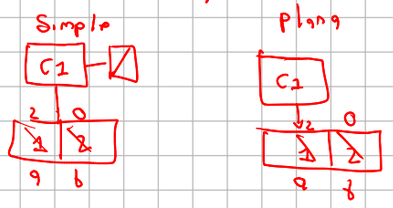
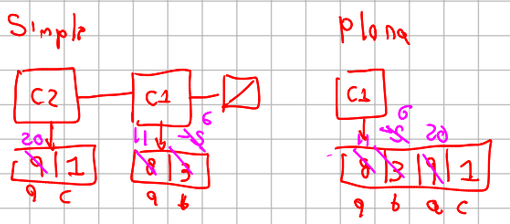
```
class c1 extends object
    field a
    field b

    method initialize(x, y)
        begin
            set a = x;
            set b = y
        end

    method p(x, y)
        begin
            set a = +(a, x);
            set b = -(b, y);
            send self r(a, b)
        end

    method r(x, y)
        +(+(x, y), *(a, b))

class c2 extends c1
    field a
    field c

    method initialize(x, y)
        begin
            super initialize(x, y);
            set a = +(x, 1);
            set c = -(y, 2)
        end

    method r(x, y)
        begin
            set a = +(a, x);
            set b = -(c, y);
            -(x, y)
        end

class c3 extends c2
    field d

    method initialize(x, y)
        begin
            super initialize(+(x, y), -(x, y));
            set d = +(x, y)
        end

    method p(x, y)
        send self r(x, +(y, a))

    method r(x, y)
        +(+(x, y), -(a, b))

let
    o1 = new c1(1, 2)
    o2 = new c2(8, 3)
    o3 = new c3(1, 7)
in
    let
        u = send o1 p(1, 2)
        v = send o2 p(3, 8)
        z = send o3 p(5, 6)
    in
        +(u, +(v, z))
```

Representación

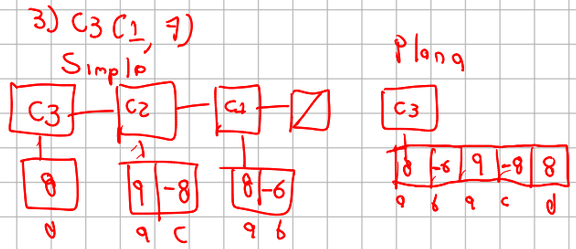
1) c1(1, 2)



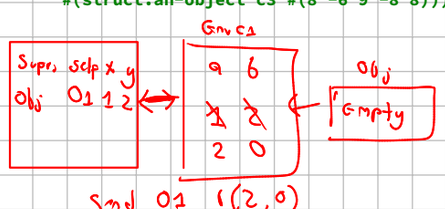
2) c2(8, 3)



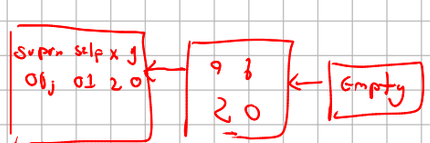
3) c3(1, 7)




```
((#(struct:a-part c1 #(1 2)))
 (#(struct:a-part c2 #(9 1)) #(struct:a-part c1 #(8 3)))
 (#(struct:a-part c3 #(8)) #(struct:a-part c2 #(9 -8))
    #(struct:a-part c1 #(8 -6))))

(#(struct:an-object c1 #(1 2))
 #(struct:an-object c2 #(8 3 9 1))
   #(struct:an-object c3 #(8 -6 9 -8 8)))
```

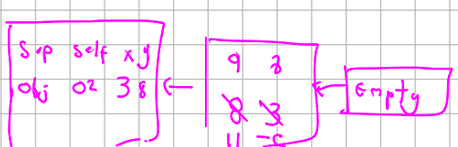


Send o1 1(2, 0)



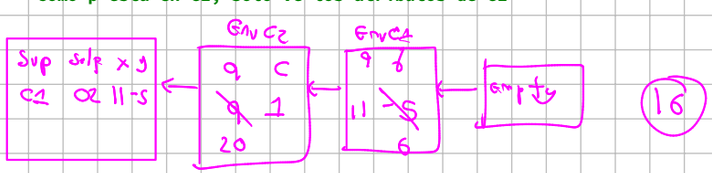

```
+(+(x, y), *(a, b))
```
2 = 2

p(3, 6) no está en c2 sino en c1



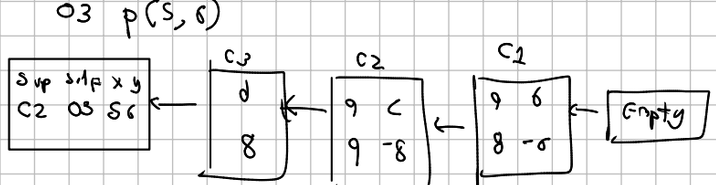

```
send self r(a, b)
send o2 r(11,-5)
```

Como p está en c1, solo ve los atributos de c1

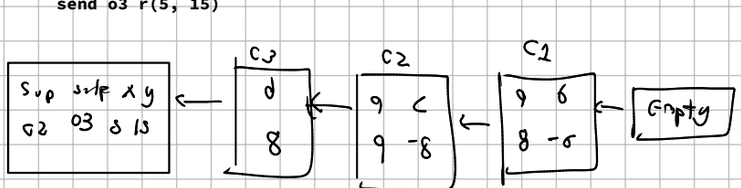


16

o3 p(5, 6)




```
send self r(x, +(y, a))
send o3 r(5, 15)
```




```
+(+(x, y), -(a, b))
+(20, 15) = 35
```