Credit Name: Chapter13
Assignment Name: StackList

How has your program changed from planning to coding to now? Please explain?

At first, the program was planned to show how a stack can be implemented using a linked list. The initial plan included creating a StackList class and a test class to manage the program. The methods like push(), pop(), peek(), isEmpty(), and size() were made to provide the basic stack functionalities.

1. Missing Node Class

Problem: The Node class was initially missing, which caused compilation errors since there was no structure to hold the stack elements.

Fix: I resolved this by defining the Node class with fields for data and a reference to the next node.

Code Example:

```
private class Node {
    private Object data;
    private Node next;
    public Node(Object data) {
        this.data = data;
        this.next = null;
    }
    public Object getData() {
        return data;
    }
    public Node getNext() {
        return next;
    }
    public void setNext(Node next) {
        this.next = next;
    }
}
```

2. Linking Nodes in push()

Problem: The push() method overwrote the top node without linking it to the existing stack, resulting in data loss.

Fix: I fixed this by linking the new node to the current top before updating the top.

```
// Before
public void push(Object item) {
    top = new Node(item);
}
// After
public void push(Object item) {
    Node newNode = new Node(item);
    newNode.setNext(top);
    top = newNode;
}
```

3. Updating top in pop()

Problem: In the pop() method, the top reference was not updated after removing the top node, causing the stack to break.

Fix: I updated the top to point to the next node after the current top was removed.

```
// Before
public Object pop() {
    return top.getData();
}
// After
public Object pop() {
    if (isEmpty()) {
        System.out.println("Stack is empty. Cannot pop.");
        return null;
    }
    Object item = top.getData();
    top = top.getNext();
    return item;
}
```

4. Error Handling in peek()

Problem: The peek() method did not handle empty stacks, leading to runtime errors.

Fix: I added error handling to return a message when the stack was empty.

```
// Before
public Object peek() {
    return top.getData();
}
// After
public Object peek() {
    if (isEmpty()) {
        System.out.println("Stack is empty. No top item.");
        return null;
    }
    return top.getData();
}
```

5. Traversing the Stack in size()

Problem: The size() method failed to traverse the stack correctly, as it did not move to the next node in the loop.

Fix: I updated the logic to increment the counter and move to the next node.

```java
// Before
public int size() {
    int count = 0;
    Node current = top;
    while (current != null) {
        count++;
    }
    return count;
}
// After
public int size() {
    int count = 0;
    Node current = top;
    while (current != null) {
        count++;
        current = current.getNext();
    }
    return count;
}
```