

Forum:[Hack Forums](#)

Thread: [Is my code workin OK???](#)

legenda008 - December 27, 2014, 11:22 am

Ok Im new to java so dont start flaiming me with bad comments Im just practicing exercises from "Back to Basics java book"

Here is problem

Write a method called `removeInRange` that accepts four parameters: a `LinkedList`, an element value, a starting index, and an ending index. The method's behavior is to remove all occurrences of the given element that appear in the list between the starting index (inclusive) and the ending index (exclusive). Other values and occurrences of the given value that appear outside the given index range are not affected. For example, for the list (0, 0, 2, 0, 4, 0, 6, 0, 8, 0, 10, 0, 12, 0, 14, 0, 16), a call of `removeInRange(list, 0, 5, 13)` should produce the list (0, 0, 2, 0, 4, 6, 8, 10, 12, 0, 14, 0, 16). Notice that the zeros located at indexes between 5 inclusive and 13 exclusive in the original list (before any modifications were made) have been removed.

And my code I know its slopy and stupid but could u point me my mistakes

```
Code public static LinkedList<Integer> removeInRange(LinkedList<Integer> list,int value,int start,int end){
```

```
    for(int i = list.size();i>=0;i--){
    if(list.contains(value)){
        int num = list.indexOf(value);
        if(num>=start && num<=end){
            list.remove(num);
        }
    }
    }

    return list;
}
```

clrlslwi - December 27, 2014, 11:35 am

You're looping an unnecessary amount of times, and you're checking the values of numbers that aren't even within the range entered. Also, something to note is that when you pass an "object" around, it's actually just a reference to the underlying object. By modifying the list within that

method, you're modifying the original list as well, therefore there's no reason to return the list.

Anywho, here's an interesting (and efficient) way to do this:

<http://pastebin.com/jVsc2N97>

Null Byte - December 27, 2014, 11:42 am

(Dec 27, 2014 12:35 PM)Demmonic Wrote: [\[CITE\]](#) You're looping an unnecessary amount of times, and you're checking the values of numbers that aren't even within the range entered. Also, something to note is that when you pass an "object" around, it's actually just a reference to the underlying object. By modifying the list within that method, you're modifying the original list as well, therefore there's no reason to return the list.

Anywho, here's an interesting (and efficient) way to do this:

<http://pastebin.com/jVsc2N97>

That's a great idea of decrementing end and i when the desired element is found. That way the program is faster and doesn't throw any OutOfBounds errors... I would not have thought of that.