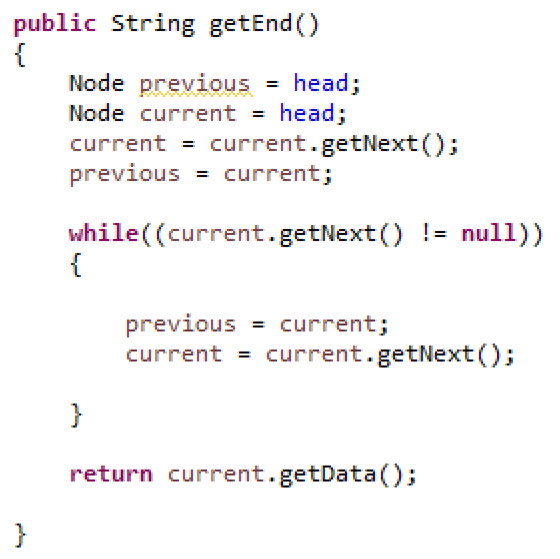
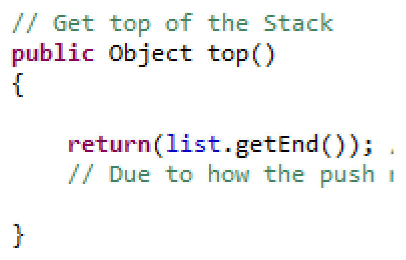
Credit: CSE3110 - Iterative Algorithm 1
Assignment: StackList

How has your code changed from planning to coding? Please explain.

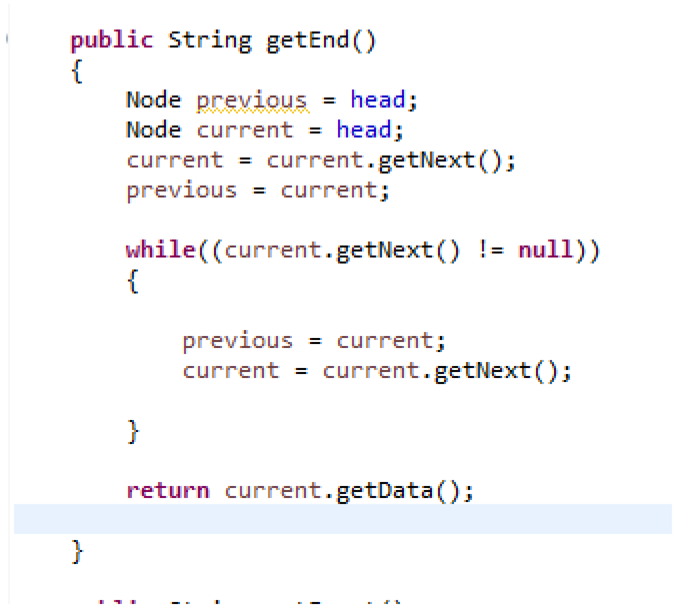
At first I planned to use the addAtFront method in the linked list class for my push method in my stack class. This would've worked but since the top method of a stack class returns the last pushed item, this means it would have to return the front node of the list. I found it easier to find the last node of a list because the next node would be null. I used the addAtEnd method to my linked list class to use as my push for my stack class. I also added a getEnd method to the linked list class to use as my top method in the stack class.



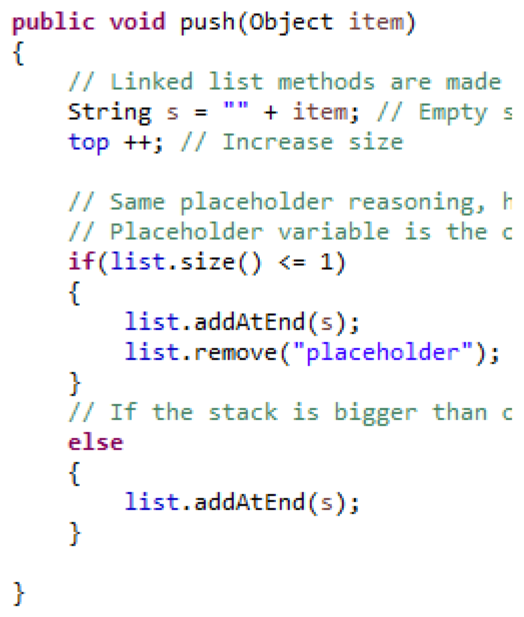
```java
public String getEnd()
{
    Node previous = head;
    Node current = head;
    current = current.getNext();
    previous = current;

    while((current.getNext() != null))
    {

        previous = current;
        current = current.getNext();

    }

    return current.getData();

}
```



```java
// Get top of the Stack
public Object top()
{

    return(list.getEnd()); .
    // Due to how the push ı

}
```

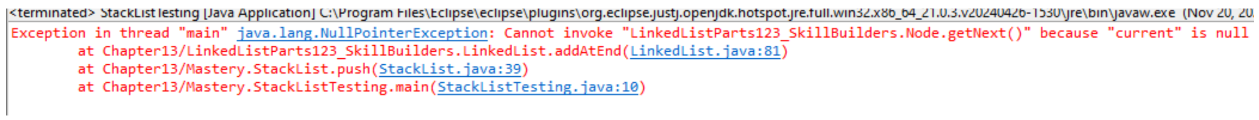


```java
public String getEnd()
{
    Node previous = head;
    Node current = head;
    current = current.getNext();
    previous = current;

    while((current.getNext() != null))
    {

        previous = current;
        current = current.getNext();

    }

    return current.getData();

}
```



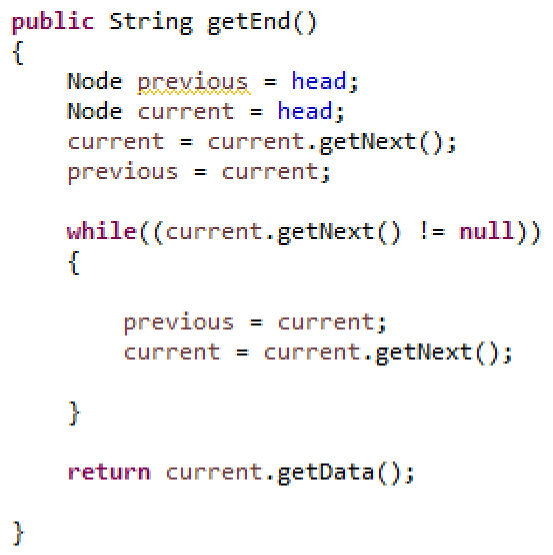
```java
public void push(Object item)
{
    // Linked list methods are made
    String s = "" + item; // Empty s
    top ++; // Increase size

    // Same placeholder reasoning, h
    // Placeholder variable is the c
    if(list.size() <= 1)
    {
        list.addAtEnd(s);
        list.remove("placeholder");
    }
    // If the stack is bigger than c
    else
    {
        list.addAtEnd(s);
    }

}
```

This system would work until the last two nodes of the linkedlist or a size 2 stack. If I tried to do any methods to it, I would run into an error.



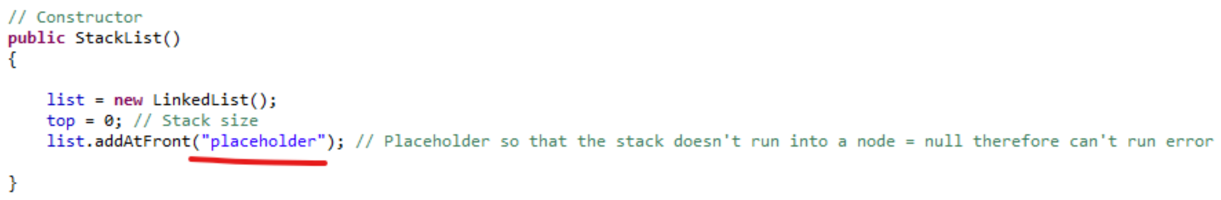
```
<terminated> StackListTesting [Java Application] C:\Program Files\Eclipse\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.3.v20240426-1530\jre\bin\javaw.exe (Nov 20, 20
Exception in thread "main" java.lang.NullPointerException: Cannot invoke "LinkedListParts123_SkillBuilders.Node.getNext()" because "current" is null
	at Chapter13/LinkedListParts123_SkillBuilders.LinkedList.addAtEnd(LinkedList.java:81)
	at Chapter13/Mastery.StackList.push(StackList.java:39)
	at Chapter13/Mastery.StackListTesting.main(StackListTesting.java:10)
```

At first I tried to use a current and previous variable combo in the getEnd method in the linkedlist class. This didn't work.

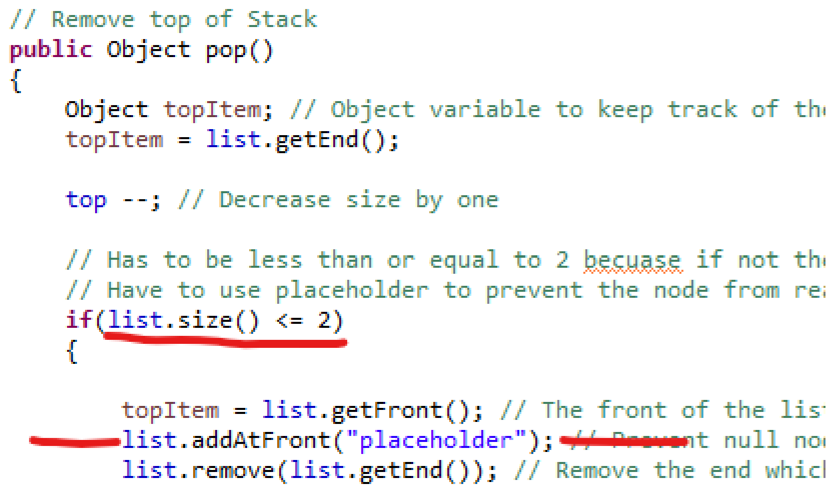


```java
public String getEnd()
{
    Node previous = head;
    Node current = head;
    current = current.getNext();
    previous = current;

    while((current.getNext() != null))
    {

        previous = current;
        current = current.getNext();

    }

    return current.getData();

}
```

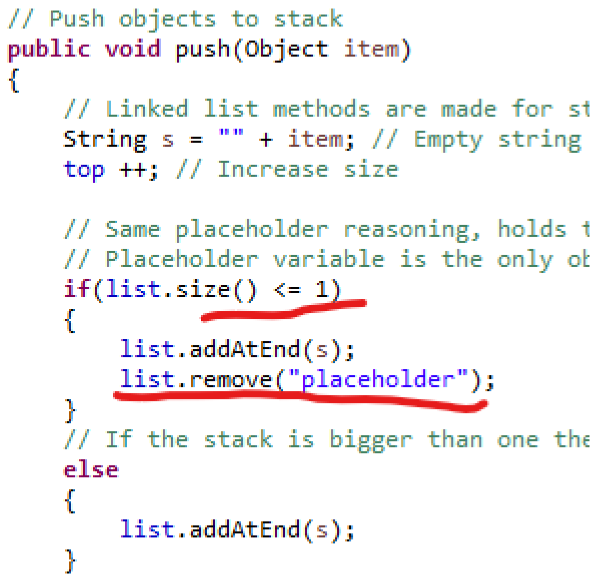
Then I realized that for the node to not be null, there has to be a node to read/get data about in the linked list. To do this I used a placeholder node that would be added to the linked list whenever the list was too small and removed whenever the list was big enough to be read properly. The user wouldn't see this node in the stack since the top variable in the stack class which records the size of the stack would be different from the linked list actual size.



```
// Constructor
public StackList()
{

    list = new LinkedList();
    top = 0; // Stack size
    list.addAtFront("placeholder"); // Placeholder so that the stack doesn't run into a node = null therefore can't run error

}
```



```
// Remove top of Stack
public Object pop()
{
    Object topItem; // Object variable to keep track of th
    topItem = list.getEnd();

    top --; // Decrease size by one

    // Has to be less than or equal to 2 becuase if not th
    // Have to use placeholder to prevent the node from re
    if(list.size() <= 2)
    {

        topItem = list.getFront(); // The front of the lis
        list.addAtFront("placeholder"); ~~// Preven~~t null no
        list.remove(list.getEnd()); // Remove the end whicl
```



```
// Push objects to stack
public void push(Object item)
{
    // Linked list methods are made for st
    String s = "" + item; // Empty string
    top ++; // Increase size

    // Same placeholder reasoning, holds t
    // Placeholder variable is the only ob
    if(list.size() <= 1)
    {
        list.addAtEnd(s);
        list.remove("placeholder");
    }
    // If the stack is bigger than one the
    else
    {
        list.addAtEnd(s);
    }
```