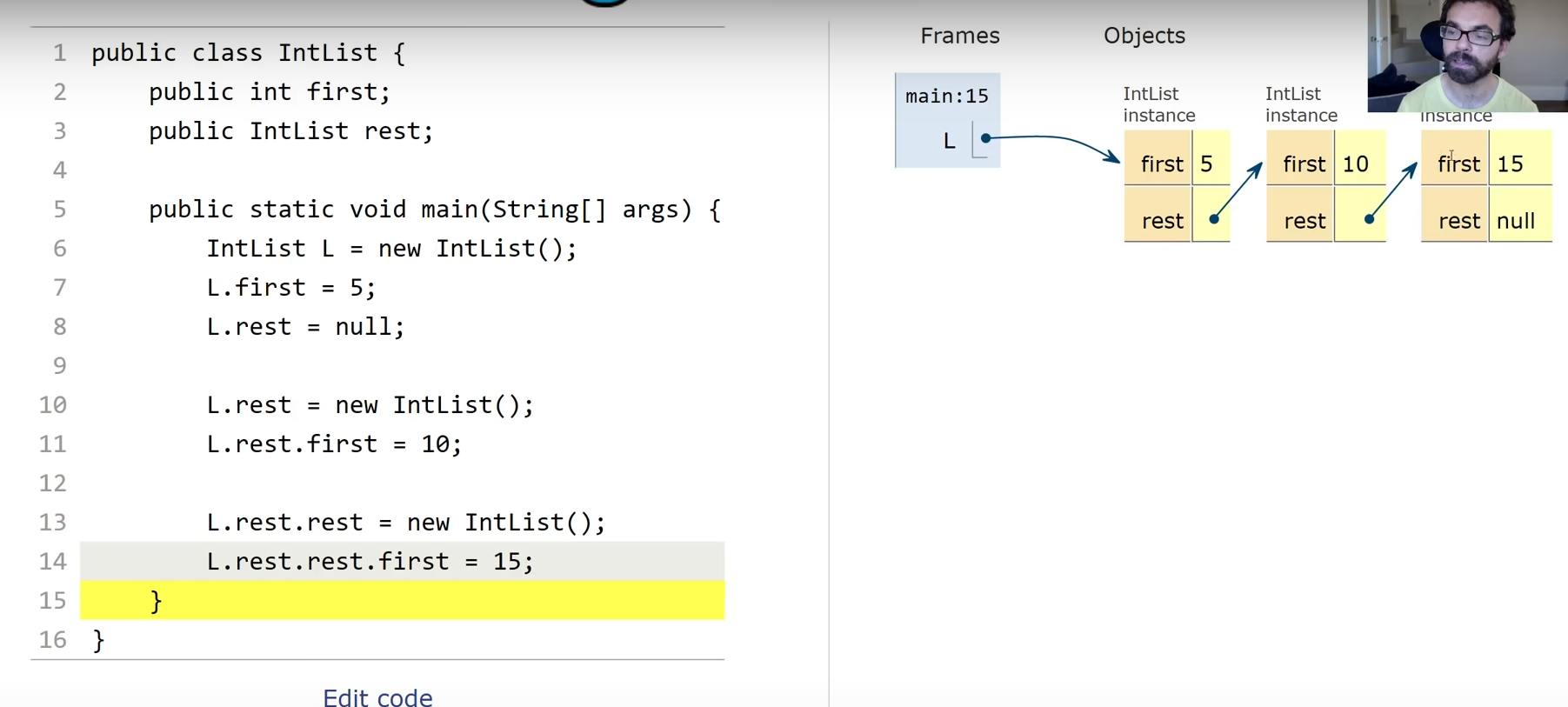
ZS

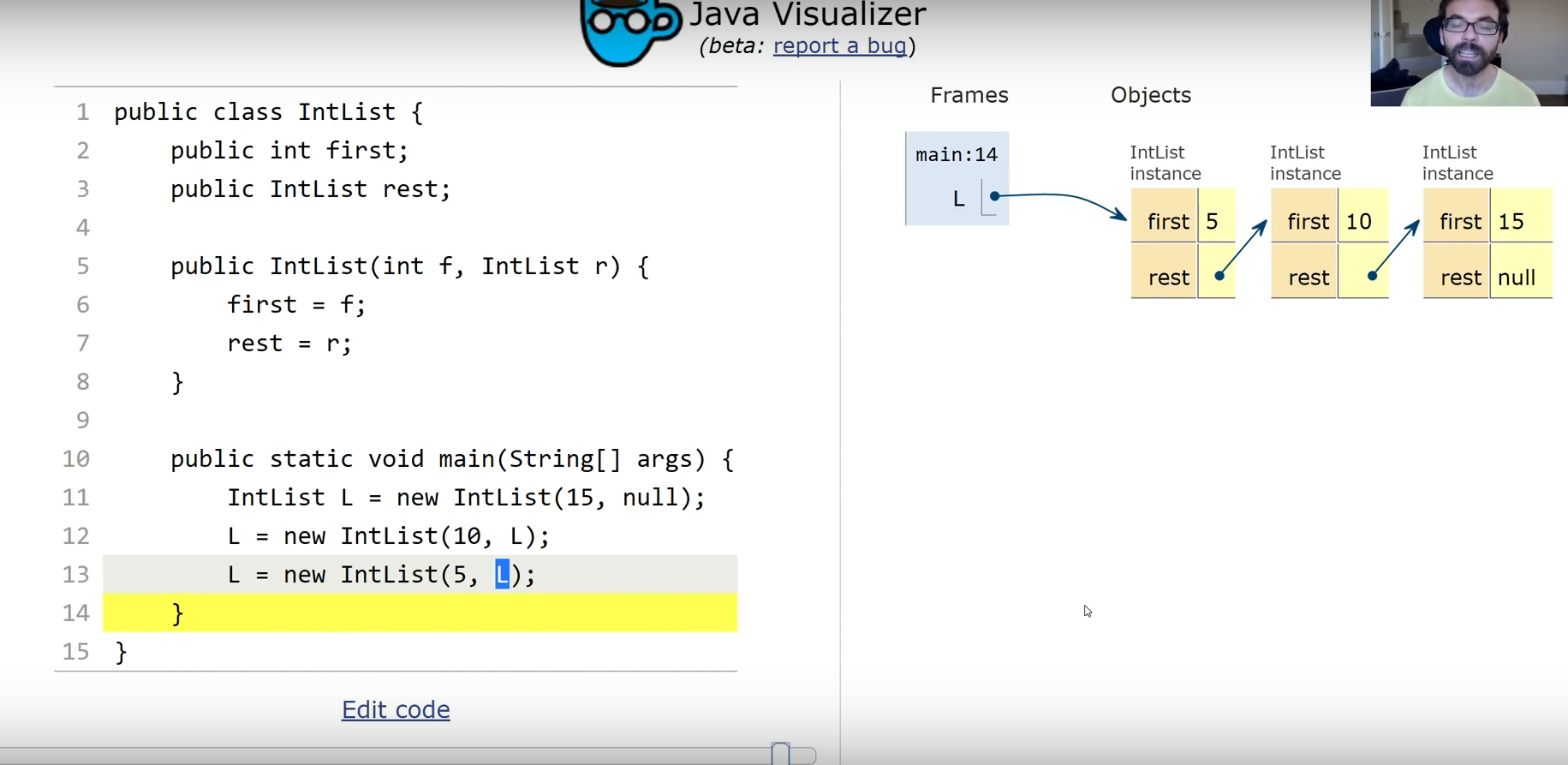
Linked List

--IntList

1. **Create an IntList where it has many InLists in within**



See above ,the reference type IntList is null when it’s declared but not instantiated.



The box of a reference type restores the address of an object. So the second line in the main method creates an object where one of the instance variable points to the original object L.

Interpretation: L becomes an new object, where its “rest” compartment copies the address of the previous object L.

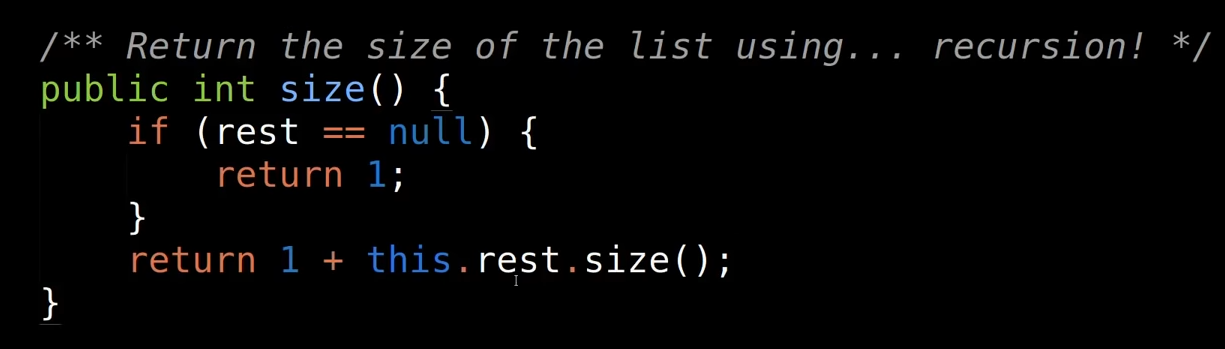
These two different blocks of codes do the same thing.

1. **How to calculate the size of L?**

Method 1:

Recursion （e.g X\_n+1 = X\_n + 8, there is an x in the representation of x, except for the base case X\_0）

So in this method size(), we call the method size() too.



Key word: Base case starts from the right hand side, and it returns 1.

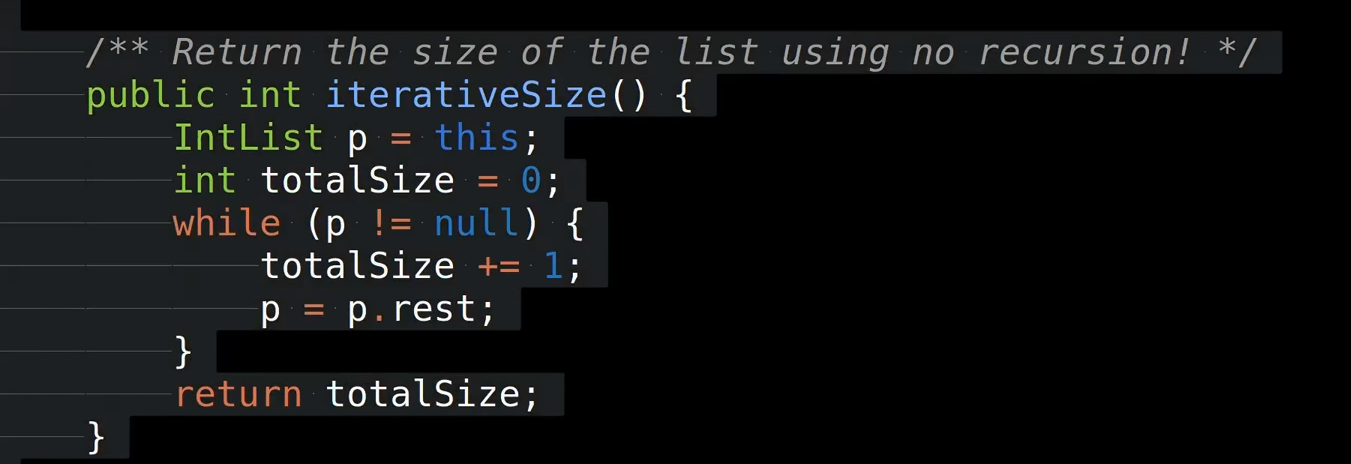
过第一个if filter. All the cases on the left hand side will have nothing to return. Only the base case returns 1.

Return （1+后面case’s size）

After we know the size of base case = 1, we know the second last case’s size = 2.反推到第一个case and returns the final size of L.

Method 2:

Iteration

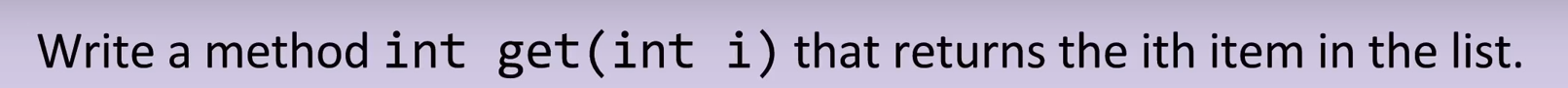


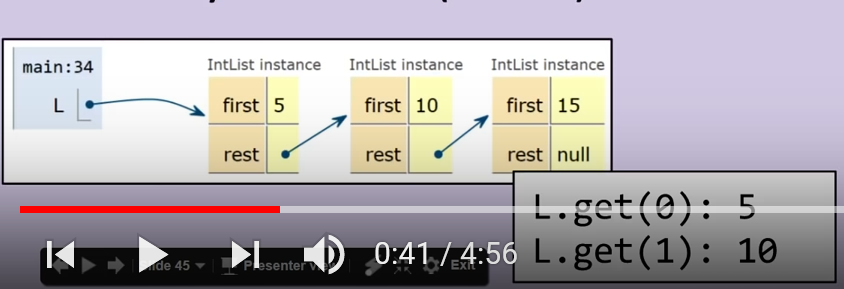
P 是左边第一个object L（p=this, this.iterativeSize() 🡪 因为是算L的size，所以p=this=L）

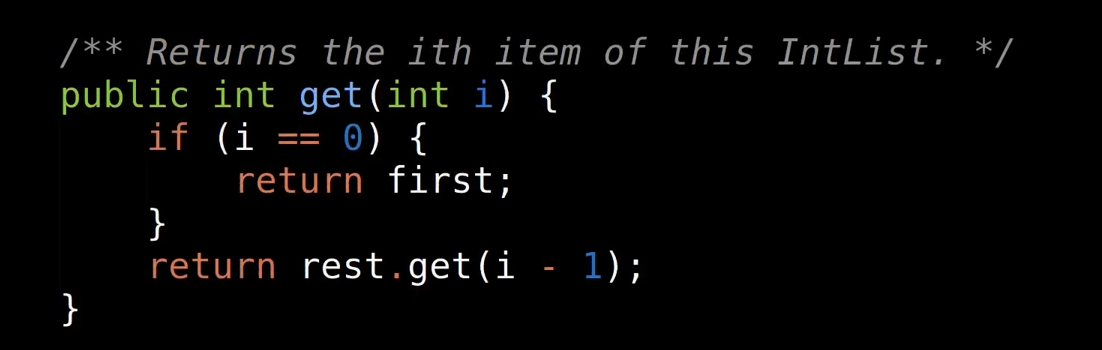
因为除了最后一个 IntList object is null, 其余的IntList都不是Null。所以我们从最左边，即是第一个object开始算size， and starts with p.

P=P.rest 解读：变成下一个IntList object。直到Loop到最后一个IntList object which is null，while loop结束。 Return p

**3.Exercise, get()**







This is another recursion example.

Look at the return, you see the get() method itself.

For the first statement, only the first case has return and it returns the “first” compartment of the object.

So think of get(1), it returns L.rest.get(0) 🡪 (from the if statement, we know get(0) returns “first”) L.rest.first 🡪 which is the “first” compartment of the second IntList object.

Thinking of get(2), it returns L.rest.get(1) 🡪 为解决get(1)，回到get method，因为1 != 0, so returns get(0) 🡪 we know get(0) return “first”.

So get(2) = L.rest.get(1) = L.rest.rest.get(0) and get(0) here is just a way to take out the value in the “first” compartment.

8/1/2020 新理解：呢个思路似while loop，只不过用Recursion写成。

1. get(int i)的终止条件：递归应该在i = 0 的时候结束

IntList p = this;

for(int j = i; j >= 0; j--）{

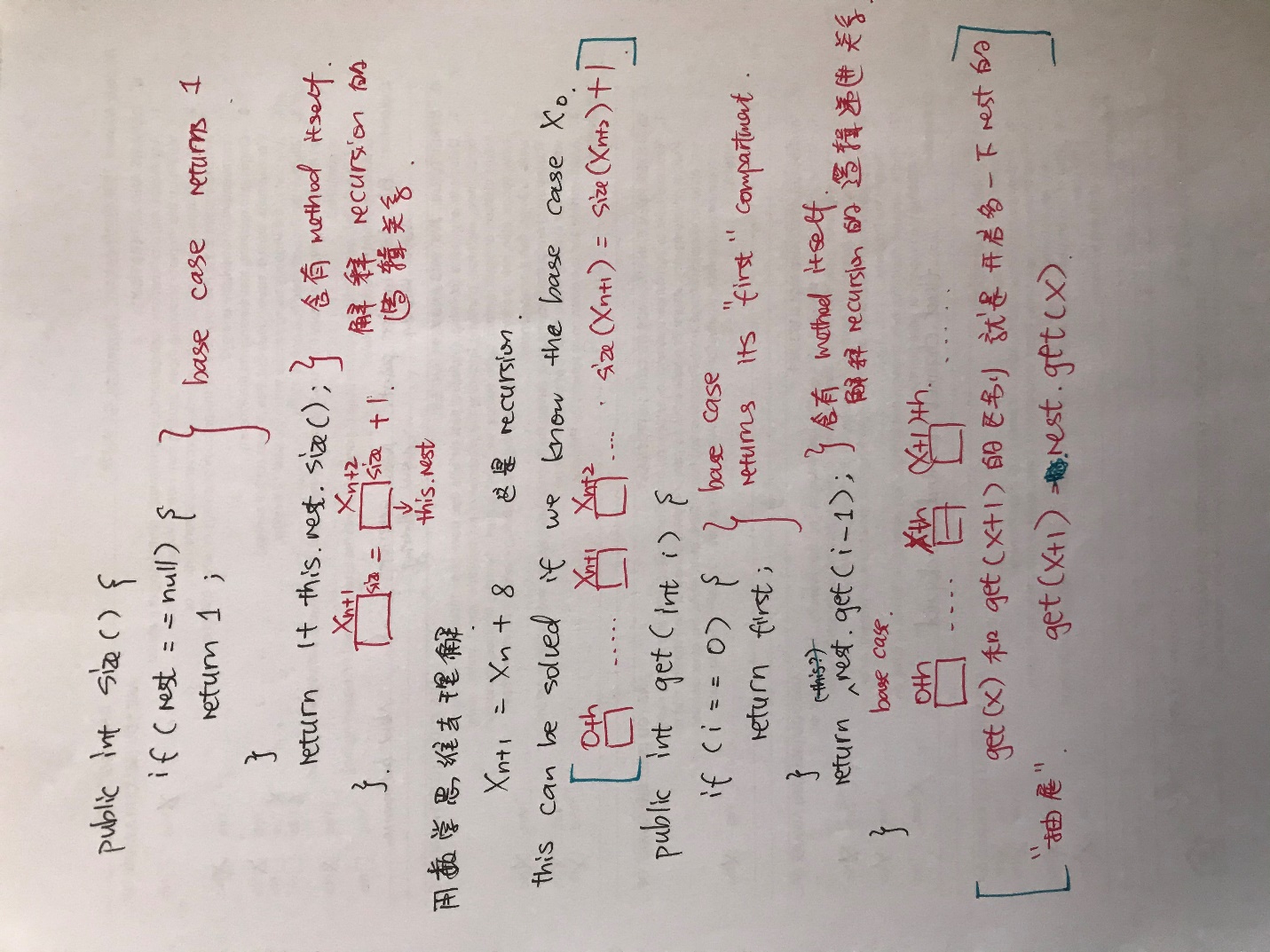
int returnItem = p.first;

p = p.next; }

return returnItem;

1. 返回值：第ith个IntList的item
2. 要i = 0，每次调用函数的时候，i--, 所以i – 1.

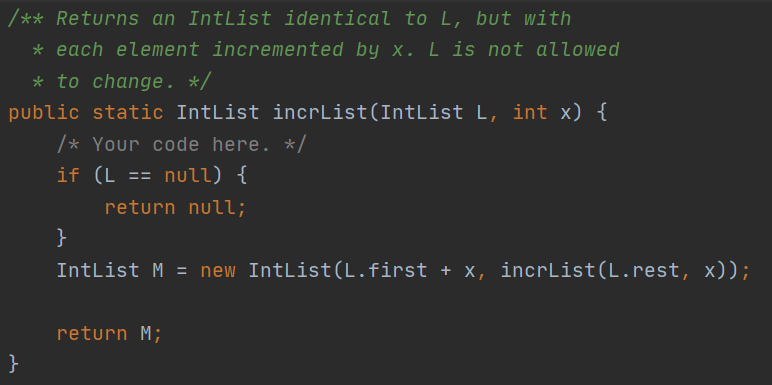
See总结和理解

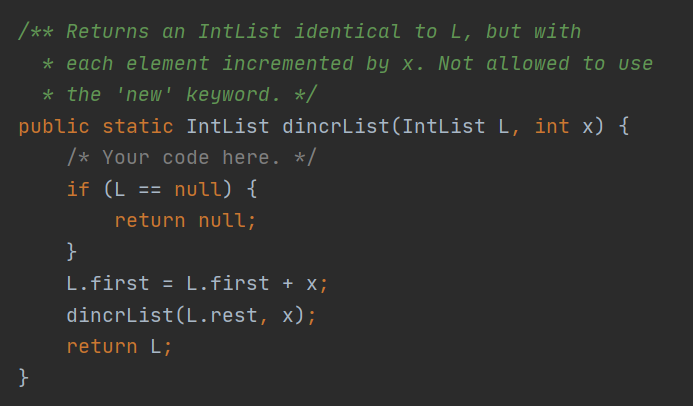


在get() code那里，get(x)和get(x+1)的区别是从L那里开多一下抽屉.

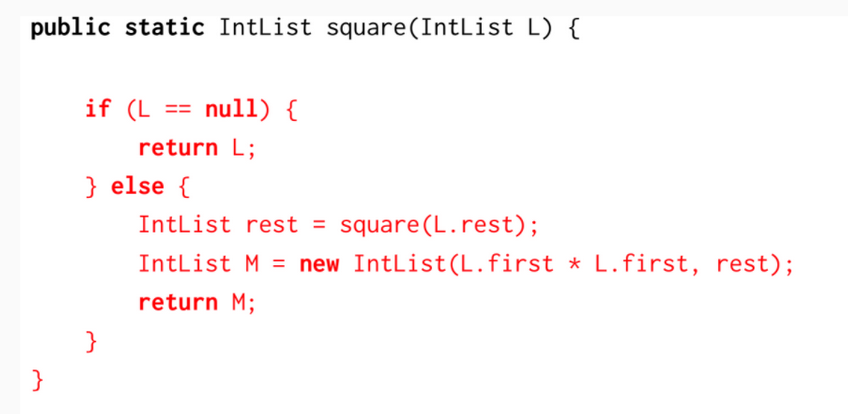
So get(x+1) = L.rest.get(x) “this” can be added.

1. **Create an IntList identical to L**



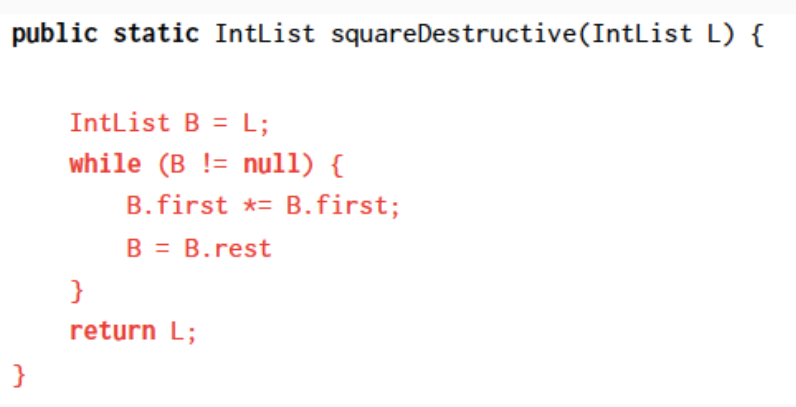


1. Create an IntList identical to L with squared values

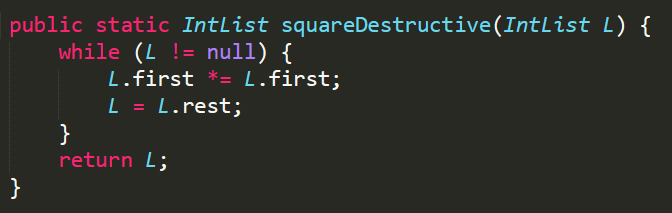


这个跟4.1 一样思路。

而且else那里可以直接写成 IntList M = new IntList(L.first \* L.first, square(L.rest));



这里一开始我是写成



但我的解法，最后L会points to the last object in the Linked List, instead of the beginning of the list。所以要用an IntList B points to L，然后把L的所有first都改掉，最后return L, which still points to the beginning of the list.