing duplicates:
1 -> 2 -> NULL
Original Linked List:
1 -> 1 -> 2 -> 3 -> 3 -> NULL
Linked List after removing duplicates:
1 -> 2 -> 3 -> NULL
PS C:\Users\HP>

Answer to question no : 4

```
class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;

        Node(int data) {
```

```

        this.data = data;
        next = null;
    }
}

public boolean isPalindrome() {
    if (head == null || head.next == null) {
        return true;
    }

    Node slow = head;
    Node fast = head;
    while (fast != null && fast.next != null) {
        slow = slow.next;
        fast = fast.next.next;
    }

    Node secondHalf = reverseList(slow);

    Node firstHalf = head;
    Node secondHalfCopy = secondHalf;
    while (secondHalf != null) {
        if (firstHalf.data != secondHalfCopy.data) {
            return false;
        }
        firstHalf = firstHalf.next;
        secondHalfCopy = secondHalfCopy.next;
    }

    reverseList(secondHalfCopy);

    return true;
}

private Node reverseList(Node head) {
    Node prev = null;
    Node current = head;

```



```

        while (current != null) {
            Node next = current.next;
            current.next = prev;
            prev = current;
            current = next;
        }
        return prev;
    }

    public void printList() {
        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.head = new Node(1);
        list.head.next = new Node(2);
        list.head.next.next = new Node(2);
        list.head.next.next.next = new Node(1);

        System.out.println("Is palindrome? " + list.isPalindrome());

        LinkedList list2 = new LinkedList();
        list2.head = new Node(1);
        list2.head.next = new Node(2);

        System.out.println("Is palindrome? " + list2.isPalindrome());
    }
}

```

Output :

The screenshot shows an IDE with a Java file named `LLpalindrome.java`. The code defines a `LinkedList` class with a `Node` inner class. The `isPalindrome()` method uses a slow-fast pointer approach to check for palindromes. The terminal output shows the program running successfully, returning `true` for a palindrome and `false` for a non-palindrome.

```
1 class LinkedList {
2     Node head;
3
4     static class Node {
5         int data;
6         Node next;
7
8         Node(int data) {
9             this.data = data;
10            next = null;
11        }
12    }
13
14
15    public boolean isPalindrome() {
16        if (head == null || head.next == null) {
17            return true;
18        }
19
20
21        Node slow = head;
22        Node fast = head;
23        while (fast != null && fast.next != null) {
24            slow = slow.next;
25            fast = fast.next.next;
26        }
27
28        Node secondHalf = reverseList(slow);
29
30
31    }
```

Terminal Output:

```
PS C:\Users\HP>
PS C:\Users\HP> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\HP\AppData\Local\jd\jdt.ls-java-project\bin' 'LinkedList'
Is palindrome? true
Is palindrome? false
PS C:\Users\HP>
```

Answer to question no : 5

```
class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;

        Node(int data) {
            this.data = data;
            this.next = null;
        }
    }

    public void reverseList() {
        Node prev = null;
        Node current = head;
        while (current != null) {
```

```

        Node nextNode = current.next;
        current.next = prev;
        prev = current;
        current = nextNode;
    }
    head = prev;
}

public static LinkedList addTwoNumbers(LinkedList l1, LinkedList l2) {

    l1.reverseList();
    l2.reverseList();

    Node p1 = l1.head;
    Node p2 = l2.head;
    LinkedList result = new LinkedList();
    Node dummyHead = new Node(0);
    Node current = dummyHead;
    int carry = 0;

    while (p1 != null || p2 != null) {
        int x = (p1 != null) ? p1.data : 0;
        int y = (p2 != null) ? p2.data : 0;
        int sum = carry + x + y;
        carry = sum / 10;
        current.next = new Node(sum % 10);
        current = current.next;

        if (p1 != null) p1 = p1.next;
        if (p2 != null) p2 = p2.next;
    }

    if (carry > 0) {
        current.next = new Node(carry);
    }

    result.head = dummyHead.next;
    result.reverseList();
    return result;
}

```

```
public void printList() {  
    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 l1 = new LinkedList();  
    LinkedList l2 = new LinkedList();  
    LinkedList l3 = new LinkedList();  
    LinkedList l4 = new LinkedList();
```

```
    l1.head = new Node(5);  
    l1.head.next = new Node(6);  
    l1.head.next.next = new Node(3);
```

```
    l2.head = new Node(8);  
    l2.head.next = new Node(4);  
    l2.head.next.next = new Node(2);
```

```
    l3.head = new Node(7);  
    l3.head.next = new Node(5);  
    l3.head.next.next = new Node(9);  
    l3.head.next.next.next = new Node(4);  
    l3.head.next.next.next.next = new Node(6);
```

```
    l4.head = new Node(8);  
    l4.head.next = new Node(4);
```

```
    System.out.println("List 1:");  
    l1.printList();  
    System.out.println("List 2:");  
    l2.printList();  
    System.out.println("List 3:");  
    l3.printList();
```

```

        System.out.println("List 4:");
        l4.printList();

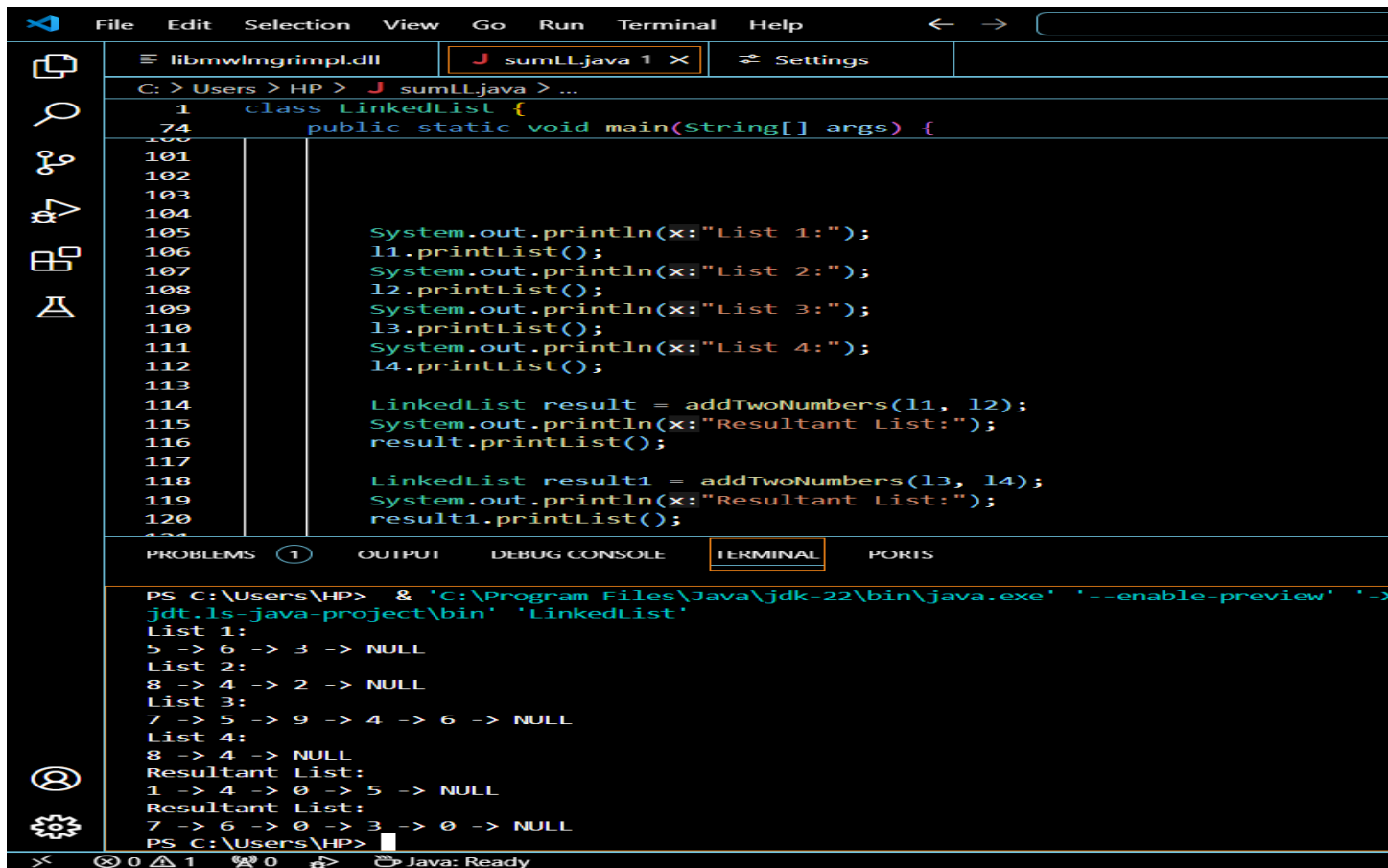
        LinkedList result = addTwoNumbers(l1, l2);
        System.out.println("Resultant List:");
        result.printList();

        LinkedList result1 = addTwoNumbers(l3, l4);
        System.out.println("Resultant List:");
        result1.printList();

    }
}

```

Output :



The screenshot shows an IDE with the following components:

- Editor:** Displays the source code for `LinkedList.java`. The code includes a `main` method that creates four linked lists (l1, l2, l3, l4) and prints their contents. It also shows two calls to `addTwoNumbers` and prints the resultant lists.
- Terminal:** Shows the output of the program. The output is as follows:


```

PS C:\Users\HP> & 'C:\Program Files\Java\jdk-22\bin\java.exe' '--enable-preview' '-jdt.ls-java-project\bin' 'LinkedList'
List 1:
5 -> 6 -> 3 -> NULL
List 2:
8 -> 4 -> 2 -> NULL
List 3:
7 -> 5 -> 9 -> 4 -> 6 -> NULL
List 4:
8 -> 4 -> NULL
Resultant List:
1 -> 4 -> 0 -> 5 -> NULL
Resultant List:
7 -> 6 -> 0 -> 3 -> 0 -> NULL
PS C:\Users\HP>

```