

In the doStuff method, W copies the bits of the **address** of walrus (because the box of walrus stores the address, not the value), so W also points to the object walrus. When doStuff alter W, walrus is altered as well. The address of W is 64 bits, while the size of the object of W is 96 bits, because int is 32 bits and double is 64 bits.

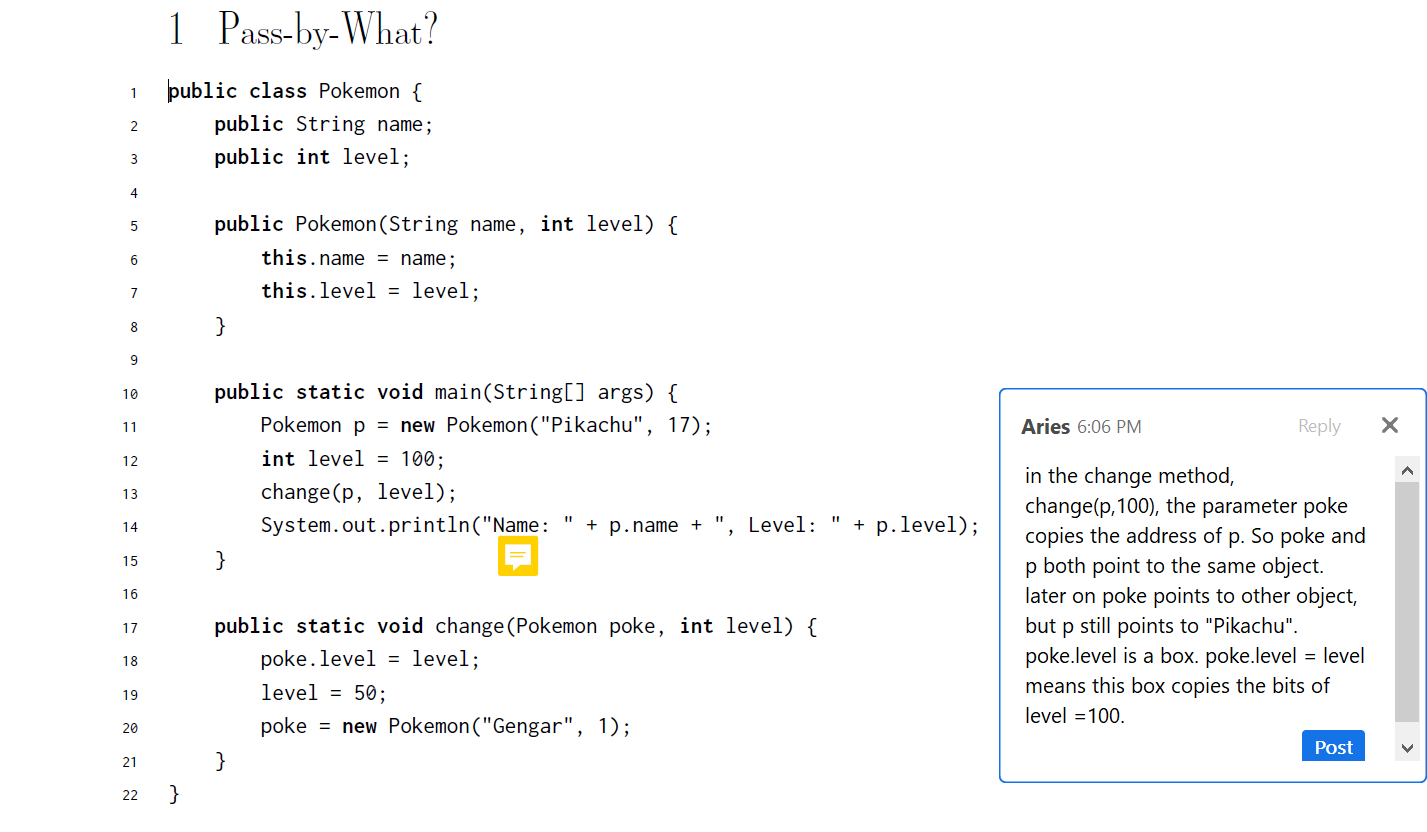
In the doStuff method, x copies the value of x in main method (because the box of x in main stores the value). Therefore, when we alter x in doStuff, it only affects the value of x in doStuff.

Declaration

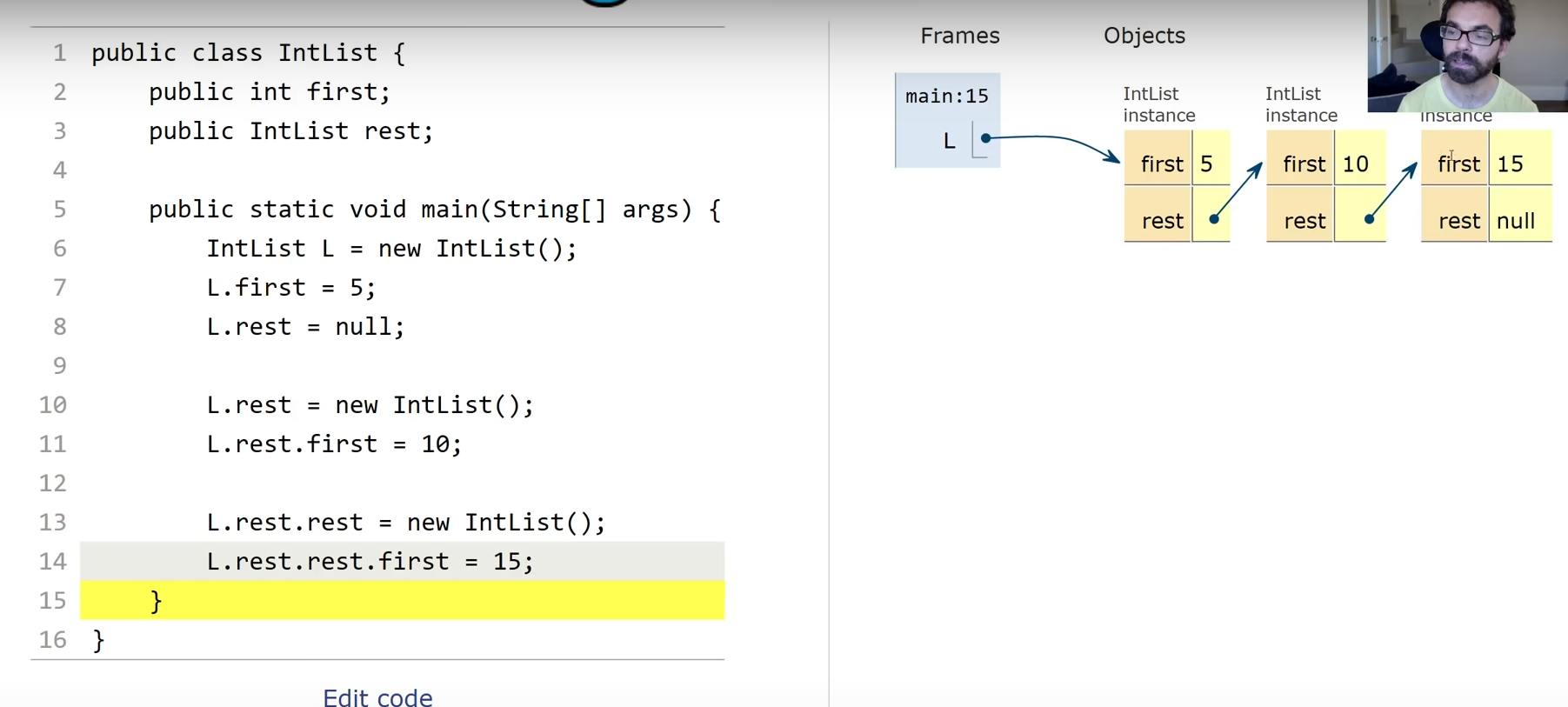
For primitive type --- according to its type(int, double, etc) having a specific size of box for the variable. Box storing a value.

For reference type --- Java allocates exactly a box of 64 bits, either null or an address (as an arrow).

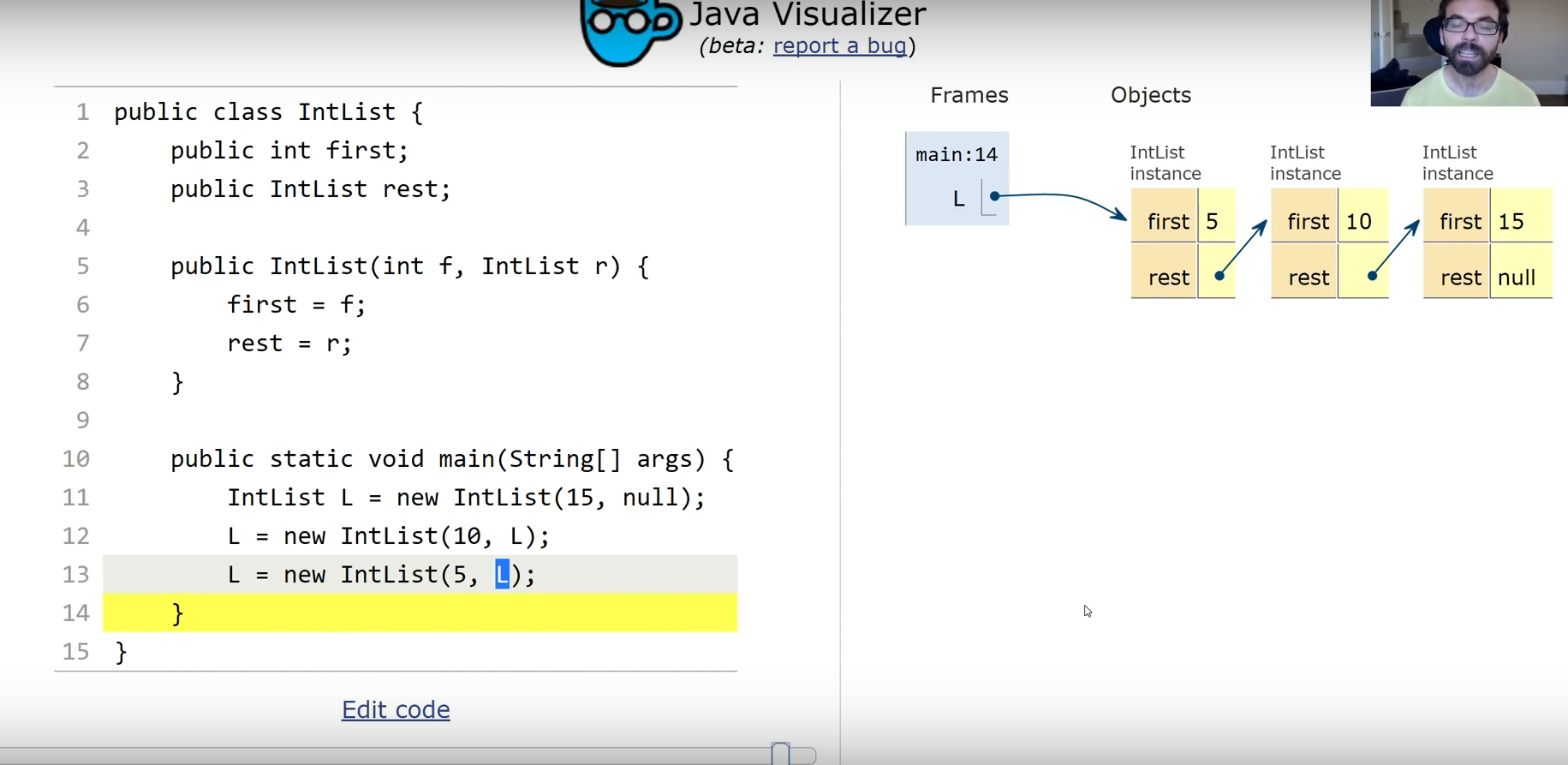
Box storing an address



**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.

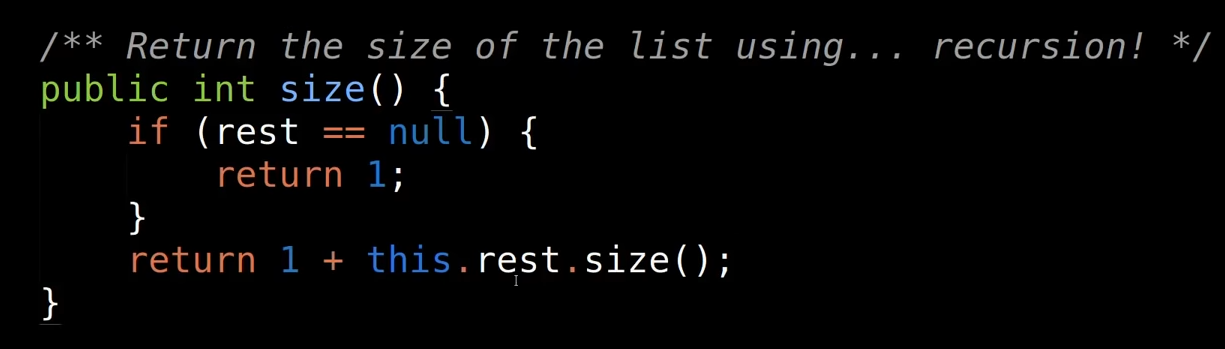
These two different blocks of codes do the same thing.

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.

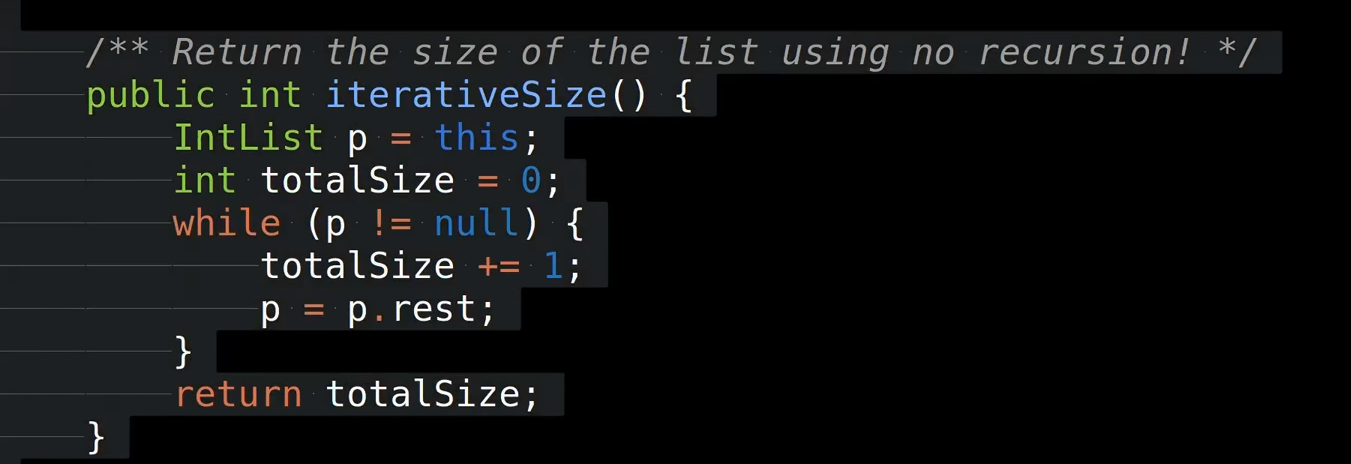
过第一个if filter. All the cases on the left hand side will have nothing to return. Only the base case returns 1.

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

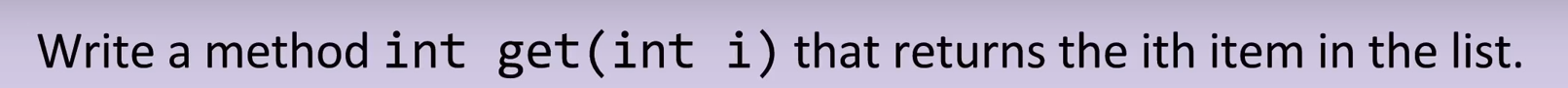


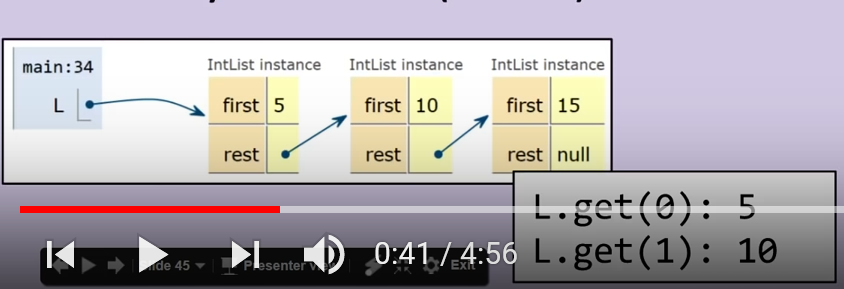
P 是左边第一个object L（p=this, this.iterativeSize() 🡪 因为是算L的size，所以p=this=L）

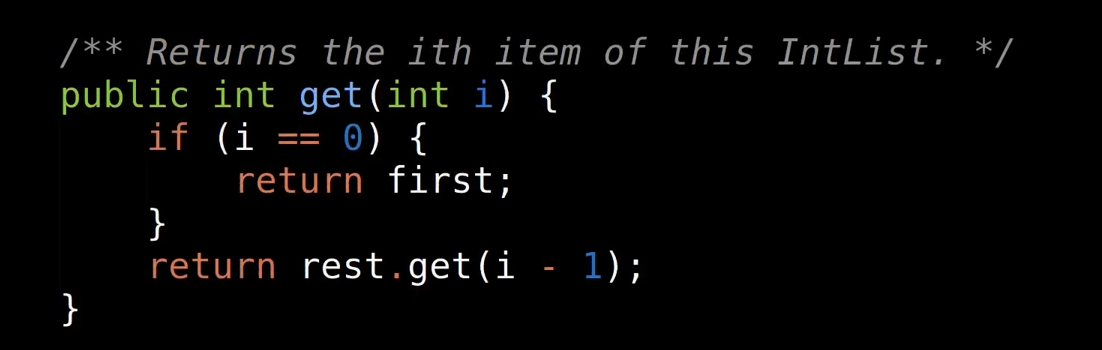
因为除了最后一个 IntList object is null, 其余的IntList都不是Null。所以我们从最左边，即是第一个object开始算size， and starts with p.

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

Exercise







This is another recursion example.

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

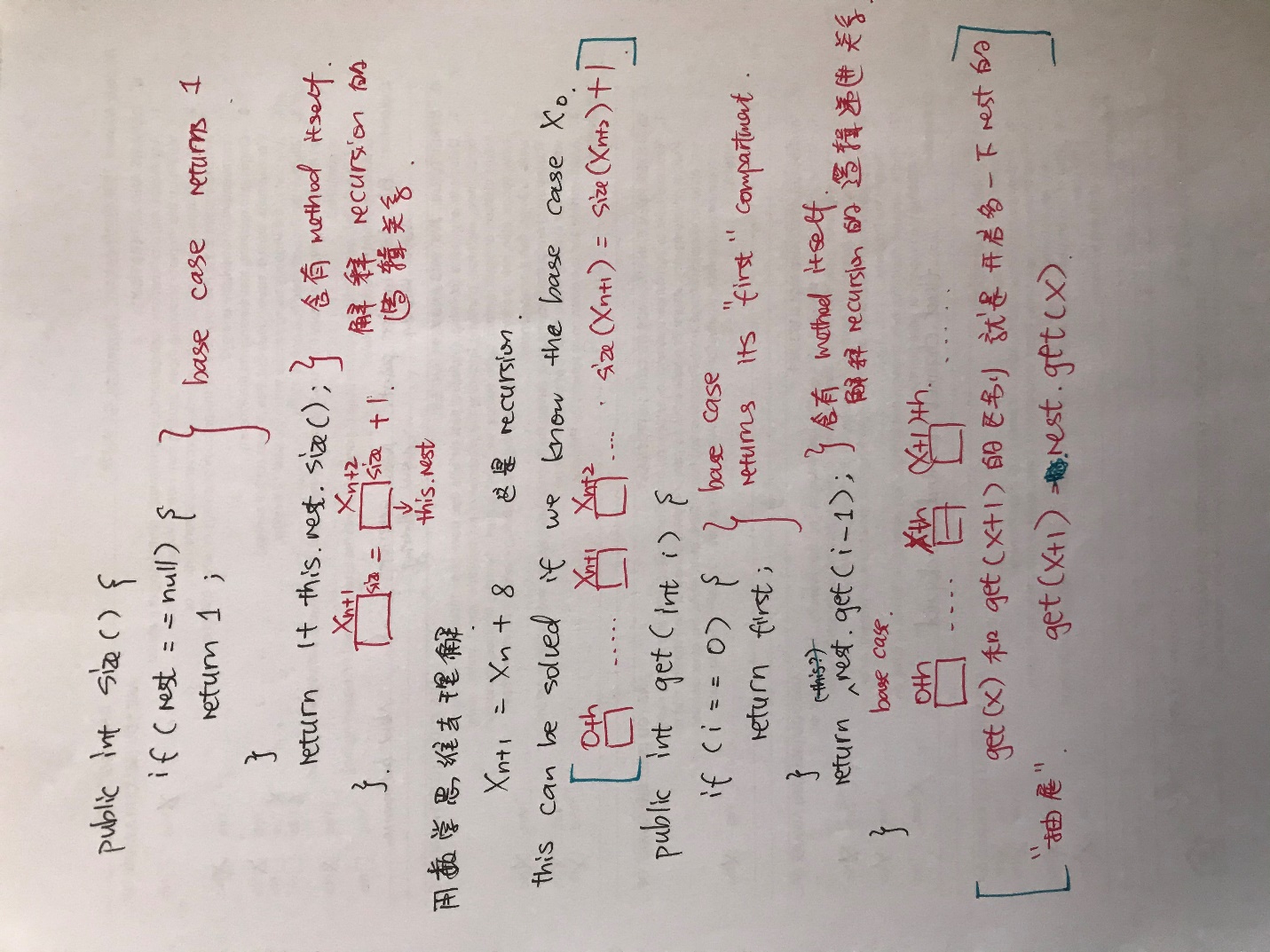
For the first statement, only the first case has return and it returns the “first” compartment of the object.

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.

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.