Table 1

Pointers

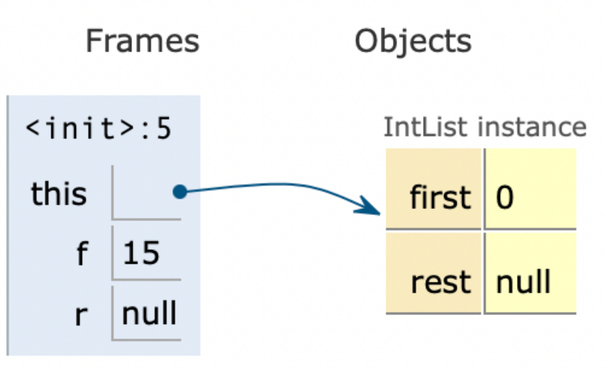
recursion code

```
public static IntList incrList(IntList L, int x) {
      if (L == null) {
          return L;
      }
      IntList incr = new IntList(L.first + x, incrList(L.rest, x));
      return incr;
  }
```

CS61B lists1/exercises/ExtraIntListPractice.java - iterative method

new creates a box with 64bits- address in it, which points to the object with default value;
When we **instantiate** an Object (e.g. Dog, Walrus, Planet): Java first allocates a box of bits for each instance variable of the class and fills them with a default value (e.g. 0, null).
parameter passing: copy the bits into the new scope

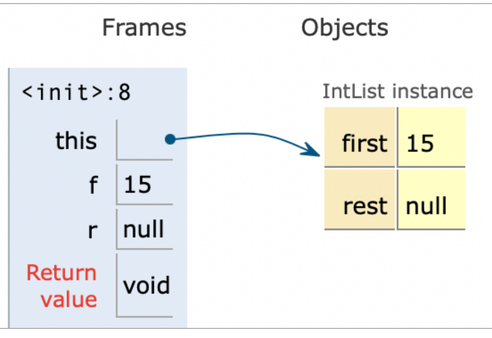
```
public static void main(String[] args) {
    IntList L = new IntList(15, null);
    L = new IntList(10, L);
    L = new IntList(5, L);
```



constructor

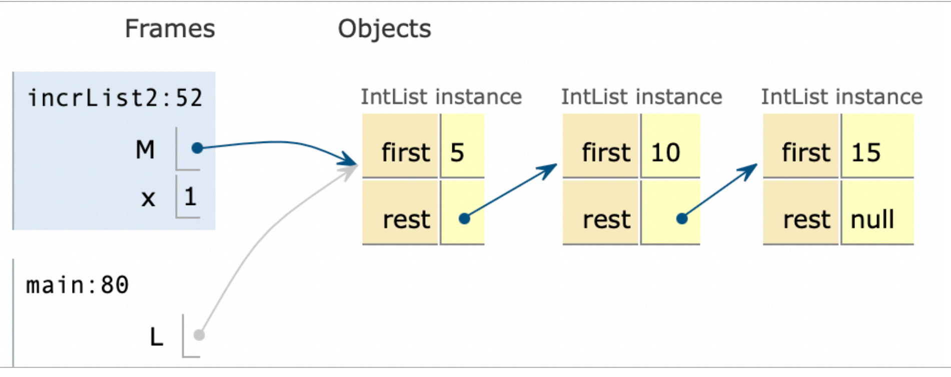
after going through the constructor: The constructor then usually fills every such box with some other value.

```
public IntList(int f, IntList r) {
     first = f;
     rest = r;
}
```



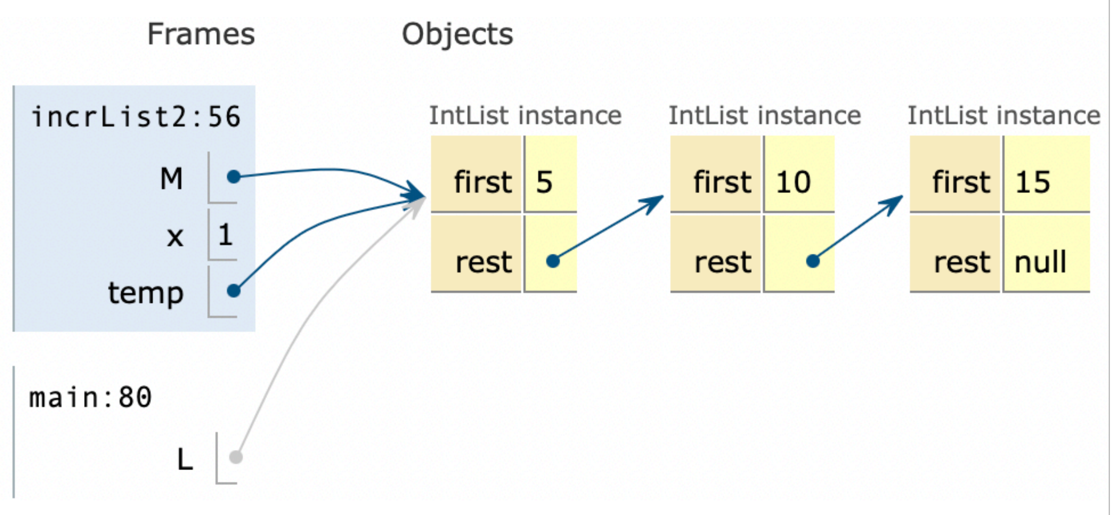
```
public static void main(String[] args) {
    IntList L = new IntList(15, null);
    L = new IntList(10, L);
    L = new IntList(5, L);

    IntList N = IntList.incrList2(L, 1);
     System.out.println(L.get_Recur(0));
```

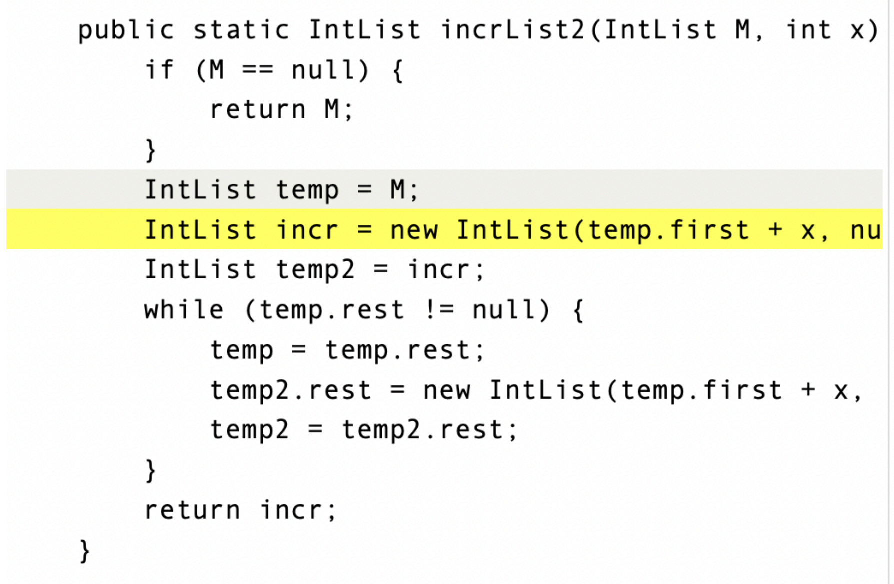


pointer temp

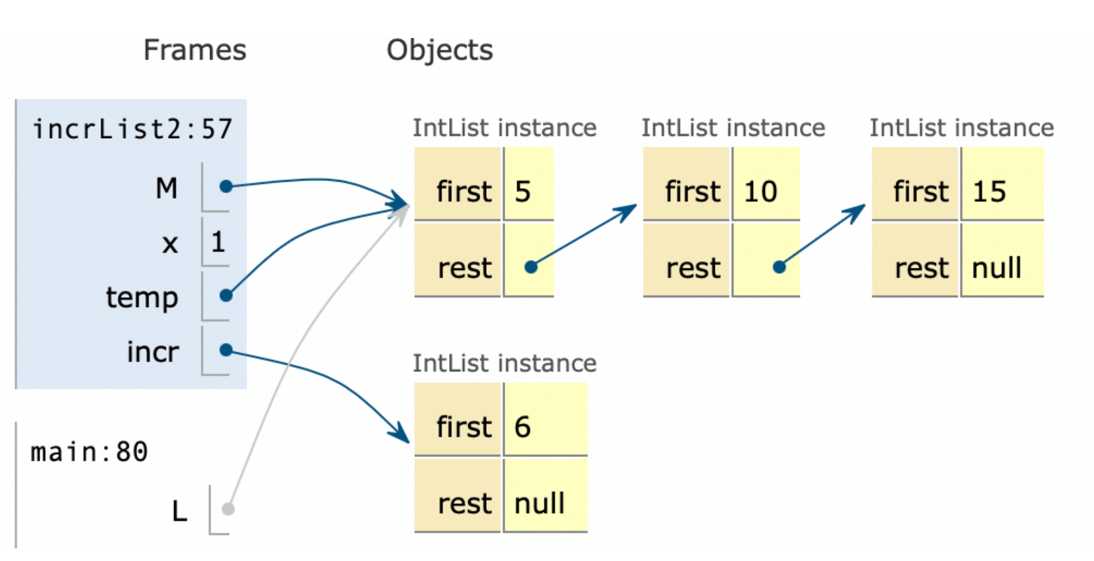
```
public static IntList incrList2(IntList M, int x)
    if (M == null) {
        return M;
    }
    IntList temp = M;
    IntList incr = new IntList(temp.first + x, nu
    IntList temp2 = incr;
    while (temp.rest != null) {
        temp = temp.rest;
        temp2.rest = new IntList(temp.first + x,
        temp2 = temp2.rest;
    }
    return incr;
}
```



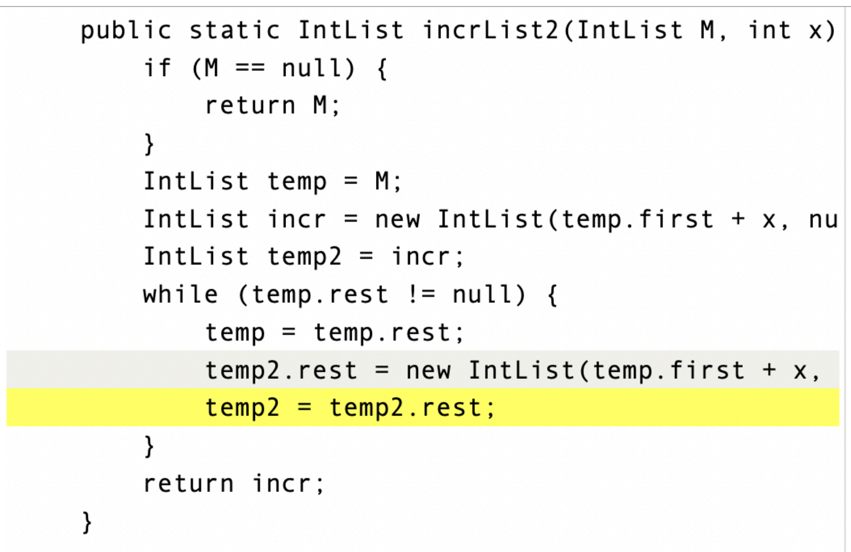
创建new IntList object



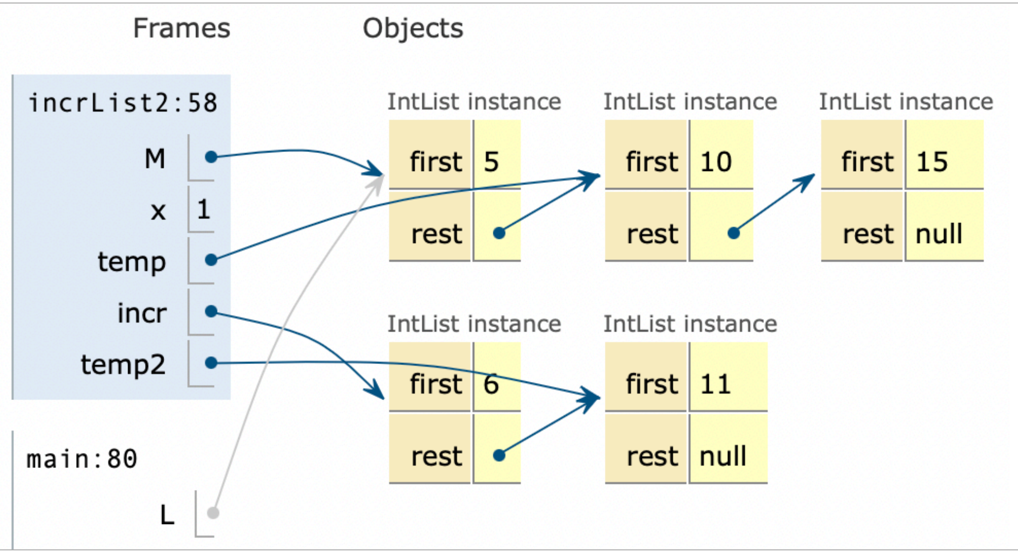
```
public static IntList incrList2(IntList M, int x)
    if (M == null) {
        return M;
    }
    IntList temp = M;
    IntList incr = new IntList(temp.first + x, nu
    IntList temp2 = incr;
    while (temp.rest != null) {
        temp = temp.rest;
        temp2.rest = new IntList(temp.first + x,
        temp2 = temp2.rest;
    }
    return incr;
}
```



第二个pointer temp2



```
public static IntList incrList2(IntList M, int x)
    if (M == null) {
        return M;
    }
    IntList temp = M;
    IntList incr = new IntList(temp.first + x, nu
    IntList temp2 = incr;
    while (temp.rest != null) {
        temp = temp.rest;
        temp2.rest = new IntList(temp.first + x,
        temp2 = temp2.rest;
    }
    return incr;
}
```



不改变原list， 上面two pointer 的优化	改变原list

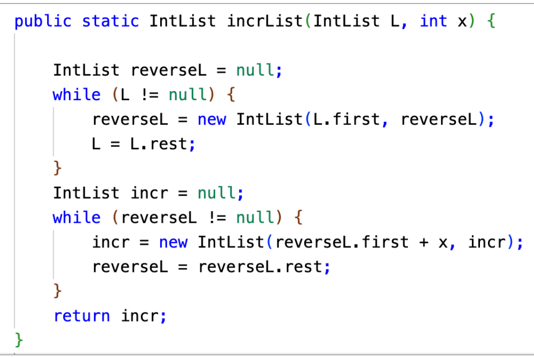
public static IntList incrList2(IntList L, int x) {
if (L == null) {
return L;
}

IntList incr = new IntList(L.first + x, null);
IntList temp2 = incr;
while (L.rest != null) {
L = L.rest;
temp2.rest = new IntList(L.first + x, null);
temp2 = temp2.rest;
}
return incr;
}

**```java**
public static IntList incrList(IntList L, int x) {
IntList p = L;
while (p != null) {
p.first += x;
p = p.rest;
}
return L;
}

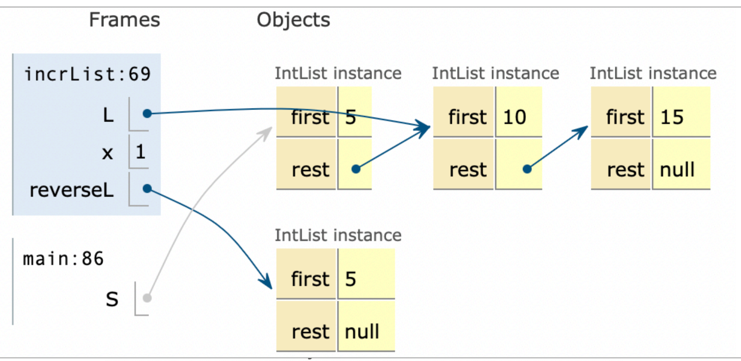
看pointer的移动

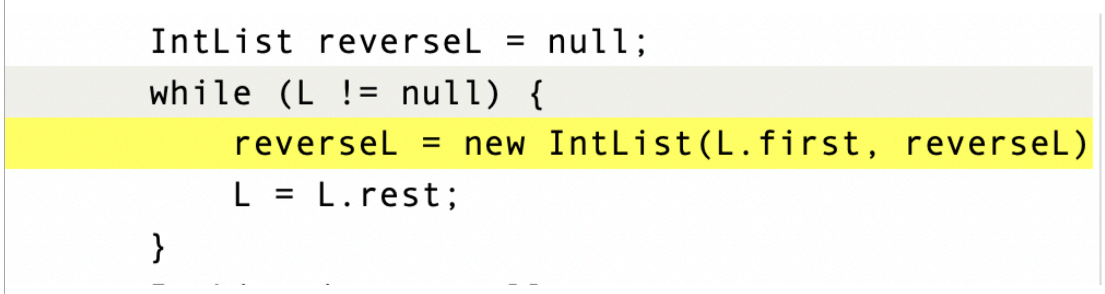
code



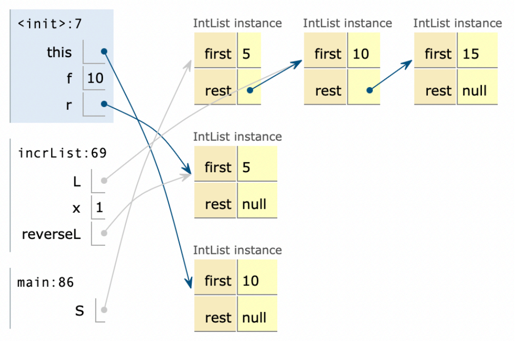
```
public static IntList incrList(IntList L, int x) {

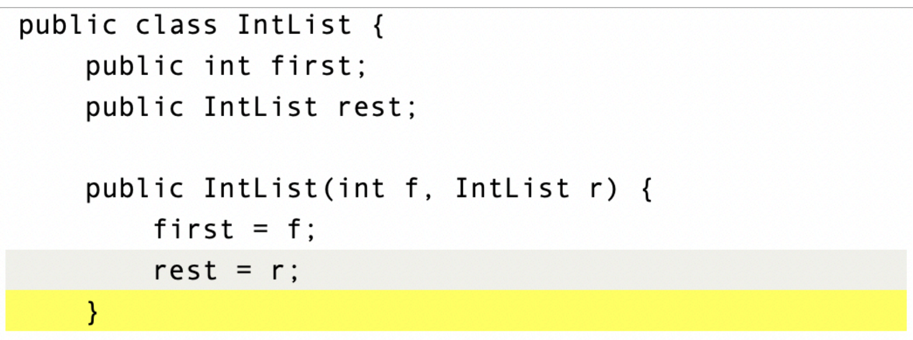
    IntList reverseL = null;
    while (L != null) {
        reverseL = new IntList(L.first, reverseL);
        L = L.rest;
    }
    IntList incr = null;
    while (reverseL != null) {
        incr = new IntList(reverseL.first + x, incr);
        reverseL = reverseL.rest;
    }
    return incr;
}
```





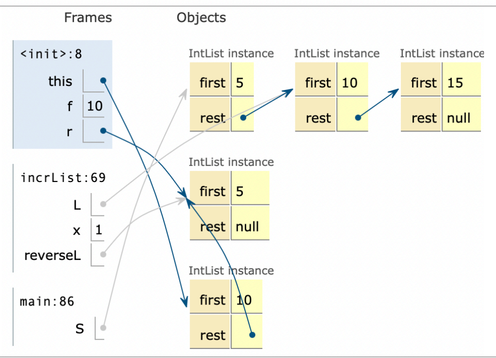
```
    IntList reverseL = null;
    while (L != null) {
        reverseL = new IntList(L.first, reverseL)
        L = L.rest;
    }
```

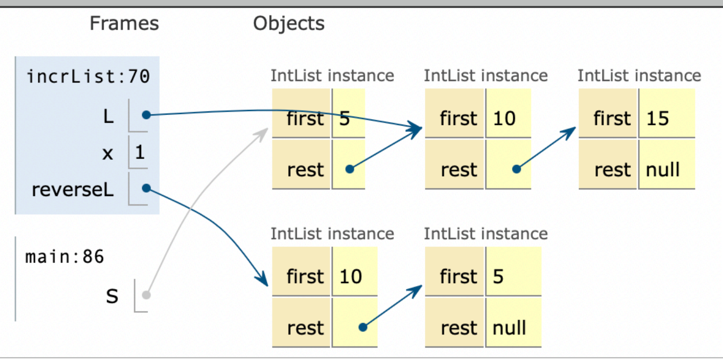




```
public class IntList {
    public int first;
    public IntList rest;

    public IntList(int f, IntList r) {
        first = f;
        rest = r;
    }
```



reverseL = new IntList(L.first, reverseL); 这一步的pointer有两步 1. new object.rest point to reverseL 2. object address is returned by new and stored in reversel: reverseL point to new object	 Frames Objects incrList:70 L x 1 reverseL main:86 S IntList instance first 5 rest IntList instance first 10 rest IntList instance first 15 rest null IntList instance first 10 rest IntList instance first 5 rest null