NESTED ITERATOR

INTERFACE ITERATOR<T>
    <T> next()
    bool hasNext()

NESTED INTEGER
    boolean isInteger();
    Integer getInteger();
    List<Integer> getList();

[1, [1, [2, 3]], 3, 4, [4, 5]]

IDEA 1: flatten out recursively in a queue – inefficient.

Better Way: Think the iterator way

List <NestedIntegers>

$[\underbrace{1}_{NI_0}, \underbrace{[\overset{NI_0}{2}, \overset{NI_1}{[3, 4]}]}_{NI_1}, \underbrace{5}_{NI_2}, \underbrace{6}_{NI_3}, \underbrace{[7, 8]}_{NI_4}]$ ← iterator()

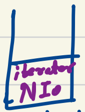
NestedInteger.

iterate our way? stack <Iterator < ↓ >>

Integer cursor; //hasNext() always invoked before next()

next()
    need to return cursor value
    need to clear cursor

hasNext()
```
//goal: put Integer into cursor
while (!stack.empty() && cursor == null)
    top ← stack.peek()
    if !top.hasNext()
        stack.pop(); continue
    NI ← top.next()
    if NI :: isInteger
        cursor ← NI, return true
    else
        stack.push (NI.iterator())
return false
```

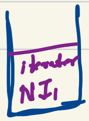


constructor

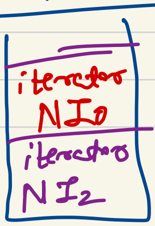
hasNext
NI←NI0
cursor←1

next()
res←cursor
clear cursor
return res.

hasNext



hasNext
NI←NI1
List.
∴ iterator NI1
pushed stack



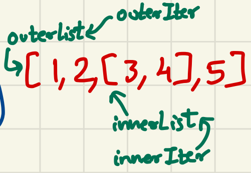
YASH PRADHAN
STUDY NOTES

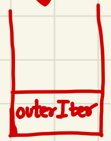
Nested Iterator

```
Stack<Iterator<NestedInteger>> stack
Integer cursor

//constructor
NestedIterator(nestedList)
    stack ← new Stack<>()
    if nestedList != null
        stack.push(nestedList.iterator())

// next: evict cursor
Integer next()
    value ← cursor
    cursor ← null
    return value
```

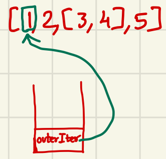




```
boolean hasNext()
    hasNext ← false
    while (true)
        if isEmpty(stack) && cursor is null
            break
        topIterator ← stack.peek()
        if topIterator.hasNext() is false
            stack.pop()
        else
            nestedInteger ← topIterator.next()
            if nestedInteger is Integer
                cursor ← nestedInteger.getInteger()
                hasNext ← true
                break
            else
                innerList ← nestedInteger.getList()
                stack.push(innerList.iterator())
```

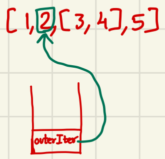
① hasNext()
outerIter gives $\boxed{1}$
nestedInteger ← $\boxed{1}$
cursor ← 1
hasNext ← true

next() // 1
evict cursor



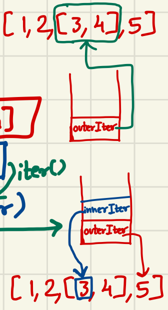
② hasNext()
outerIter gives $\boxed{2}$
nestedInteger ← $\boxed{2}$
cursor ← 2
hasNext ← true

next() // 2
evict cursor



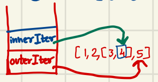
③ hasNext()
outerIter gives $\boxed{[3,4]}$
nestedInteger ← $\boxed{[3,4]}$ iter()
its a list
stack.push(innerIter)
— while iterates —
innerIter gives $\boxed{3}$
nestedInteger ← $\boxed{3}$
cursor ← 3
hasNext ← true

next() //3
evict cursor



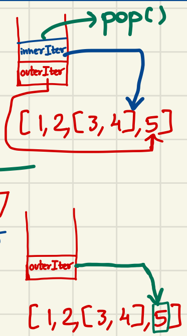
④ hasNext()
innerIter gives $\boxed{4}$
nestedInteger ← $\boxed{4}$
cursor ← 4
hasNext ← true

next() //4
evict cursor



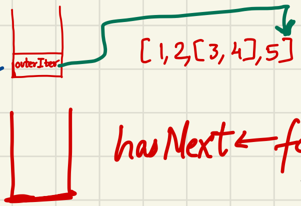
⑤ hasNext()
innerIter out of elements
stack.pop()
— while iterates —
outerIter gives $\boxed{5}$
nestedInteger ← 5
cursor ← 5
hasNext ← true

next() //5
evict cursor



hasNext()
outerIter out of elements
stack.pop()
— while —
stack empty
cursor null

hasNext ← false



Yash Pradhan
Study Notes