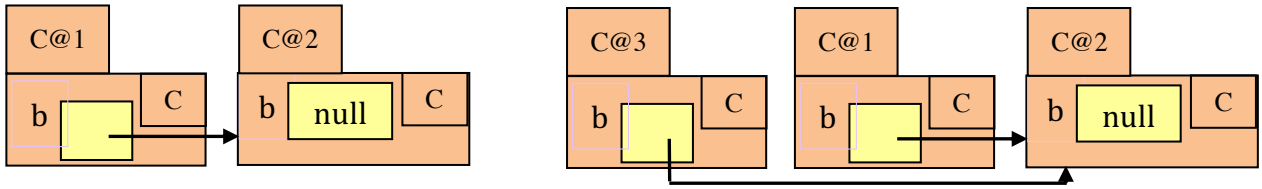
An introduction to cloning

Cloning refers to the process of duplicating something exactly. On 5 July 1996, Dolly was born as a clone of another sheep. She had three mothers: one provided the egg, another provided the DNA, and the third carried the cloned embryo to term. Dolly was the first animal clone, a sheep that was the exact duplicate of another sheep — see en.wikipedia.org/wiki/Dolly_(sheep).

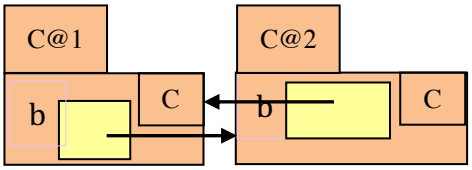
In Java, to clone an object means to make an exact duplicate of it. The original and the clone should be independent: changing one shouldn't affect the other. For example, consider cloning object C@1 below to the left, calling its duplicate *C*@3, shown to the right. Objects *C*@1 and its duplicate *C*@3 on the right are not completely independent because changing field *C*@1.b.b also changes field *C*@3.b.b. This is a *shallow copy*.



To clone *C*@1 shown below on the left, create *C*@3 and *C*@4 on the right. The "data structure" *C*@1 is exactly the same as the data structure *C*@3. In fact, if function *equals* is defined in class *C*, *C*@1.*equals*(*C*@3) should evaluate to true. Further, the two data structures are independent; changing one has no effect on the other. In effect, then, we clone not just object *C*@1 but the whole "data structure" that starts at *C*@1 —all fields of C@1, all fields of those objects, etc. This would be a *deep copy*.

Watch out for cycles

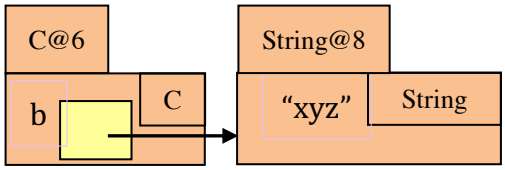
But be careful! Consider cloning *C*@1 shown to the right. There is is a cycle: *C*@1.b points to *C*@2 and *C*@2.b points to *C*@1. Cloning by blindly cloning the object pointed to be each field could result in an infinite loop of cloning. This can be cloned, but it requires care.



In our later discussions of cloning, we will assume that there is no cycle in the data structure to be cloned. For example, our explanation will handle cloning a tree but not a circular linked list or a graph.

No need to copy fields that are immutable or have a primitive type

Consider cloning *C*@6 to the right. Only object *C*@6 needs to be copied and not *String*@8 because the contents of *String*@8 cannot be changed. Strings are *immutable*. If a field contains a pointer to an immutable object, there is no need to clone it. For example, fields that are objects of wrapper classes need not be cloned.



Is the idea of cloning and independence straightforward?

Suppose object *s* implements a stack of elements e_0, …, e_n, where the e_i are objects of some class. The discussion above implies that cloning *s* requires cloning the e_i as well, so that the stack and its clone are independent. But is this necessary? Operations on the stack *never* change the elements in it, so the notion of independence doesn't come into play. It should be enough to copy the data structure that implements the stack, and not its elements.

Perhaps we should ask why stack *s* is being cloned and how *s* and its clone will be used. The answers to such questions will determine how the cloning should be done.In summary, the idea of cloning seems simple, but doing it right will require carefully analyzing the situation.