Credit Name: Chapter 13

Assignment Name: StackList

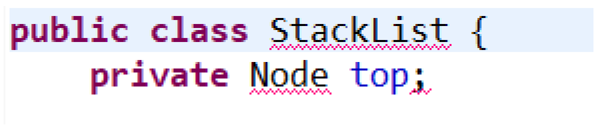
Describe the errors you've encountered while working on this assignment. What caused the error and how do you overcome the error?

1. Missing Node Class Implementation

Error: The Node class was missing from the StackList implementation, causing a compilation error.

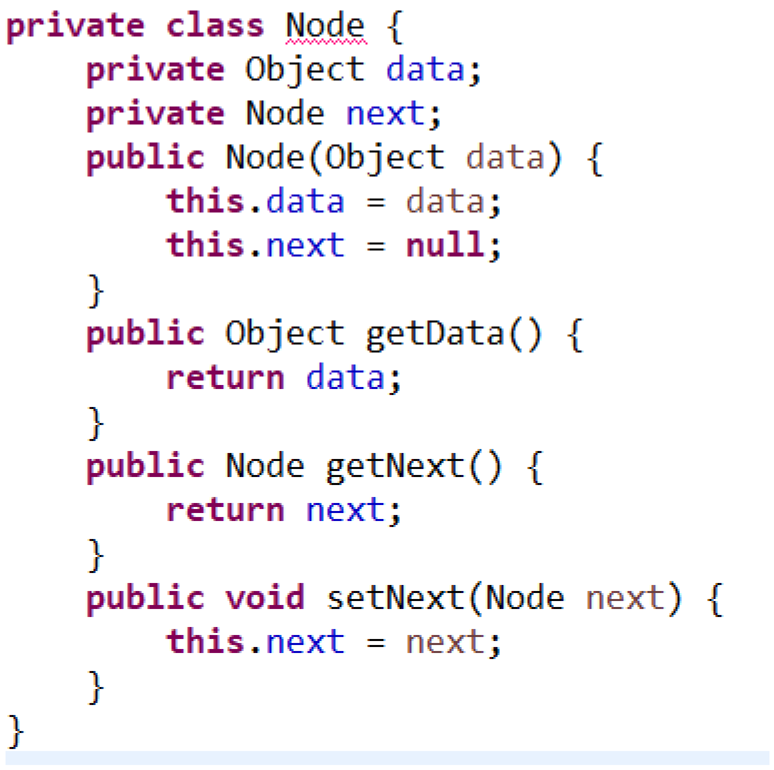
Cause: The Node class is required for creating nodes in the linked list and was not defined.

Error Code:



```java
public class StackList {
    private Node top;
```

Fix: Added the Node class as a nested class within StackList the node logic and ensures it's accessible to the StackList class only.



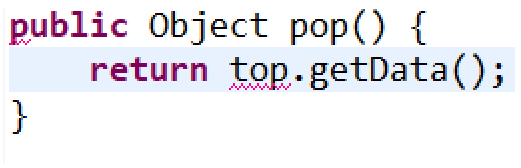
```java
private class Node {
    private Object data;
    private Node next;
    public Node(Object data) {
        this.data = data;
        this.next = null;
    }
    public Object getData() {
        return data;
    }
    public Node getNext() {
        return next;
    }
    public void setNext(Node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

2. Forgetting to Update top in pop()

Error: The pop() method did not update the top reference, leading to an incorrect stack state.

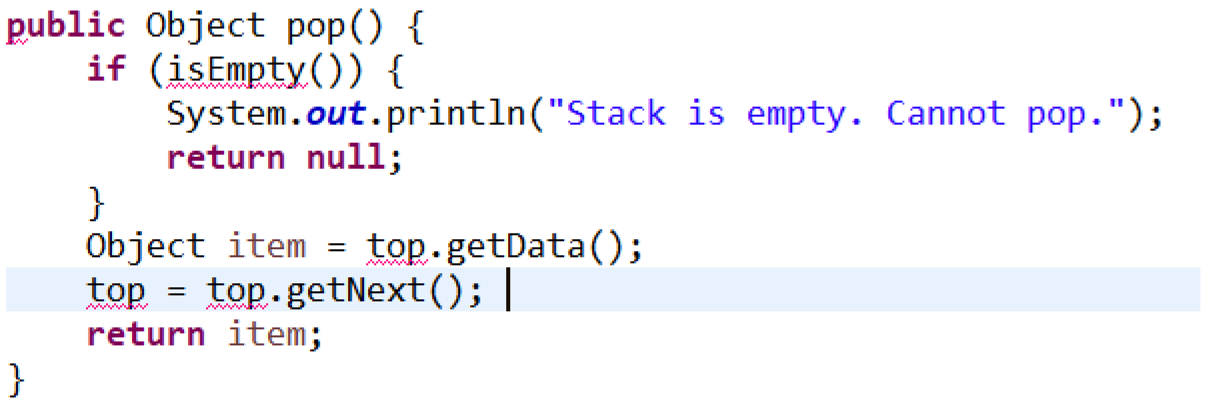
Cause: After popping the top element, the stack pointer (top) was not moved to the next node.

Error Code:



```
public Object pop() {
    return top.getData();
}
```

Fix: Updated the top reference in the pop() method to point to the next node after removing the current top. This ensures the stack remains consistent.



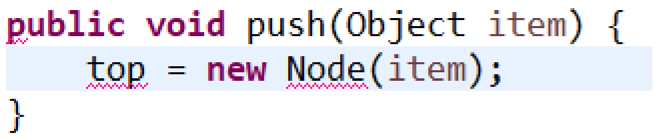
```
public Object pop() {
    if (isEmpty()) {
        System.out.println("Stack is empty. Cannot pop.");
        return null;
    }
    Object item = top.getData();
    top = top.getNext();
    return item;
}
```

3. push() Overwrites Top Node

Error: The push() method overwrote the top node without linking the new node to the existing stack.

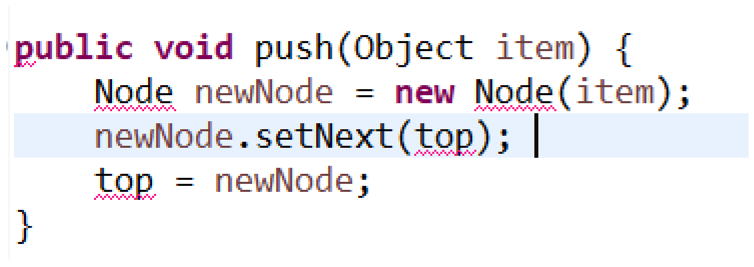
Cause: The previous top node was not linked to the new node, leading to data loss.

Error Code:



```
public void push(Object item) {
    top = new Node(item);
}
```

Fix: Linked the new node to the existing top before updating the top reference. This ensures the new node becomes the new top while keeping the stack intact.



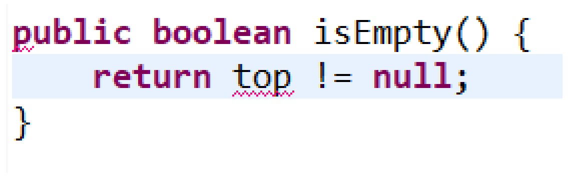
```java
public void push(Object item) {
    Node newNode = new Node(item);
    newNode.setNext(top);
    top = newNode;
}
```

4. Incorrect isEmpty() Logic

Error: The isEmpty() method returned incorrect results, causing incorrect program behavior.

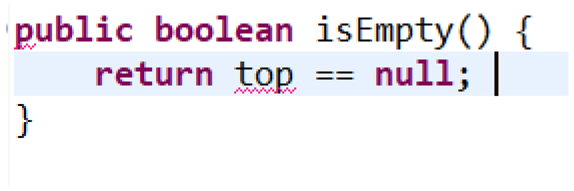
Cause: The condition in isEmpty() checked top != null instead of top == null.

Error Code:



```java
public boolean isEmpty() {
    return top != null;
}
```

Fix: Corrected the condition in the isEmpty() method to return true when the stack is empty. This checks if top is null, which indicates the stack is empty.



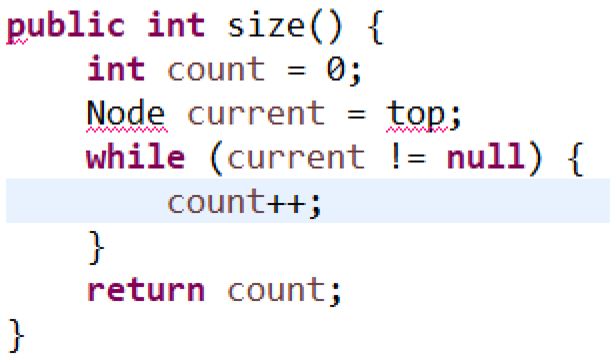
```java
public boolean isEmpty() {
    return top == null;
}
```

5. Traversal Error in size()

Error: The size() method resulted in an infinite loop or an incorrect node count.

Cause: The current node reference was not updated in the while loop, causing an infinite loop.

Error Code:



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

Fix: Added current = current.getNext() within the loop to correctly traverse the stack. This ensures that the method counts all nodes and terminates when the end of the stack is reached.

```java
public int size() {

    int count = 0;

    Node current = top; // Start at the top node

    while (current != null) {

        count++; // Increment count for each node

        current = current.getNext(); // Move to the next node

    }

    return count;

}
```